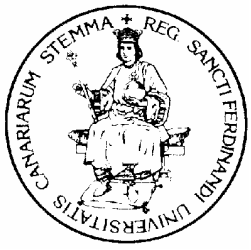
UNIVERSIDAD DE LA LAGUNA

Diseño de sistemas borrosos recurrentes mediante estrategias evolutivas y su aplicación al análisis de señales y reconocimiento de patrones

Autor: Alayón Miranda, Silvia

Directores: Lorenzo Moreno Ruiz
y José Ignacio Estévez Damas

Departamento de Física Fundamental y Experimental, Electrónica y Sistemas



UNIVERSIDAD DE LA LAGUNA

D. Lorenzo Moreno Ruiz, Catedrático de Universidad del Departamento de Física Fundamental y Experimental, Electrónica y Sistemas de la Universidad de La Laguna, y D. José Ignacio Estévez Damas, Doctor por la Universidad de La Laguna y Profesor Asociado del Departamento de Física Fundamental y Experimental, Electrónica y Sistemas de la Universidad de La Laguna,

CERTIFICAN:

Que Silvia Alayón Miranda ha realizado bajo nuestra dirección el trabajo titulado: "Diseño de Sistemas Borrosos Recurrentes mediante Estrategias Evolutivas y su Aplicación al Análisis de Señales y Reconocimiento de Patrones", que presenta para optar al grado de Doctor por la Universidad de La Laguna.

Con esta fecha, autorizamos la presentación del mismo.

La Laguna, a 2 de Junio de 2003

Los Directores

Lorenzo Moreno Ruiz

José Ignacio Estévez Damas

AGRADECIMIENTOS

Es muy poco el espacio con el que cuento, y son muchas las personas a las que quiero agradecer su ayuda y apoyo en la realización de este trabajo. Me gustaría aprovechar estas líneas para mostrar mi agradecimiento a estas personas.

En primer lugar, quisiera agradecer al Dr. D. José Ignacio Estévez Damas su labor de dirección. No tengo palabras para agradecer su esfuerzo, su dedicación y su optimismo durante las duras horas de trabajo en esta tesis. Pero, sobre todo, muchas gracias por su amistad, que ha sido lo mejor que he encontrado desde que he entrado a trabajar en este grupo.

Al Dr. D. Lorenzo Moreno Ruiz, además de agradecerle su labor de dirección y sus útiles consejos y paciencia, me gustaría agradecerle en estas líneas la confianza que siempre ha depositado en mí, no sólo en la realización de esta tesis.

Al Dr. Lucio Díaz Flores y su equipo del Hospital Universitario de Canarias, por su inestimable ayuda y asesoramiento. En especial, me gustaría agradecer al Dr. Lucio Díaz Flores la paciencia, la disponibilidad en cualquier momento y la ilusión que siempre muestra a la hora de colaborar con nosotros.

Trabajar en lo que me gusta y hacerlo en tan buena compañía es una suerte en la vida. Por eso siento la necesidad de agradecer a los miembros del grupo de Computadoras y Control su ayuda y colaboración día a día: Dr. D. Leopoldo Acosta Sánchez, Dra. Dña. Rosa María Aguilar Chinea, Dr.D.Juan Albino Méndez, D. Roberto Betancor Bonilla, D. Manuel Fernández Vera, Dra. Dña. Carina Soledad González, D. Evelio José González González, D. Germán Carlos González Rodríguez, Dr. D. Alberto Hamilton Castro, D. Sergio Hernández Alonso, Dr. D. Graciliano Nicolás Marichal Plasencia, Dr. D. Roberto Marichal Plasencia, D.Carlos Martín Galán, D. Juan Julián Merino Rubio, D. Jesús Fco. Montañés Tomás, Dña.Vanesa Muñoz Cruz, D. Agustín José Padrón, Dr. D. José Demetrio Piñeiro Vera, D. Héctor Reboso Morales, D. José Julio Rodríguez Bello, Dr. D. José Luis Sánchez de la Rosa, Dr. D. José Sigut Saavedra, Dra. Dña. Marta Sigut Saavedra, D. Jonay Toledo Carrillo, D. Santiago Torres Álvarez, D. Jesús M. Torres Jorge.

También me gustaría agradecer a todos mis alumnos la alegría que me contagian cada día y todo lo que he aprendido de ellos.

Amigos de verdad no hay muchos, y yo me considero muy afortunada por contar con este tesoro. Gracias a todos, por vuestra ayuda y comprensión. En especial, aunque estén en la distancia, a mis dos tesoros favoritos, que siempre me acompañan y me dan ánimos: Marta y Esther. También quiero agradecerle a José y a su familia su compañía en este trayecto.

Y familia no hay más que una. Me siento muy afortunada por la mía, no puedo imaginar una mejor. Tengo tantas cosas que agradecerles a mis padres y a mi hermana que no cabrían en todas las hojas de esta tesis (¡y son muchas!). Todo lo que soy, los valores que tengo y todo lo bueno que he sido capaz de hacer en mi vida es gracias a ellos. En especial, me gustaría agradecer a mi madre todo su apoyo y preocupación durante el transcurso de esta tesis, y a mi hermana el haber compartido tantos momentos alegres y tristes conmigo. Gracias por la confianza que siempre habéis depositado en mí y la fuerza de voluntad y ganas de luchar que me habéis inculcado.

Y por último, y no por eso menos importante, gracias a Dios, por haber puesto tantas personas buenas en mi camino.

A mis padres,
a mi hermana.

A todas las personas que dedican
su vida a cuidar la de los demás.

Índice.

Introducción.

Este trabajo de investigación debe enmarcarse en la línea seguida por el Grupo de Computadoras y Control de la Universidad de La Laguna desde principios de los años 90, en la que se han mantenido colaboraciones multidisciplinares en diversos ámbitos de la medicina donde resulta provechoso incorporar las disciplinas del procesamiento de señales, las ciencias de la computación y la automatización.

En particular, se puede considerar esta tesis como la continuación de diversos trabajos de investigación previos centrados en la detección y clasificación de patrones en señales biomédicas [Aguilar, 1994], [Sánchez, 1993], [Piñeiro, 1996], [Sigut, 2001], [Estévez, 2001] y [Marichal, 2003].

Tiene especial relación con el trabajo sobre detección de señales biomédicas expuesto en [Estévez, 2001], ya que ahí se introdujo el modelo denominado máquina de estados borrosa como algoritmo para la monitorización de señales. Se trata de una máquina de estados generalizada, construida con sistemas de inferencia borrosos, y cuya entrada es una serie temporal que fomenta el cambio en el nivel de activación de los estados internos. A diferencia de un autómata finito determinista, todos los estados pueden estar activados al mismo tiempo, aunque con diferentes niveles de activación que cambian en función de las entradas recibidas.

El trabajo que se presenta en esta tesis y el mencionado anteriormente [Estévez, 2001] son complementarios. En el trabajo previo, se propusieron algunos métodos para

la síntesis de la máquina de estados borrosa, y tras identificar los principales problemas de diseño, se aportaron soluciones en las que intervenía en gran medida la acción directa del experto médico. Su conocimiento experto sobre las señales a analizar se explicita en un modelo que luego es usado para la generación de la máquina de estados borrosa. Esta técnica es usada también por otros autores, por ejemplo consultar [Steimann, 1996] y [Steimann, 1997].

Por el contrario, el trabajo de investigación que se presenta aquí utiliza como única base para el diseño de la máquina de estados borrosa un conjunto de señales previamente clasificadas, es decir, se plantea un procedimiento de aprendizaje puramente inductivo, en el que no se explicita el conocimiento del experto sobre las señales a analizar.

Otra de las diferencias importantes de esta tesis respecto a trabajos previos con máquinas de estado borrosas en este grupo y por otros autores, es su utilización como parte esencial de un clasificador. La idea fundamental es utilizar una medida de la reactividad de los estados del sistema a la serie temporal que constituye la entrada, como característica para la clasificación de la serie temporal. Es decir, dadas dos clases de señales biomédicas a diferenciar, el objetivo es diseñar una máquina de estados borrosa donde la medida de los cambios sufridos en la activación del estado de detección esté bien diferenciada para ambas clases de señales. Esta aproximación al problema es muy diferente a la utilizada por [Estévez, 2001] o [Steimann, 1997], ya que en esos casos el modelo de máquina de estados borrosa prácticamente pretende describir cada uno de los cambios en la señal a detectar.

Puesto que se trata de un problema de aprendizaje inductivo, se ha usado una técnica de minimización de cierta función objetivo. La función objetivo mide el grado de acierto en la clasificación de las series temporales utilizadas como conjunto de entrenamiento. Podemos ver entonces el problema como la minimización de una función objetivo compleja que depende de la configuración de la máquina de estados borrosa y del conjunto de entrenamiento utilizado. Se trata, por tanto, de un proceso de búsqueda en el espacio de las configuraciones de máquinas de estado borrosas, que en el caso ideal debe encontrar aquella que produzca un mínimo en la función objetivo respecto al conjunto de entrenamiento utilizado.

En general, podemos dividir los algoritmos de minimización en dos clases. Aquellos que se basan en el uso de información sobre el gradiente en el espacio de búsqueda y aquellos que no lo usan. Uno de los problemas potenciales de usar una búsqueda basada en el gradiente es su propensión a quedarse atrapados en mínimos locales, es decir en zonas del espacio de búsqueda donde la variación de la función objetivo es nula o casi nula. Los métodos que no se basan en el gradiente tienen la ventaja de no padecer este problema, aunque sufren otros, como por ejemplo, una mayor lentitud en la convergencia cuando se dan las circunstancias favorables para la aplicación de técnicas basadas en el gradiente. En [Estévez, 2001] se comprobó que la sintonización de los parámetros de la máquina de estados borrosa mediante un método basado en el gradiente es muy problemática en cuanto al problema de los mínimos locales.

Por este motivo, esta tesis se basa en la utilización de un método de minimización de la función objetivo no basado en el gradiente: los algoritmos genéticos.

Una propiedad interesante de los algoritmos genéticos es la flexibilidad con la que se puede definir la función a minimizar. Este hecho influyó en su elección ya que se pretendía utilizar a la máquina de estados borrosa como parte integrante de un clasificador, donde podría requerirse la inclusión de términos en la función objetivo más allá del error de clasificación. Además, se constató la existencia de una línea de investigación muy sólida en el campo de la utilización de algoritmos genéticos para el diseño automático de sistemas borrosos, como se puede comprobar en [Cordón et al., 2001].

Desde el punto de vista de la investigación, también resultó atractiva la idea de la utilización de los algoritmos genéticos en el proceso de diseño de la máquina de estados borrosa, ya que se encontraron pocas experiencias similares en la literatura científica en donde los sistemas borrosos recurrentes fuesen construidos utilizando técnicas evolutivas.

La investigación comenzó con un sistema muy próximo al algoritmo genético clásico [Goldberg, 1989], es decir, un sistema tipo Pittsburgh donde el proceso evolutivo es puramente competitivo, y al menos, en teoría, con el suficiente número de iteraciones se debe poder llegar a alcanzar el mínimo global. En ese sistema se abordó el problema de la codificación de la máquina de estados borrosa para formar parte de una población que es evolucionada generación a generación por la aplicación de

operadores genéticos. Este sistema sirvió para probar la viabilidad de la solución planteada, es decir, la configuración de la máquina de estados borrosa por medio de algoritmos genéticos, pero también supuso el encontrar los problemas computacionales derivados de la evaluación de cada máquina de estados borrosa con un conjunto de entrenamiento. En este problema, no es sólo el tamaño de la población el que determina el número de evaluaciones, y por tanto la carga computacional, sino también el tamaño del conjunto de entrenamiento.

Por este motivo, se decidió explorar otra alternativa, el enfoque Michigan, en donde la población de individuos evoluciona combinando el carácter cooperativo de un sistema de asignación de créditos junto con el carácter competitivo de un algoritmo genético. Como aspecto novedoso, se decidió investigar un enfoque diferente al normalmente utilizado cuando se aplican los sistemas Michigan en el diseño de sistemas borrosos. En lugar de hacer que las propias reglas del sistema borroso sean los individuos de la población se planteó la utilización de meta-reglas como individuos.

Una meta-regla se describe con un par condición-acción. Se aplica sobre las reglas del sistema borroso cuando existe correspondencia entre su parte de condición y el estado actual del sistema borroso. Su parte de acción determina los cambios a realizar sobre las reglas de la máquina de estados borrosa. Nos interesa encontrar las mejores meta-reglas, entendiendo como meta-regla "buena" aquella que introduce cambios en la máquina que mejoran su eficiencia de clasificación.

El motivo para usar esta estrategia fue el obtener una medida directa de la eficiencia de la máquina de estados borrosa con la que decidir la recompensa a asignar a la meta-regla responsable de la modificación. De esta manera, la máquina de estados borrosa va sufriendo un proceso de cambio iteración a iteración establecido por las meta-reglas de la población, que reciben recompensas en función de lo positivo o negativo del cambio introducido. Este es un mecanismo cooperativo que se complementa con el sistema competitivo proporcionado por un algoritmo genético que actúa también sobre los meta-reglas. Las ventajas computacionales de este sistema son grandes, comparadas con el sistema tipo Pittsburgh, pero presenta algunas desventajas, relativas sobre todo a la complejidad del algoritmo y al mayor número de parámetros necesarios para regular su comportamiento.

Los dos sistemas fueron estudiados sobre un modelo de referencia, un modelo oculto de Markov, donde se simularon series temporales con las que validar los sistemas diseñados y estudiar la influencia de algunos de sus parámetros. También se pudo establecer una medida comparativa de su eficiencia al utilizar un método de clasificación alternativo basado en la identificación del modelo subyacente.

Finalmente, se abordó un problema de aplicación real en el ámbito de la medicina. El problema sobre el que se trabajó fue el de la clasificación de núcleos celulares en imágenes médicas de citologías.

La caracterización del aspecto del núcleo a partir de series de datos es el problema investigado en este caso. El objetivo es clasificar los núcleos en sanos y patológicos extrayendo información de su textura mediante un proceso novedoso. Se plantea la aplicación del clasificador basado en la máquina de estados borrosa para determinar si estas series de datos provenientes del procesamiento de la imagen médica y que tratan de describir la textura del núcleo contienen información relacionada con la naturaleza benigna o maligna del mismo.

Esta memoria se compone de las siguientes tres partes.

La primera parte (capítulos del 1 al 5) realiza una revisión de la teoría y las técnicas relacionadas con los métodos utilizados en el trabajo de investigación. En el capítulo 1 se introduce la lógica borrosa y los sistemas de inferencia borrosos, conceptos básicos en la máquina de estados borrosa. El capítulo 2 describe en detalle la estructura de las máquinas de estados borrosas. El capítulo 3 resume algunos conceptos básicos relativos al reconocimiento de patrones y el análisis y evaluación de clasificadores, haciendo hincapié en la evaluación de los clasificadores desde el punto de vista del diagnóstico médico, ya que es un aspecto importante para el análisis de los resultados de los experimentos realizados sobre las imágenes de citologías que se presentan en la última parte de la tesis. El capítulo 4 realiza una introducción a los algoritmos genéticos, en particular se describe el algoritmo genético básico. El capítulo 5 continua con los algoritmos genéticos y su aplicación al diseño de sistemas borrosos, describiendo los sistemas clasificadores tanto en el enfoque de Pittsburgh como en el enfoque de Michigan.

La segunda parte (capítulo 6) describe los sistemas y algoritmos diseñados en este trabajo de investigación.

Finalmente, la tercera parte (capítulos 7 y 8) contienen los resultados de la aplicación de los sistemas en la clasificación de datos simulados procedentes del modelo de referencia (capítulo 7) y en la clasificación de los datos reales procedentes de imágenes médicas (capítulo 8).

Capítulo 1
La lógica borrosa y el razonamiento aproximado.

1.1 Introducción.

La lógica borrosa es una extensión de la lógica clásica. Utilizar lógica clásica para modelar un sistema en base al conocimiento que se dispone del mismo es, en muchas ocasiones, una tarea ardua. La lógica clásica no permite manejar adecuadamente información con incertidumbre y el conocimiento del comportamiento de los sistemas suele ser normalmente impreciso: determinadas magnitudes pueden tomar valores que difícilmente se pueden clasificar en un conjunto determinado, y quedan al mismo tiempo excluidas del resto de los conjuntos.

La idea principal de la lógica borrosa es modelar las imprecisiones en el conocimiento del comportamiento del sistema a través de conjuntos borrosos y de reglas definidas de una manera vaga o poco precisa [Estévez et al., 2001]. Las variables del sistema son definidas como variables lingüísticas, de tal manera que los valores que pueden tomar son también términos lingüísticos (modelados como conjuntos borrosos), y las reglas se establecen en función de dichas variables.

Una variable lingüística hace referencia a una magnitud que toma sus valores en un espacio continuo o discreto (por ejemplo, la recta real R, o bien un subconjunto de la misma). Además, en el caso de variables lingüísticas, el espacio donde la magnitud toma valores está particionado de modo que aproximadamente cada intervalo de la partición se corresponde con un adjetivo que usamos en la vida diaria para calificar la magnitud descrita. Por ejemplo, al referirnos a grupos de edades, podemos establecer tres grupos: *jóvenes*, *adultos* y *ancianos*. Si se limita la pertenencia al grupo *jóvenes* a los individuos que tienen como máximo 30 años, no tiene sentido rechazar a aquellos que tienen 30 años y un mes. La partición es sólo aproximada ya que normalmente no existen unos límites estrictos en donde pasamos de un adjetivo a otro. En este ejemplo, nadie ha definido la edad en la que pasamos de decir "persona joven" a decir "persona adulta". Los adjetivos asociados a la magnitud tratada de esta manera los llamaremos *valoraciones de la variable lingüística o términos lingüísticos.*

Trabajar con variables lingüísticas facilita la interacción del sistema con el ser humano y la incorporación de conocimiento experto al sistema, ya que normalmente un profesional experto de cierto dominio está acostumbrado a razonar en términos de este tipo de variables.

Sin embargo, el uso de variables lingüísticas introduce una gran dificultad: cómo definir adecuadamente las valoraciones lingüísticas. Esta definición está expuesta a subjetividad: si le pidiésemos a una persona de 15 años que nos indicara en qué grupo englobaría a una de 30 años, nos contestaría posiblemente que al grupo de los adultos, mientras que si la definición la diera una persona de 50 años, nos diría que el individuo de 30 años pertenece al grupo de los jóvenes. Además, esta subjetividad depende del contexto: una misma persona podría opinar que un individuo de 30 años es joven para casarse pero no tan joven como para empezar a estudiar una carrera. Puesto que la definición de una variable lingüística suele estar llena de subjetividad, y por esta vaguedad en los términos, el proceso de razonamiento puede ser notablemente complejo, la representación del conocimiento de un experto en términos de estas variables puede ser enormemente complicada.

1.2 Los conjuntos borrosos.

La teoría clásica de conjuntos es el fundamento de la lógica clásica bivalente. Esta teoría define la noción de conjunto en términos de la función característica. Es decir, sea X el espacio de objetos y x un elemento perteneciente a X. Un conjunto clásico A se define como una colección de objetos de X que pertenecen a A. En términos de la función característica, un conjunto A tiene asociada una función sobre los elementos de X a los que asigna el valor 0 o 1. Los elementos de X que tengan asociado el valor 1 se dice que pertenecen a A ($x \in A$), mientras que los elementos que tengan asociado el valor 0 se dice que no pertenecen a A.

$$\chi_A : X \rightarrow \{0,1\} \tag{1.1}$$

Si bien hay conjuntos en los que se puede delimitar claramente la pertenencia o no a los mismos, hay otros en los que esta limitación choca con el sentido común de clasificación que utilizamos normalmente. La definición clásica de conjunto, muy útil en muchos contextos, es insuficiente para el manejo de variables lingüísticas. La teoría de conjuntos borrosos permite la definición adecuada de conjuntos que modelan situaciones de imprecisión.

El concepto de "conjunto borroso" fue introducido por primera vez por Zadeh en 1965 [Zadeh, 1965]. La propuesta de Zadeh fue un paso más en la línea de trabajos previos relativos a la definición y estudio de lógicas multivalentes, como se detalla en [Trillas, 1980] y consistió en generalizar el concepto de función característica, de forma que ahora un conjunto borroso A tiene asociada una *función de pertenencia* con dominio X (normalmente X se conoce como *universo de discurso*) y rango [0,1] ($\mu_A(x) \rightarrow [0,1]$). De esta manera, el conjunto borroso A queda definido como una colección de pares ordenados:

$$A = \{(x, \mu_A(x)) \setminus x \in X\} \tag{1.2}$$

La función de pertenencia puede tomar todos los valores del intervalo [0,1]. El valor 0 representa la no-pertenencia al conjunto A y el valor 1 representa la pertenencia total a dicho conjunto. Valores intermedios implican un grado de pertenencia intermedio. Es muy importante destacar que la especificación de una función de

pertenencia implica un grado de subjetividad. Dicha subjetividad proviene de la forma abstracta en la que diferentes personas pueden representarse un mismo concepto. No es producto del azar, como en el caso de la teoría de la probabilidad.

Los conjuntos borrosos son una herramienta útil para la definición y manipulación de las variables imprecisas. Usaremos estos conjuntos para representar variables lingüísticas. De este modo, una magnitud cualquiera puede verse como una variable lingüística cuyos valores son conjuntos borrosos que están definidos en términos lingüísticos. La totalidad de estos conjuntos borrosos, que abarca todo el universo de discurso de la variable, es denominada *partición borrosa.*

Por ejemplo, si la temperatura se interpreta como una variable lingüística, el conjunto de valoraciones que puede tomar podría ser {*muy fría*, *fría*, *media*, *templada*, *tibia*, *calurosa*}. Cada uno de estos términos está caracterizado por un conjunto borroso definido en el universo de discurso ([-6ºC, 48ºC]) de la variable temperatura, tal y como muestra la figura 1.1.

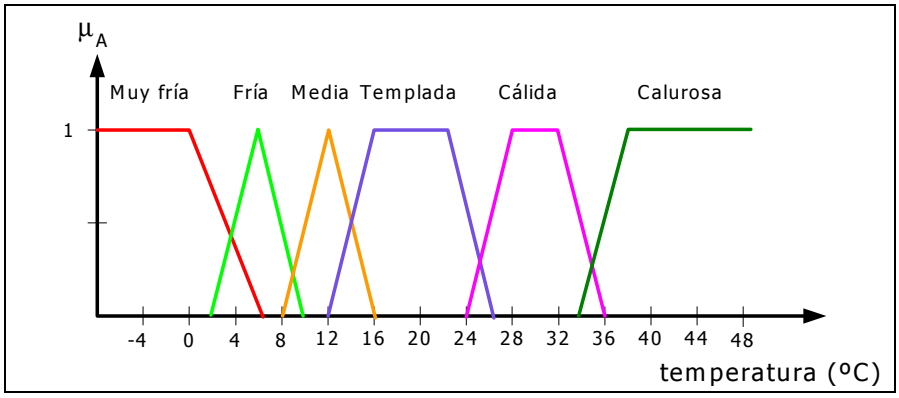


Figura 1.1. Definición de la variable lingüística Temperatura en el universo de discurso [-6,48] (ºC) y partición borrosa de la misma en seis conjuntos borrosos.

En muchas ocasiones, se suele utilizar un conjunto de términos lingüísticos normalizado que son los siguientes:

- GN Grande Negativo
- MN Medio Negativo
- PN Pequeño Negativo
- ZE Cero
- PP Pequeño Positivo

- MP Medio Positivo
- GP Grande Positivo

Para el ejemplo de la variable lingüística Temperatura, la partición borrosa con el universo del discurso normalizado entre [-1,1] se representa en la figura 1.2. El número de conjuntos borrosos que componen la partición borrosa se suele tomar según el grado de precisión requerido para esa variable. Tomar una gran cantidad de conjuntos borrosos (siete o más de siete, en general), tiene como ventaja el poder precisar las acciones que se van a llevar a cabo en el sistema en función de esta variable. La definición de sistemas borrosos en la partición se vuelve más compleja al tener que contemplar un mayor número de casos.

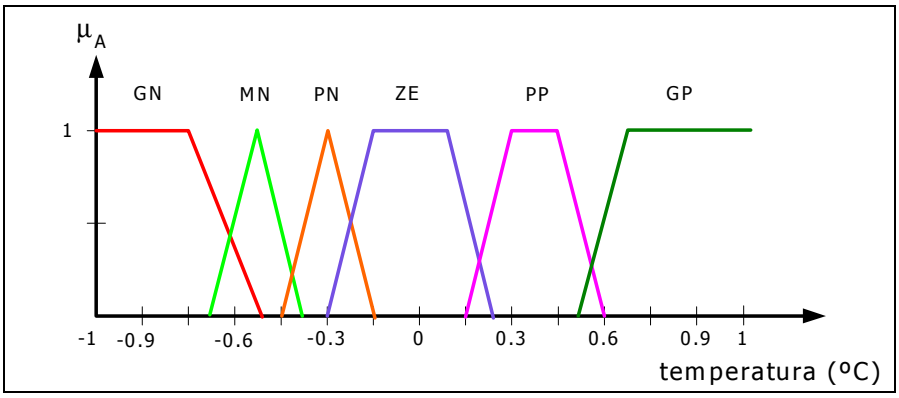


Figura 1.2. Partición borrosa para la variable lingüística Temperatura, con el universo del discurso normalizado [-1,1] en seis conjuntos borrosos.

1.2.1 Definiciones relativas a la caracterización de funciones de pertenencia.

Soporte. El soporte de un conjunto borroso A es el conjunto de elementos $x \in X$, tales que $\mu_A(x) \geq 0$.

$$sop(A) = \{x \setminus \mu_A(x) > 0\} \tag{1.3}$$

Núcleo. El núcleo de un conjunto borroso A es el conjunto de los elementos $x \in X$, tales que $\mu_A(x) = 1$.

$$nuc(A) = \{x \setminus \mu_A(x) = 1\} \tag{1.4}$$

Normalidad. Un conjunto borroso se dice normal si posee un núcleo diferente del conjunto vacío. Es decir, existe $x \in X$ tal que $\mu_A(x) = 1$.

El α-corte. El α-corte o conjunto de nivel α de un conjunto borroso A es un conjunto conciso definido de la siguiente forma:

$$A_\alpha = \{x \setminus \mu_A(x) \geq \alpha\} \tag{1.5}$$

El α-corte estricto. El α-corte estricto o conjunto de nivel α estricto se define similarmente mediante:

$$A'_\alpha = \{x \setminus \mu_A(x) > \alpha\} \tag{1.6}$$

Con estas definiciones resulta que:

$$sop(A) = A'_0$$
$$nuc(A) = A_1$$

Convexidad. Un conjunto borroso A es convexo si y sólo si $\forall x_1, x_2 \in X$ y para cualquier $\lambda \in [0,1]$,

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_A(x_1), \mu_A(x_2)\} \tag{1.7}$$

Esta definición indica que la función de pertenencia evaluada entre dos puntos cualesquiera del universo de discurso tomará valores mayores o iguales que en estos dos puntos. Esta definición prohíbe las oscilaciones dentro de una función de pertenencia.

En la figura 1.3 se pueden observar algunos de los atributos mencionados en el texto. Existen, por supuesto, más caracterizaciones de las funciones de pertenencia (consultar por ejemplo [Jang et al., 1997]), pero éstas son las necesarias para comprender el resto del trabajo.

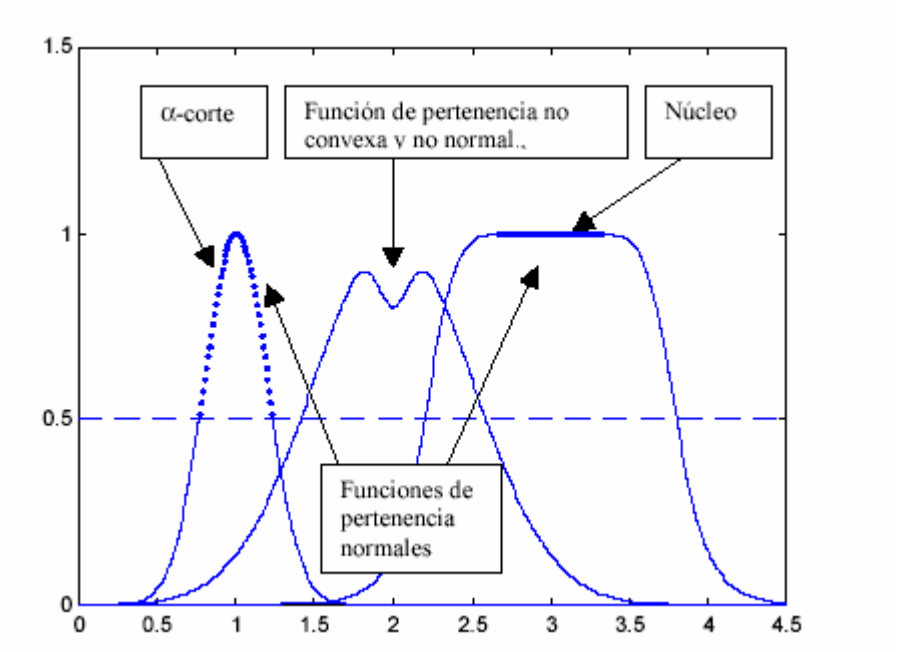


Figura 1.3. Algunas características de las funciones de pertenencia: normalidad, convexidad, núcleo y α-corte.

Número borroso. Un número borroso es un caso particular de conjunto borroso, y se define como un conjunto borroso cuya función de pertenencia es continua, convexa y definida sobre un intervalo cerrado de los números reales. La expresión matemática de cualquier número borroso definido sobre un intervalo cerrado de los números reales sería:

$$\mu_A(x) = \begin{cases} L(\frac{a-x}{\alpha}) & si \quad x \in \lfloor a-\alpha, a \rfloor \\ 1 & si \quad x \in \lfloor a, b \rfloor \\ R(\frac{x-b}{\beta}) & si \quad x \in \lfloor b, b+\beta \rfloor \\ 0 & en \quad otro \quad caso \end{cases} \tag{1.8}$$

donde $L:[0,1]\rightarrow[0,1]$ y $R:[0,1]\rightarrow[0,1]$ son funciones continuas no crecientes tales que $L(0) = R(0) = 1$ y $L(1) = R(1) = 0$. Los parámetros α y β son dos constantes. En el caso de conjuntos trapezoidales o triangulares ($L(x) = 1-x$ y $R(x) = 1-x$), α y β determinan la pendiente de los tramos de subida y bajada, respectivamente, de la función de pertenencia.

Como puede observarse, esta definición es una generalización de la función trapezoidal. Pero no contempla los casos en que el intervalo sobre el que se definan los

números borrosos sea abierto o semi-abierto, como es el caso de la gaussiana o de la campana gaussiana. Para estos casos podemos establecer la siguiente expresión general:

$$\mu_A(x) = \begin{cases} L(\frac{a-x}{\alpha}) & si \quad x \le a \\ 1 & si \quad x \in |a,b| \\ R(\frac{x-b}{\beta}) & si \quad x \ge b \end{cases} \tag{1.9}$$

donde $L:[0,\infty) \to [0,1]$ y $R:[0,\infty) \to [0,1]$ son funciones continuas tales que $L(0) = R(0) = 1$ y:

$$\lim_{x\to\infty} L(x) = \lim_{x\to\infty} R(x) = 0 \tag{1.10}$$

Las funciones de pertenencia más habituales para los conjuntos borrosos adoptan una forma triangular o trapezoidal, aunque también son habituales la gaussiana o la campana generalizada. Estas se caracterizan por un número reducido de parámetros (entre 2 y 4) y permiten representar situaciones de valores puntuales o intervalos.

1.2.2 Algunas funciones de pertenencia.

La función de pertenencia triangular. Una función de pertenencia triangular queda especificada con tres parámetros (*a,b,c*) de la siguiente manera:

$$trian(x;a,b,c) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{1.11}$$

La función de pertenencia trapezoidal. Una función de pertenencia trapezoidal queda especificada con cuatro parámetros (*a,b,c,d*) de la siguiente manera:

$$trap(x;a,b,c,d)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ 1, & b\le x\le c\\ \dfrac{d-x}{d-c}, & c\le x\le d\\ 0, & d\le x\end{cases} \tag{1.12}$$

Debido a la simplicidad de su formulación y a su eficiencia computacional, tanto la función de pertenencia triangular como trapezoidal son ampliamente utilizadas, especialmente en implementaciones en tiempo real. Sin embargo, presentan el inconveniente de que no tienen cambios suaves en los puntos definidos por sus parámetros. Las siguientes funciones de pertenencia que se presentan tienen cambios suaves y son funciones no lineales.

La función de pertenencia gaussiana. Está especificada con dos parámetros *c* (centro) y σ (desviación) de la siguiente manera:

$$gauss(x;c,\sigma)=e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{1.13}$$

La función de pertenencia campana generalizada. Está especificada por tres parámetros (*a,b,c*) como sigue:

$$camp(x;a,b,c)=\frac{1}{1+\left\|\dfrac{x-c}{a}\right\|^{2b}} \tag{1.14}$$

siendo el parámetro *b* positivo. Con una adecuada selección de los parámetros (*a,b,c*) se define la campana deseada. Concretamente, podemos ajustar los parámetros *c* y *a* para variar el ancho y centro de la función de pertenencia; y utilizar el parámetro *b* para controlar las pendientes en los puntos de inflexión.

Debido a su suavidad y definición concisa, tanto la gaussiana como la campana generalizada son muy utilizadas para la definición de números borrosos. En este trabajo hemos utilizado funciones de pertenencia gaussianas.

La función de pertenencia sigmoidal. Está especificada por dos parámetros (*a,c*) y viene definida por:

$$sig(x;a,c) = \frac{1}{1+\exp(-a(x-c))} \tag{1.15}$$

donde *a* controla la pendiente en el punto $x = c$. Dependiendo del signo de *a*, la sigmoide aparecerá orientada hacia la izquierda o hacia la derecha.

En la figura 1.4 se muestran algunas de estas funciones de pertenencia.

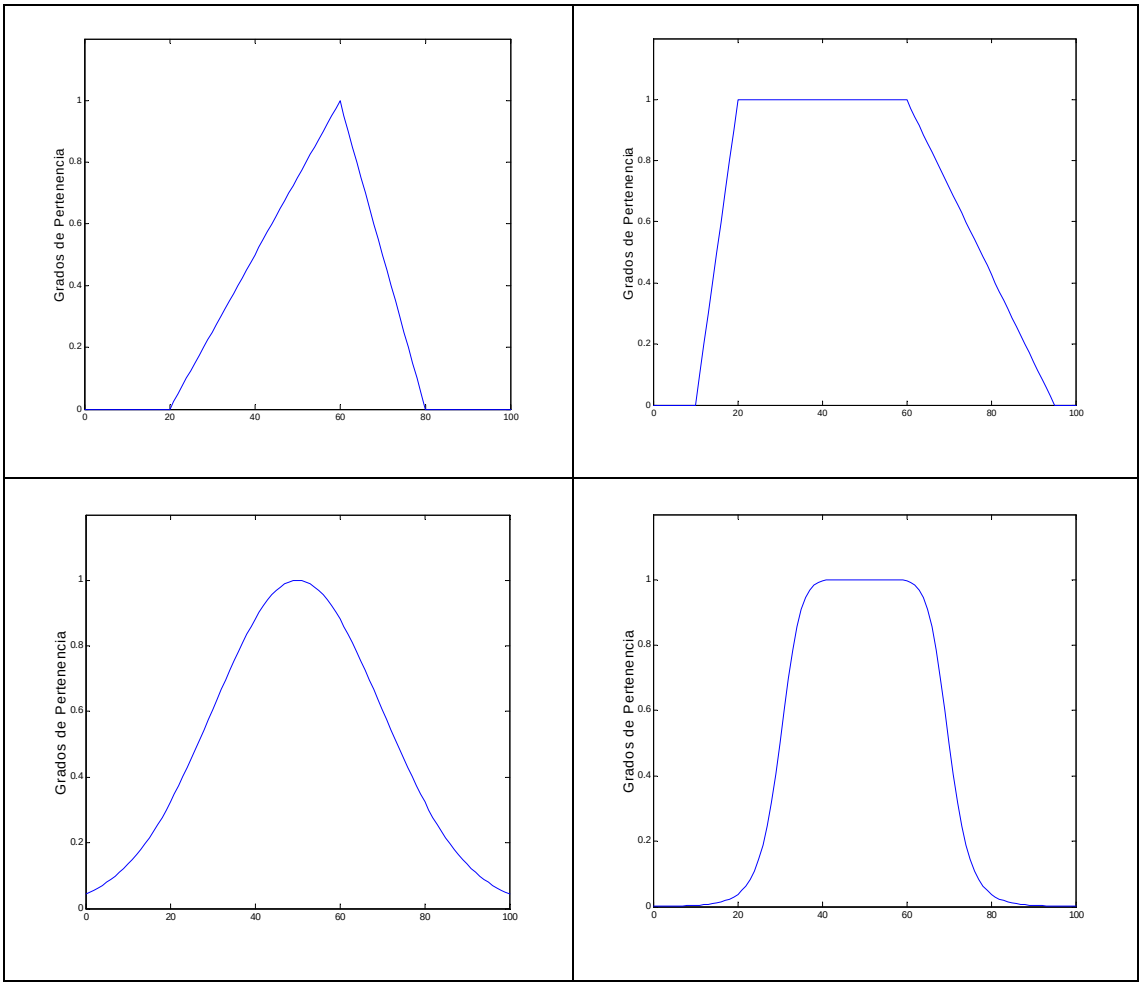


Figura 1.4. De izquierda a derecha: función de pertenencia triangular(x, 20,60,80), función de pertenencia trapezoidal (x,10,20,60,95), función de pertenencia gaussiana (x; 50,20), función de pertenencia campana generalizada (x; 20,4,50).

1.2.3 Funciones de pertenencia en dos dimensiones.

Aquí introduciremos algunos conceptos relacionados con las funciones de pertenencia en dos dimensiones que tienen relevancia en el proceso del "razonamiento

aproximado". Por simplicidad de notación, utilizaremos un modo alternativo para designar conjuntos borrosos. Un conjunto borroso puede expresarse como se indica a continuación:

$$A = \begin{cases} \sum_{x_i \in X} \mu_A(x_i)/x_i, & si \quad X \quad es \quad una \quad colección \quad de \quad objetos \quad discretos. \\ \int_X \mu_A(x)/x, & si \quad X \quad es \quad un \quad espacio \quad continuo. \end{cases} \tag{1.16}$$

Los signos de sumatorio e integración se usan para representar la unión de los pares $(x,\ \mu_A(x))$, no indican suma ni integración. De modo análogo, "/" sólo es un marcador, no implica división.

Extensión cilíndrica de un conjunto borroso en una dimensión. Sea A un conjunto borroso en X, entonces definimos su extensión cilíndrica en $X \times Y$ como un nuevo conjunto borroso $c(A)$ definido por:

$$c(A) = \int_{X \times Y} \mu_A(x)/(x,y) \tag{1.17}$$

Proyección de conjuntos borrosos. Sea R un conjunto borroso en dos dimensiones definido sobre $X \times Y$. Entonces las proyecciones de R sobre X e Y están definidas como:

$$\begin{aligned} R_X &= \int_X \left[\max_y \mu_R(x,y)\right]/x \\ R_Y &= \int_Y \left[\max_x \mu_R(x,y)\right]/y \end{aligned} \tag{1.18}$$

1.2.4 Operaciones con conjuntos borrosos.

Las operaciones definidas en la lógica clásica pueden extenderse para ser aplicadas sobre los conjuntos borrosos. De hecho, las operaciones que a continuación se definen pueden reducirse a las definiciones clásicas si el grado de pertenencia a los conjuntos borrosos se limita al conjunto {0,1}. Por esta razón, se utiliza la misma notación que para los conjuntos concisos.

Estas operaciones son los mecanismos fundamentales con los que se modelará el razonamiento aproximado. Las definiciones de estos operadores no son únicas,

existiendo muchas variantes de las mismas en la literatura. Inicialmente, cada operador se definirá mediante lo que denominaremos el conjunto de operadores borrosos clásico o estándar, que toman como base las operaciones *max*{*x,y,...*}(valor máximo de una lista de números) y *min*{*x,y,...*}(valor mínimo de una lista de números). Las definiciones basadas en los operadores *min* y *max* son generalizadas mediante el uso de dos clases de funciones: las T-normas o S-conormas para el operador *min* y las T-conormas o S-normas para el operador *max*.

Complemento o negación. El complemento de un conjunto borroso A, que denotaremos por $\overline{A}$, se define de forma estándar como un conjunto borroso cuya función de pertenencia es:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{1.19}$$

El operador complemento borroso se define de forma general como una función continua $N : [0,1] \to [0,1]$ que cumple los siguientes axiomas:

$$\begin{aligned} &N(0) = 1 \quad y \quad N(1) = 0 \quad (contorno) \\ &N(a) \geq N(b) \quad si \quad a \leq b \quad (monotonía) \end{aligned} \tag{1.20}$$

Un requerimiento adicional, aunque no necesario es la propiedad de involución:

$$N(N(a)) = a \tag{1.21}$$

Algunas generalizaciones del operador complemento borroso estándar que cumplen los dos primeros axiomas se muestran en la Tabla 1.1. De ellas, la más usada es el complemento estándar.

Estándar	*N(a)=1-a*
Sugeno	$N_S(a) = \frac{1-a}{1+sa}$, con s>-1
Yager	$N_w(a) = (1-a^w)^{1/w}$, w>0

Tabla 1.1. Definición de operadores de complemento.

Inclusión. Un conjunto borroso A está contenido en un conjunto borroso B, o también A es subconjunto de B si y sólo si $\mu_A(x) \leq \mu_B(x)$ para cualquier x del universo del discurso.

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x) \tag{1.22}$$

Unión. La unión de dos conjuntos borrosos A y B, es a su vez un conjunto borroso $C = A \cup B$, cuya función de pertenencia queda definida de forma estándar por la operación:

$$\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \vee \mu_B(x) \tag{1.23}$$

Según esta expresión, intuitivamente la unión de dos conjuntos borrosos es el "menor" de los conjuntos borrosos que los contiene a ambos.

La generalización de este operador se realiza en términos de la denominada S-norma o T-conorma (conorma triangular). La función de pertenencia del conjunto unión queda entonces definida como:

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) \tag{1.24}$$

La S-norma es una función de dos variables que satisface los siguientes axiomas:

$$\begin{aligned} &S(1,1) = 1, \quad S(0,a) = S(a,0) = a \quad (contorno) \\ &S(a,b) \leq S(c,d) \quad si \quad a \leq c \quad y \quad b \leq d \quad (monotonía) \\ &S(a,b) = S(b,a) \quad (conmutatividad) \\ &S(a,S(b,c)) = S(S(a,b),c) \quad (asociatividad) \end{aligned} \tag{1.25}$$

El primer axioma permite la generalización del operador a conjuntos concisos. El segundo establece que al decrecer la pertenencia asociada a los conjuntos que se van a unir, también decrecerá la pertenencia al conjunto unión. El tercer axioma permite que el orden de los conjuntos que se van a operar no influya en el resultado. Finalmente, el último axioma permite realizar la unión de más de dos conjuntos borrosos agrupándolos en parejas de la forma que se desee, sin que varíe el resultado.

En la tabla 1.2 se muestra una lista de S-normas (T-conormas). Algunas de ellas están parametrizadas. En la figura 1.5 podemos observar el efecto de aplicar cuatro S-

normas diferentes (máximo, suma algebraica, suma limitada o acotada y suma drástica) sobre los conjuntos borrosos X e Y para el caso en que $X = Y$ = trapezoide(3,8,12,17).

T-conorma máximo	$S(a,b) = \max(a,b) = a \vee b$
T-conorma suma algebraica (probabilística)	$S(a,b) = a + b - ab$
T-conorma suma limitada	$S(a,b) = 1 \vee (a+b)$
T-conorma suma drástica	$S_{sdra}(a,b) = \begin{cases} a, & si \quad b = 0 \\ b, & si \quad a = 0 \\ 1, & si \quad a,b > 0 \end{cases}$
T-conorma de Schweizer y Sklar	$S_{SS}(a,b,p) = 1 - (\max\{0,((1-\alpha)^{-p} + (1-b)^{-p} - 1)\})^{-\frac{1}{p}}$, p>0
T-conorma de Yager	$S_Y(a,b,q) = \min\{1,(a^q + b^q)^{\frac{1}{q}}\}$, q>0
T-conorma de Dubois y Prade	$S_{DP}(a,b,\alpha) = (a+b-ab-\min\{a,b,(1-\alpha)\}) / \max\{1-a,1-b,\alpha\}$, $\alpha \in [0,1]$

Tabla 1.2. Definición de S-normas básicas.

Intersección. La intersección de dos conjuntos borrosos A y B, es a su vez un conjunto borroso $C = A \cap B$, cuya función de pertenencia queda definida por la operación:

$$\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \wedge \mu_B(x) \tag{1.26}$$

En este caso, el significado intuitivo de la intersección se refiere al "mayor" conjunto borroso contenido por ambos conjuntos. La generalización de este operador se realiza en términos de una función T que se denomina T-norma. Entonces,

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) \tag{1.27}$$

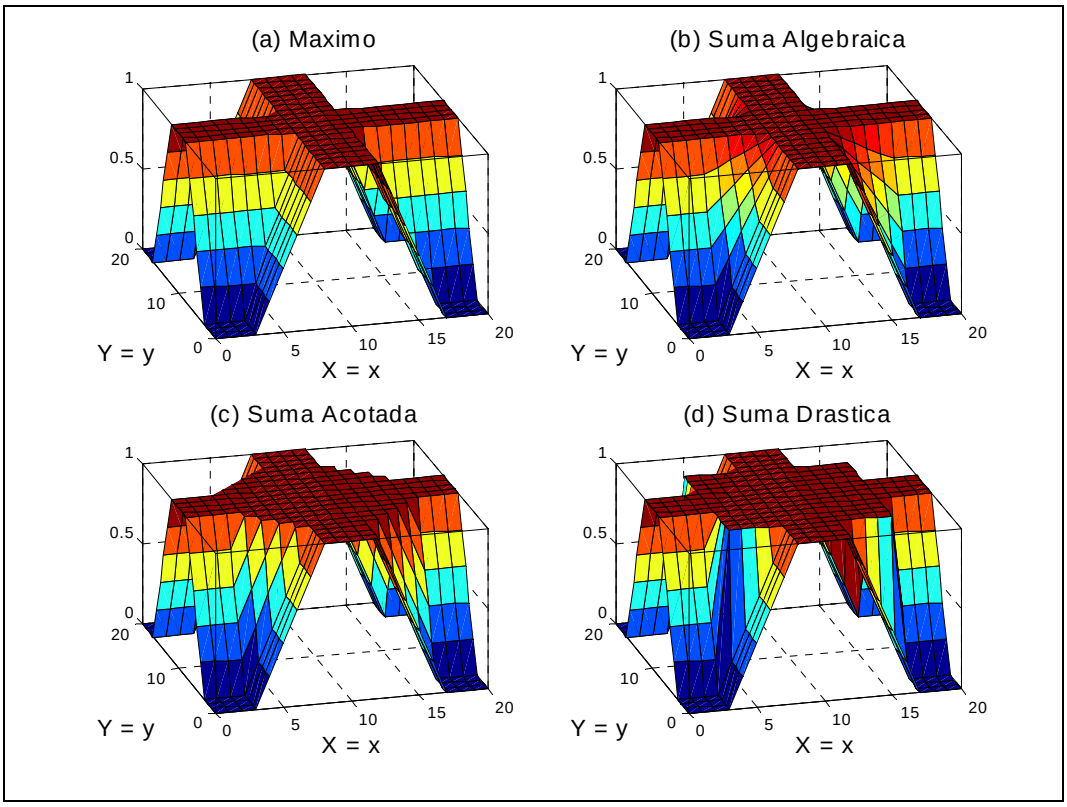


Figura 1.5. S-norma máximo, suma algebraica, suma acotada y suma drástica para $X = Y =$ trapezoide (3,8,12,17).

La T-norma se define como la función $T : [0,1]\times[0,1] \rightarrow [0,1]$ con las siguientes propiedades:

$$\begin{aligned} &T(0,0) = 0, \quad T(a,1) = T(1,a) = a \quad (contorno) \\ &T(a,b) \le T(c,d) \quad si \quad a \le c \quad y \quad b \le d \quad (monotonía) \\ &T(a,b) = T(b,a) \quad (conmutatividad) \\ &T(a,T(b,c)) = T(T(a,b),c) \quad (asociatividad) \end{aligned} \tag{1.28}$$

Algunos ejemplos de T-normas muy utilizadas aparecen en la tabla 1.3. En la figura 1.6 podemos observar el efecto de aplicar cuatro T-normas diferentes (mínima, producto algebraico, producto acotado y producto drástico) sobre los conjuntos borrosos X e Y para el caso en que $X = Y =$ trapezoide(3,8,12,17).

T-norma mínimo	$T_{\min}(a,b) = \min(a,b) = a \wedge b$
Producto algebraico	$T_{prod}(a,b) = ab$
Producto acotado	$T_{pac}(a,b) = 0 \vee (a+b-1)$
Producto drástico	$T_{pdra}(a,b) = \begin{cases} a, & si \quad b=1 \\ b, & si \quad a=1 \\ 0, & si \quad a,b<1 \end{cases}$
T-norma de Schweizer y Sklar	$T_{SS}(a,b,p) = (\max\{0,(\alpha^{-p}+b^{-p}-1)\})^{-\frac{1}{p}}$, p>0
T-norma de Yager	$T_Y(a,b,q) = 1-\min\{1,((1-a)^q+(1-b)^q)^{\frac{1}{q}}\}$, q>0
T-norma de Dubois y Prade	$T_{DP}(a,b,\alpha) = (ab)/\max\{a,b,\alpha\}$, $\alpha \in [0,1]$

Tabla 1.3. Definición de T-normas básicas.

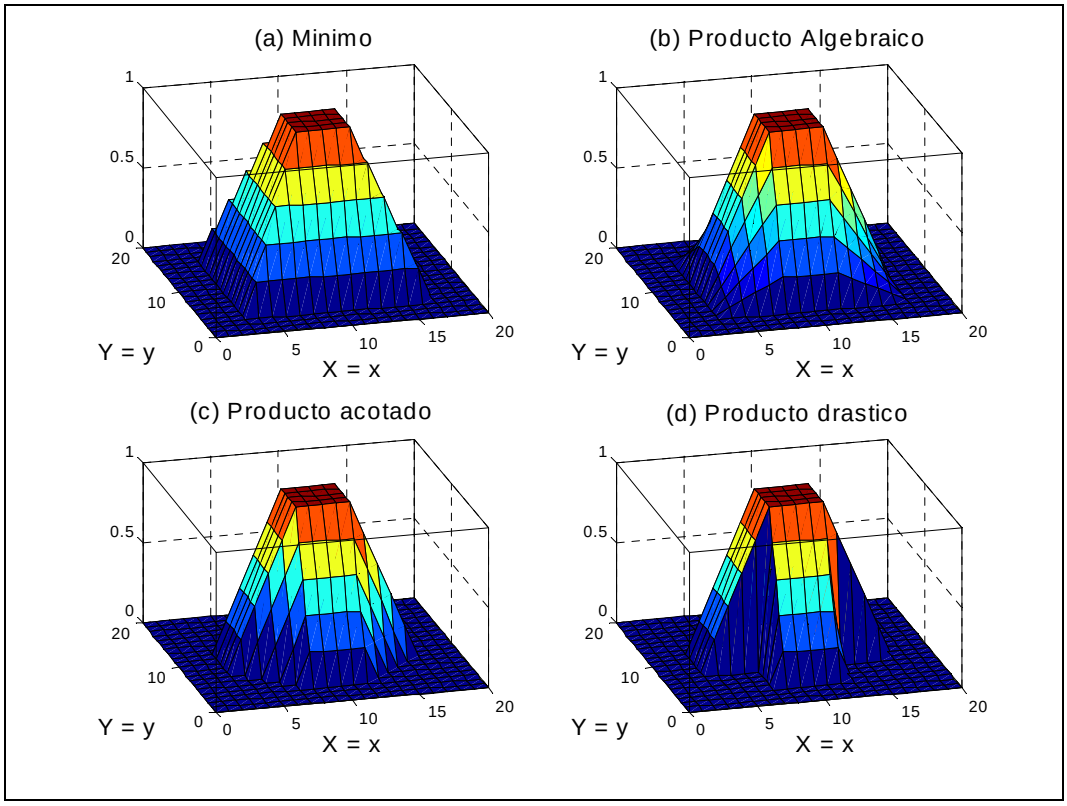


Figura 1.6. T-norma mínima, producto algebraico, producto acotado y producto drástico para $X = Y =$ trapezoide(3,8,12,17).

1.2.5 Relación entre las operaciones.

Por las relaciones que se cumplen para las uniones y las intersecciones se ve que la T-norma mínimo es el límite superior para las intersecciones mientras que la S-norma máximo es el límite inferior para las uniones.

Asimismo, la definición de la T-norma producto drástico, T_{pdra} y la S-norma suma drástica, S_{sdra}, viene caracterizada por una importante propiedad de acotación. Dadas una S-norma cualquiera *s* y una T-norma cualquiera *t*, se tiene que:

$$T_{pdra} \le t(a,b) \le \min(a,b) \le \max(a,b) \le s(a,b) \le S_{sdra}(a,b) \qquad (1.29)$$

Como consecuencia, de todos los pares de uniones e intersecciones el par max-min es el que da valores más cercanos mientras que el par T_{pdra} - S_{sdra} es el que da los valores más lejanos.

Por otro lado, los pares de uniones e intersecciones presentados cumplen las leyes de Morgan bajo el complemento estándar. Esto es, si *T* es una T-norma, la igualdad:

$$S(a,b) = 1\text{-}T(1\text{-}a,1\text{-}b) \qquad (1.30)$$

Define una S-norma y se dice que *S* está asociada con *T*. Análogamente, dada una S-norma *S*, la T-norma asociada *T* cumple la igualdad:

$$T(a,b) = 1\text{-}S(1\text{-}a,1\text{-}b) \qquad (1.31)$$

Estas definiciones no son más que una representación de las leyes de Morgan en el caso de la lógica borrosa donde la operación que implementa el complemento es el complemento estándar.

1.3 Reglas borrosas si-entonces.

Una regla borrosa del tipo *si-entonces* tiene la forma:

$$\text{Si } x \text{ es } A \text{ entonces } y \text{ es } B \qquad (1.32)$$

siendo A y B valoraciones particulares de sendas variables lingüísticas. El término "x es A" se denomina antecedente, mientras que el término "y es B" se denomina consecuente. Un ejemplo sencillo de una de tales reglas es:

"Si la presión es alta, entonces el volumen es pequeño"

Para poder utilizar una regla borrosa es necesario formalizar el significado de la expresión "si x es A entonces y es B", que abreviadamente se indica como $A \rightarrow B$. Para ello es útil definir la *relación borrosa binaria.* Una *relación borrosa binaria* es un conjunto borroso R sobre $X \times Y$ (X e Y son dos universos de discurso), que asigna a cada elemento de $X \times Y$ un número entre 0 y 1:

$$R = \{((x,y),\mu_R(x,y)) \,||\, (x,y) \in X \times Y\} \tag{1.33}$$

En una regla borrosa pueden intervenir dos conjuntos borrosos o una parte antecedente borrosa y una parte consecuente borrosa. Es natural, por lo tanto, interpretar una regla en términos de una relación borrosa. La forma específica en las que se traducen las reglas borrosas a relaciones borrosas depende en primera instancia de cómo se entienda la regla lógica.

Existen dos maneras de entender el significado de la regla lógica "si x es A entonces y es B": primero, considerando que A y B están acoplados (*coupled*), o segundo, interpretando la regla como A supone B (*entails*). Basándonos en estas dos interpretaciones y en diferentes T-normas o S-normas, existen diferentes métodos para calcular la relación borrosa $R = A \rightarrow B$. El conjunto borroso R puede ser considerado un conjunto en dos dimensiones tal que:

$$\mu_R(x,y) = f(\mu_A(x), \mu_B(y)) = f(a,b) \tag{1.34}$$

donde $a = \mu_A(x)$, $b = \mu_B(x)$ y la función f, llamada *función de implicación borrosa*, realiza la tarea de transformar los grados de pertenencia de x en A y de y en B en el grado de pertenencia de (x,y) en $A \rightarrow B$.

1.3.1 Interpretación de A → B como A y B están acoplados.

Si entendemos que $A \rightarrow B$ como que A está acoplado a B la relación borrosa que está asociada a la regla lógica es:

$$R = A \rightarrow B = T(\mu_A(x), \mu_B(x)) \tag{1.35}$$

siendo T una T-norma. Estamos expresando, por tanto, una relación del tipo *A* Y-lógico *B*. Por lo tanto, la relación borrosa asociada se puede obtener sin más usando alguna de las T-normas que se citaron anteriormente. En la tabla 1.4 y en la figura 1.7 se presentan varios ejemplos.

Mínimo (Mamdani)	$R_m = \min\{(\mu_A(x), \mu_B(x))\}/(x,y)$
Producto (Larsen)	$R_p = \mu_A(x)\mu_B(x)/(x,y)$
Producto acotado (Lukasiewicz)	$R_{pac} = \max\{0,(\mu_A(x)+\mu_B(x)-1)\}/(x,y)$
Producto drástico	$R_{pd} = \begin{cases} \min\{\mu_A(x), \mu_B(y)\} & si \quad \max\{\mu_A(x), \mu_B(y)\} = 1 \\ 0 & en \quad otro \quad caso \end{cases} /(x,y)$

Tabla 1.4. Distintas funciones de implicación borrosa para el caso en que *A* y *B* estén acoplados.

En los sistemas borrosos que hemos empleado en este trabajo, interpretamos las reglas borrosas de este modo, y la T-norma que utilizamos es la T-norma mínimo (implicador de Mamdani).

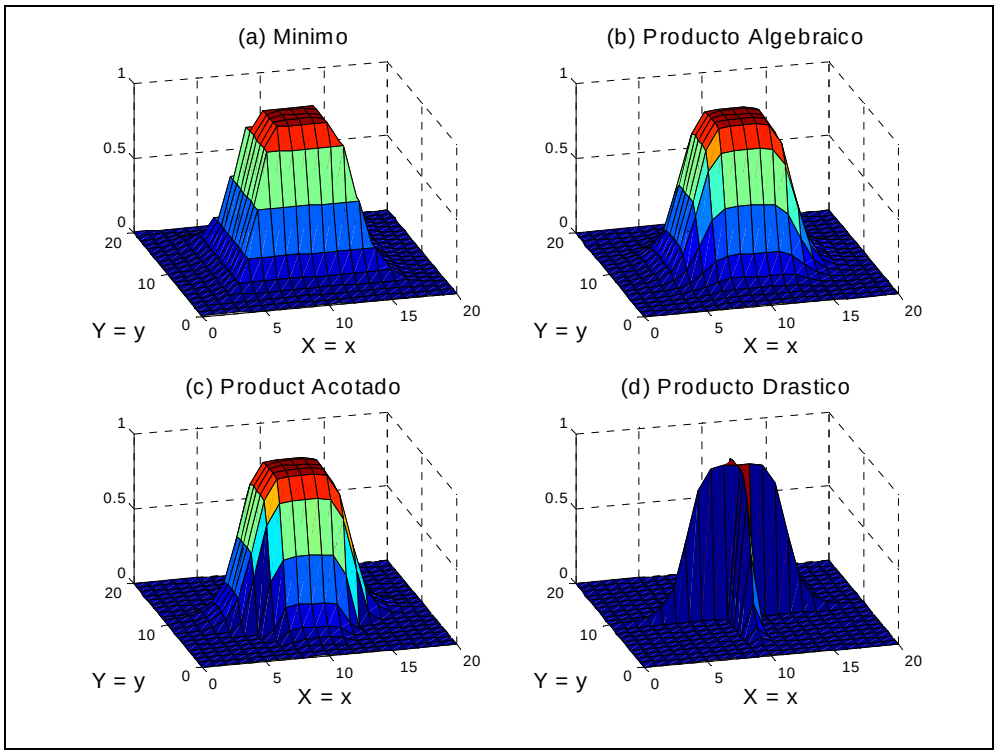


Figura 1.7. Implicador de Mamdani (mínimo), de Larsen (producto), de Lukasiewicz (producto acotado) e implicador borroso utilizando el producto drástico, considerando $\mu_A(x)$ = campana (4,3,10) y $\mu_B(y)$ = campana (4,3,10).

1.3.2 Interpretación de A → B como A supone B.

Por otra parte, si leemos la regla como *A* supone *B*, se pueden encontrar diferentes interpretaciones lógicas como la implicación material $(\neg A \cup B)$, la implicación del cálculo proposicional $(\neg A \cup (A \cap B))$, o la implicación del cálculo proposicional extendido $((\neg A \cap \neg B) \cup B)$. Para una explicación más detallada ver [Jang et al., 1997]. Tomando esta interpretación para la relación borrosa, existen distintas funciones de implicación borrosa, algunos ejemplos se muestran en la tabla 1.5 y en la figura 1.8.

Suma acotada para el operador unión (regla aritmética de Zadeh)	$R_a = \neg\ A \cup B = \min(1, (1-\mu_A(x)+\mu_B(y)))/(x,y)$
Mínimo para la intersección y Máximo para la unión (regla max-min de Zadeh)	$R_{mm} = \neg\ A \cup (A \cap B) = \max\{(1-\mu_A(x)), \min(\mu_A(x), \mu_B(y))\}/(x,y)$
Máximo para la unión (implicación borrosa booleana)	$R_a = \neg\ A \cup B = \max\{(1-\mu_A(x)), \mu_B(y)\}/(x,y)$
Producto algebraico como T-norma (Goguen)	$R_g = \begin{cases} 1 & si \quad \mu_A(x) \le \mu_B(y) \\ \dfrac{\mu_B(y)}{\mu_A(x)} & en \quad otro \quad caso \end{cases} /(x,y)$

Tabla 1.5. Distintas funciones de implicación borrosa para la interpretación de la regla "*A* supone *B*".

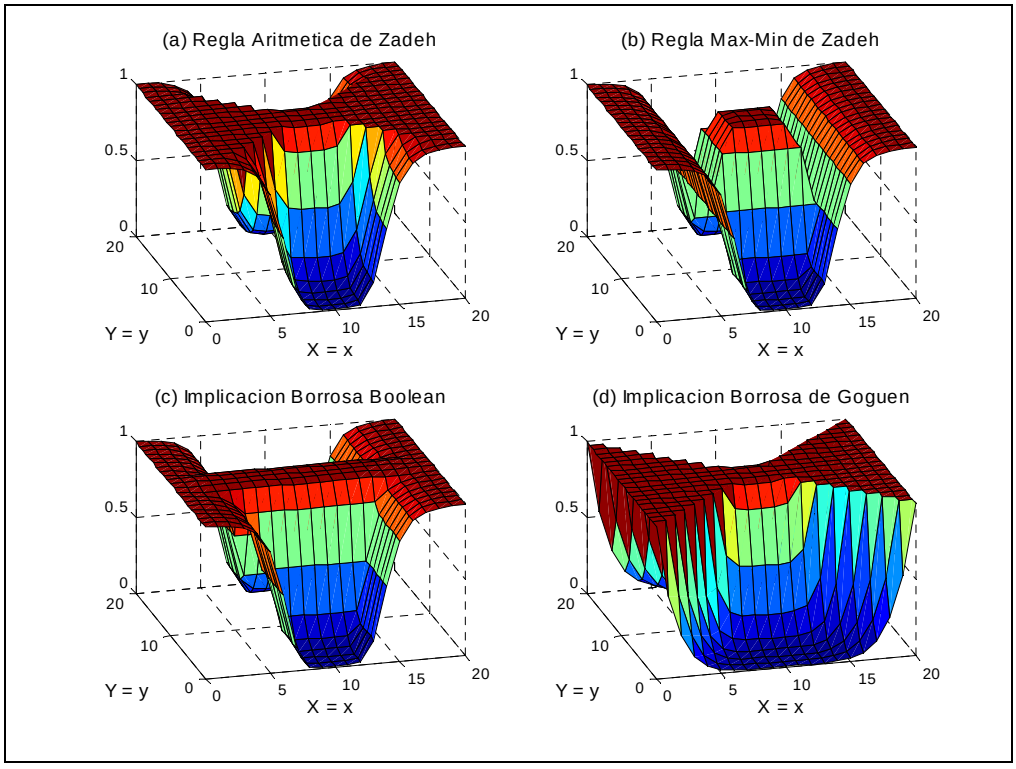


Figura 1.8. Implicación aritmética de Zadeh, Regla Max-Min de Zadeh Implicación Borroso Booleano e Implicación Borrosa de Goguen considerando $\mu_A(x)$ = campana (4,3,10) y $\mu_B(y)$ = campana (4,3,10).

1.4 El razonamiento aproximado.

El método de razonamiento aproximado permite obtener conclusiones a partir de un conjunto de reglas borrosas y un conjunto de hechos borrosos. La base de este procedimiento es la conocida *regla composicional de inferencia*.

La regla de inferencia clásica es el Modus Ponens que nos permite inferir la verdad de la proposición B a partir de la proposición A y de la implicación $A \rightarrow B$. De forma esquematizada tendríamos:

Premisa 1 (regla)	Si x es A	entonces	y es B
Premisa 2 (hecho)	x es A		
Consecuente (conclusión):			y es B (1.36)

Sin embargo, en muchos razonamientos humanos, el modus ponens es utilizado de forma aproximada. Por ejemplo, si tenemos la misma regla de implicación y x es A' entonces podemos inferir que y es B', que se podría esquematizar de la siguiente manera:

Premisa 1 (regla)	Si x es A	entonces	y es B
Premisa 2 (hecho)	x es A'		
Consecuente (conclusión):			y es B' (1.37)

donde A' es "más o menos" A y B' es "más o menos" B. Este tipo de razonamiento es el razonamiento aproximado o borroso, siendo A, B, A' y B' conjuntos borrosos del universo del discurso correspondiente. A esta regla de inferencia se le conoce como Modus Ponens Generalizado (GMP) ya que la regla de Modus Ponens es un caso especial de ésta.

Por lo tanto, el razonamiento borroso nos permite obtener conclusiones a partir de reglas borrosas de tipo si-entonces y de hechos conocidos. La base de este procedimiento es la regla composicional de inferencia.

La regla composicional de inferencia es una generalización del concepto de curva. Una curva viene dada por una función que permite obtener el valor de y a partir

del valor de *x* mediante *y = f(x)*. El papel de la variable independiente lo jugará el hecho, el papel de la función lo tomará la relación borrosa asociada a la regla y finalmente, el papel de la variable dependiente será para la conclusión borrosa, como se muestra en la figura 1.9.

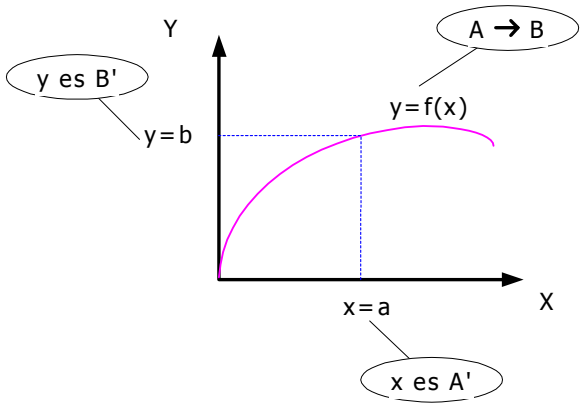


Figura 1.9. Regla compocisional de inferencia como generalización del concepto de curva.

Sea *F* una relación borrosa sobre *X×Y* y sea *A* un conjunto borroso sobre *X*. Para encontrar el conjunto borroso resultante *B*, construiremos una extensión cilíndrica de *A*, *c(A)*, que tome como base *A*. La extensión cilíndrica es un conjunto borroso sobre *X×Y*, por lo que podemos calcular la intersección con el conjunto borroso que define la relación borrosa *F*. Finalmente proyectaremos $c(A) \cap F$ sobre el dominio *Y*, obteniendo así un conjunto borroso *B* que representa la conclusión. Este procedimiento está representado gráficamente en la figura 1.10.

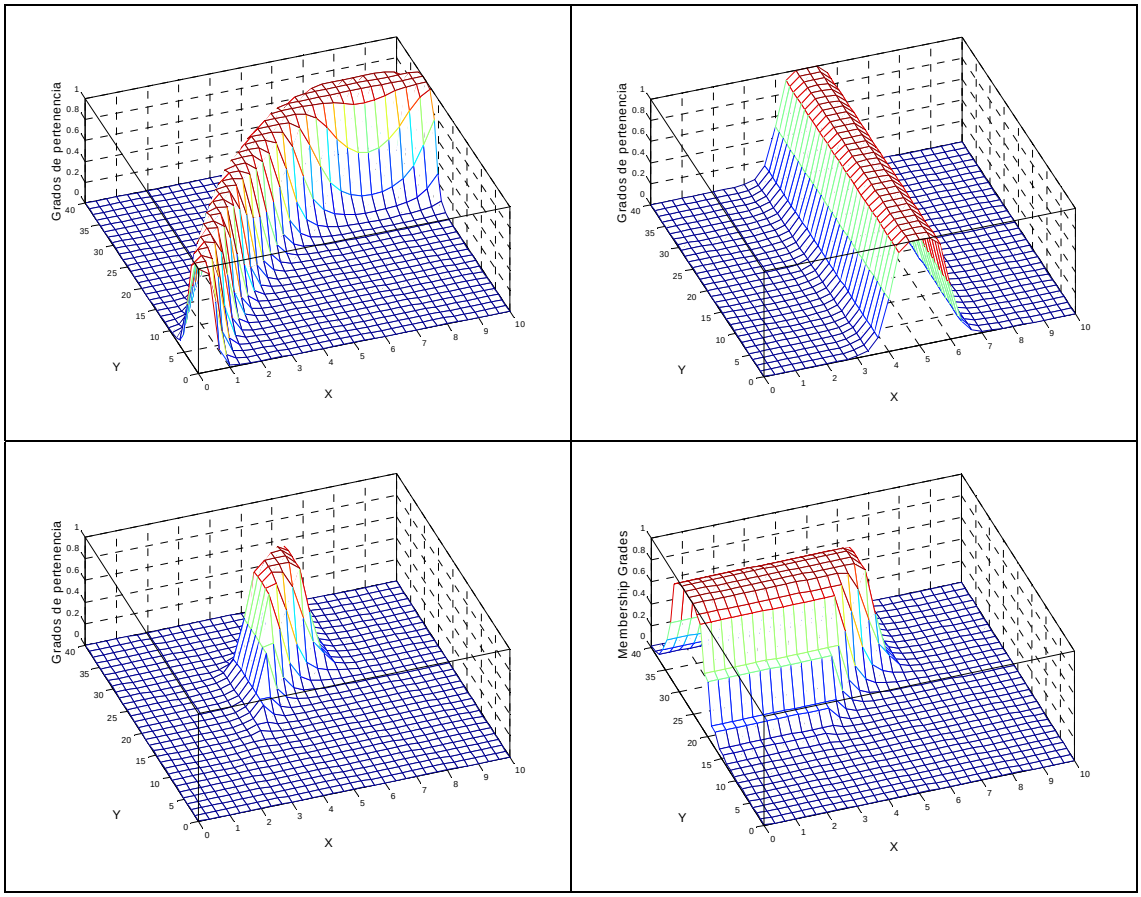


Figura 1.10. De izquierda a derecha y de arriba a abajo: relación borrosa *R* sobre *X* e *Y*, extensión cilíndrica de *A'*, *c(A')*, intersección (mínimo) de la relación borrosa *R* y la extensión cilíndrica de *A'* y proyección (máximo) sobre el eje *Y* de esta intersección para obtener la conclusión.

Expresándolo matemáticamente y recordando las anteriores definiciones sobre extensión cilíndrica de un conjunto borroso y proyección tenemos:

$$\mu_C(x,y) = \mu_A(x) \tag{1.38}$$

$$\mu_C(A) \cap F(x,y) = min[\mu_C(A)(x,y), \mu_F(x,y)] = min[\mu_A(x), \mu_F(x,y)] \tag{1.39}$$

Finalmente se realiza la proyección:

$$\mu_B(y) = max_x min[\mu_A(x), \mu_F(x,y)] = V_x[\mu_A(x) \wedge \mu_F(x,y)] \tag{1.40}$$

Como se observa, se llega a una regla composicional del tipo max-min. El tipo de razonamiento que nos permite esta regla composicional es una extensión del *modus ponens*. Es decir, si se tiene la regla "Si *x* es *A* entonces *y* es *B*" y se tiene el hecho "*x* es

A" entonces se deduce la verdad de la proposición "*y* es *B*". La extensión de esta forma de proceder permite cubrir situaciones como la representada por la regla "Si *x* es *A* entonces *y* es *B*" y el hecho "*x* es aproximadamente *A*", pudiéndose obtener "*y* es *B'* ", siendo *B'* un concepto "cercano" a *B*.

Definimos el razonamiento aproximado más formalmente. Sean *A, A'* y *B* conjuntos borrosos sobre *X, X* e *Y* respectivamente. Supongamos que la implicación $A \rightarrow B$ es expresada como una relación borrosa *R* sobre $X \times Y$. Entonces el conjunto *B'* que se infiere de la regla "Si *x* es *A* entonces *y* es *B*", cuando se da el hecho "*x* es *A'* ", tiene como función de pertenencia:

$$\mu_{B'}(y) = max_x min[\mu_{A'}(x), \mu_R(x,y)] = V_x[\mu_A(x) \wedge \mu_R(x,y)] \qquad (1.41)$$

La notación para la regla composicional de inferencia es:

$$B' = A' \circ R = A' \circ (A \rightarrow B) \qquad (1.42)$$

Esta definición contempla el caso de una regla con un solo antecedente. Veamos ahora, algunas generalizaciones de este procedimiento.

Empecemos con una sola regla y múltiples antecedentes. Sea la regla "si *x* es *A* e *y* es *B* entonces *z* es *C*", junto con el hecho "*x* es *A'* e *y* es *B'* ". Se trata de obtener la conclusión *z* es *C'*. La Generalización del Modus Ponens para este problema se esquematizaría de la siguiente manera:

Premisa 1 (regla)	Si *x* es *A* e *y* es *B*	entonces	*z* es *C*
Premisa 2 (hecho)	*x* es *A'* e *y* es *B'*		
Consecuente (conclusión):			*z* es *C'* (1.43)

El método para obtener *C'* se basa en utilizar una relación ternaria borrosa para describir esta regla con dos antecedentes. Los pormenores de la demostración pueden ser consultados en [Jang et al., 1997]. El resultado es el siguiente:

$$C' = [A' \circ (A \rightarrow C)] \cap [B' \circ (B \rightarrow C)] \qquad (1.44)$$

O también,

$$\mu_{C'}(z) = \{ V_x[\mu_{A'}(x) \wedge \mu_A(x)]\} \ \wedge \{V_y[\mu_{B'}(y) \wedge \mu_B(y)]\} \wedge \mu_C(z) \qquad (1.45)$$

Las expresiones calculadas mediante la composición max-min entre el conjunto borroso del hecho y el antecedente son denominadas *grados de compatibilidad*. Si $\omega_{A,A'}$ y $\omega_{B,B'}$ son los grados de compatibilidad para los antecedentes A y B respectivamente, la fórmula anterior puede escribirse:

$$\mu_{C'}(z) = \omega_{A,A'} \wedge \omega_{B,B'} \wedge \mu_C(z) \qquad (1.46)$$

El resultado de los grados de compatibilidad calculados sobre los antecedentes es denominado *fuerza de disparo de la regla*. Es decir, la fuerza de disparo $f((A',B');(A,B))$ viene dada por:

$$f((A',B');(A,B)) = \omega_{A,A'} \wedge \omega_{B,B'} \qquad (1.47)$$

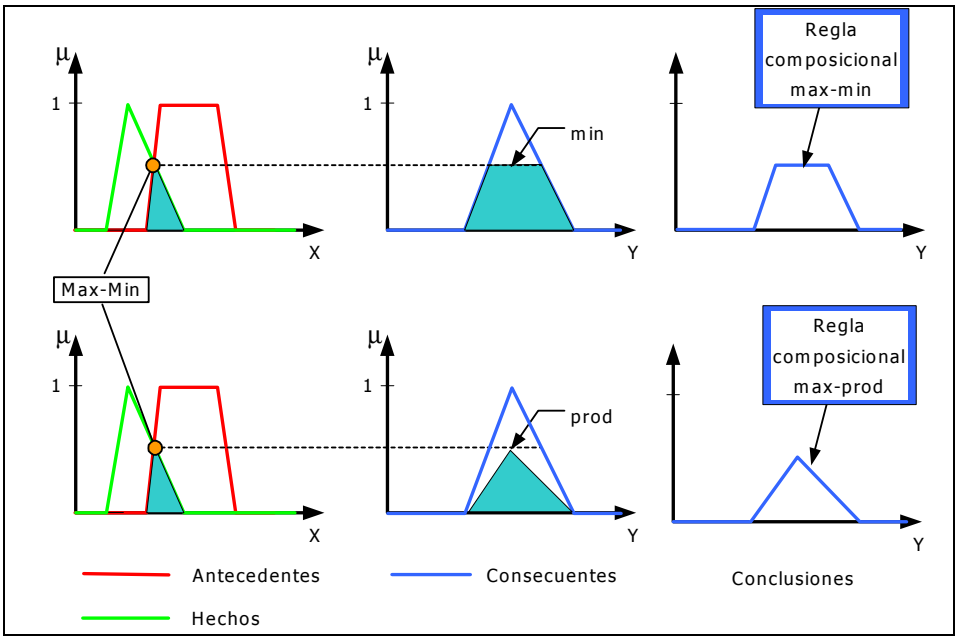


Figura 1.11. Razonamiento borroso utilizando regla composicional max-min en las tres gráficas superiores y utilizando max-prod en las tres gráficas inferiores.

En la figura 1.11 se representa gráficamente el mecanismo de razonamiento aproximado para una regla con un antecedente. Las tres gráficas superiores se refieren a la regla composicional del tipo max-min. La primera gráfica (de izquierda a derecha) muestra como se componen el antecedente y el hecho. En este caso el hecho tiene

asociada una función de pertenencia triangular, y el antecedente una función de pertenencia trapezoidal. La composición de ambas funciones de pertenencia mediante la operación max-min es lo que denominamos grado de compatibilidad. En este caso, como sólo hay un antecedente, el grado de compatibilidad es igual a la fuerza de disparo de la regla. La siguiente gráfica se refiere al consecuente de la regla. La fuerza de disparo y la función de pertenencia asociada al consecuente son utilizadas para obtener la función de pertenencia del consecuente cualificado (resultado de la inferencia de la regla). En las tres gráficas superiores se utiliza el operador mínimo mientras que en las tres gráficas inferiores se utiliza el operador producto. Dependiendo del operador utilizado se obtiene un conjunto borroso resultado de la inferencia diferente.

En la figura 1.12 se muestra la interpretación gráfica del Modus Ponens Generalizado en una regla con múltiples antecedentes, utilizando el implicador borroso de Mamdani (mínimo) y el operador composicional max-min. El resultado *C'* es igual a la función de pertenencia del conjunto borroso *C* recortada por la fuerza de disparo de la regla.

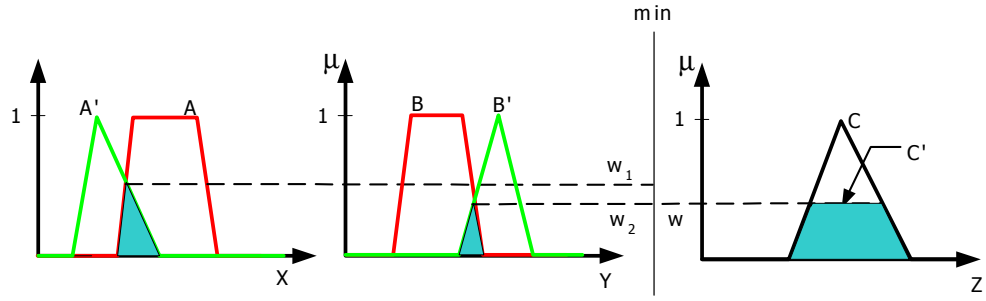


Figura 1.12. Interpretación gráfica del Modus Ponens Generalizado en una regla con múltiples antecedentes, utilizando el implicador borroso mínimo y el operador composicional max-min.

El siguiente grado de complejidad lo obtenemos cuando en lugar de sólo una regla se tienen varias. En ese caso se interpreta la relación borrosa sobre la que debemos componer los hechos como la unión de las relaciones borrosas de cada una de las reglas individuales. De este modo, dado un problema con las reglas "si x es A_1 e y es B_1 entonces z es C_1" y si "si x es A_2 e y es B_2 entonces z es C_2" y el hecho "x es A' e y es B'", la conclusión "z es C'" con:

$$C' = ([A' \circ (A_1 \rightarrow C_1)] \cap [B' \circ (B_1 \rightarrow C_1)]) \cup ([A' \circ (A_2 \rightarrow C_2)] \cap [B' \circ (B_2 \rightarrow C_2)]) \qquad (1.48)$$

y

$$\mu_{C'}(z) = (\{ V_x[\mu_{A'}(x) \wedge \mu_{A1}(x)]\} \wedge \{V_y[\mu_{B'}(y) \wedge \mu_{B1}(y)]\} \wedge \mu_{C1}(z)) \vee (\{ V_x[\mu_{A'}(x) \wedge \mu_{A2}(x)]\} \wedge \{V_y[\mu_{B'}(y) \wedge \mu_{B2}(y)]\} \wedge \mu_{C2}(z)) \quad (1.49)$$

o en términos de grados de compatibilidad:

$$\mu_{C'}(z) = (\omega_{A1,A'} \wedge \omega_{B1,B'} \wedge \mu_{C1}(z)) \vee (\omega_{A2,A'} \wedge \omega_{B2,B'} \wedge \mu_{C2}(z)) \quad (1.50)$$

y en términos de las fuerzas de disparo:

$$\mu_{C'}(z) = (f((A',B');(A_1,B_1)) \wedge \mu_{C1}(z)) \vee (f((A',B');(A_2,B_2)) \wedge \mu_{C2}(z)) \quad (1.51)$$

Denominaremos *consecuente cualificado* al conjunto borroso que se obtiene cuando se opera la fuerza de disparo y el consecuente de una regla. Llamaremos *salida conjunta* a la agregación de todos los consecuentes cualificados del sistema de reglas.

Aunque en este apartado se han desarrollado los principios del razonamiento aproximado utilizando la T-norma mínimo y la T-conorma máximo, las definiciones siguen siendo válidas usando cualquier otra T-norma y T-conorma, siempre que se respete la ubicación respectiva de las T-normas y T-conormas. Además, el tipo de regla lógica utilizada no tiene por qué basarse exclusivamente en el operador Y-lógico. También es admisible la aparición del operador O-lógico. En ese caso se sustituye la intersección de los resultados de la regla composicional de inferencia por la unión, usándose entonces una T-conorma.

La figura 1.13 muestra gráficamente el razonamiento borroso para múltiples reglas con múltiples antecedentes, utilizando el implicador borroso de Mamdani y la regla de inferencia composicional max-min. Este mecanismo es el utilizado en este trabajo. Como se puede apreciar, en cada una de las reglas se realiza la inferencia empezando por la obtención de los grados de compatibilidad (operación max-min) representado por $\omega_{A1,A'}$ y $\omega_{B1,B'}$ en la primera y $\omega_{A2,A'}$ y $\omega_{B2,B'}$ en la segunda. A continuación se aplica el operador Y-lógico mediante una T-norma, en este caso la T-norma mínimo. El resultado es la fuerza de disparo de la regla (ω_1 y ω_2 para las reglas uno y dos respectivamente) que es utilizada junto a la función de pertenencia del consecuente y un método de inferencia para obtener el consecuente cualificado (C_1' y C_2' para las reglas uno y dos respectivamente). La regla composicional de inferencia

utilizada es la T-norma mínimo. Los consecuentes cualificados obtenidos se observan a la derecha de cada una de las filas que representan las reglas. La agregación de los consecuentes cualificados se realiza en este caso mediante la aplicación de la S-norma máximo, cuyo resultado se observa en la parte inferior de la figura. El borde superior del área sombreada es la función de pertenencia asociada al conjunto borroso resultado de la inferencia con múltiples reglas y múltiples antecedentes.

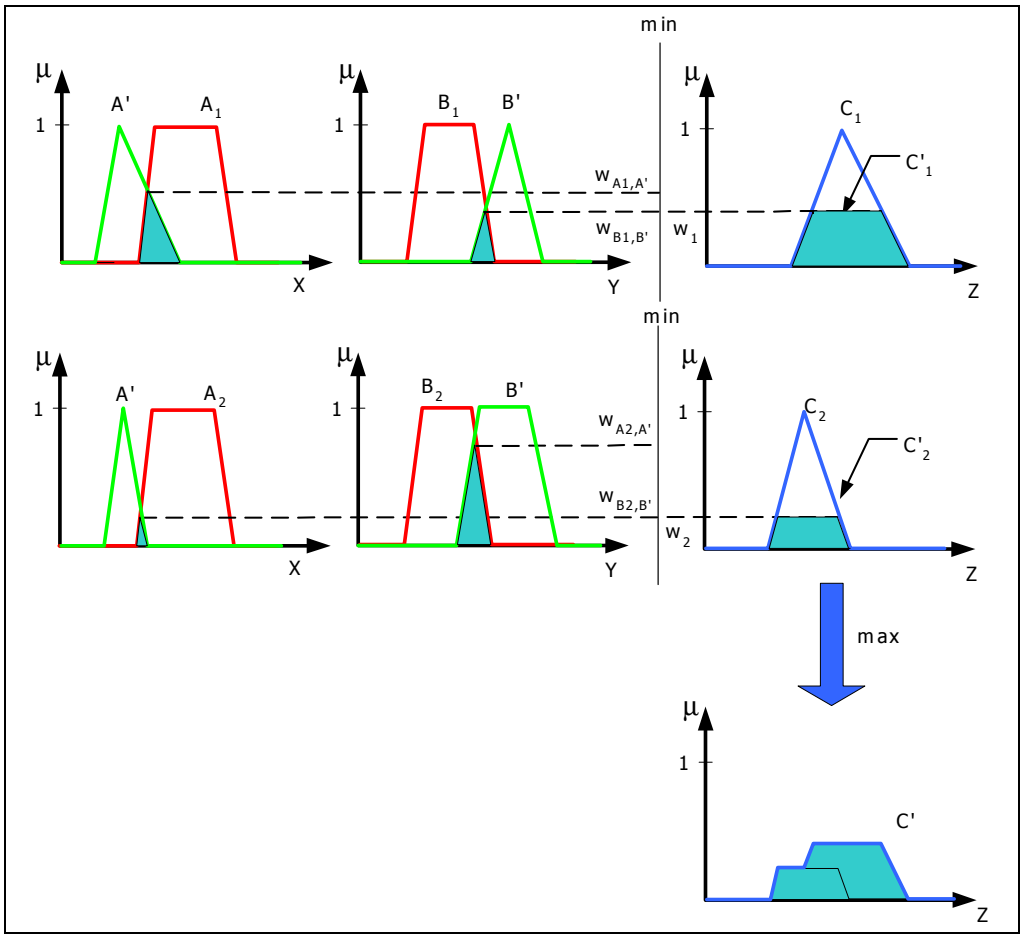


Figura 1.13. Interpretación gráfica del Modus Ponens Generalizado para varias reglas con múltiples antecedentes, usando el implicador borroso de Mamdani (mínimo) y el operador composicional max-min.

1.5 Los sistemas de inferencia borrosos.

Los sistemas de inferencia borrosos (SIB) constituyen una clase de algoritmos de cómputo basados en los conjuntos borrosos y el razonamiento aproximado. Tienen otras denominaciones como “sistemas expertos borrosos”, “modelos borrosos”, “memorias asociativas borrosas”, “controladores de lógica borrosa”, etc...

Se componen de tres elementos. En primer lugar, un conjunto de reglas lógicas del tipo "múltiples antecedentes un consecuente" denominado "base de reglas". En segundo lugar, el llamado "diccionario" o base de datos que contiene la definición de los conjuntos borrosos asociados a los antecedentes y consecuentes de las reglas. Por último, hay que definir un mecanismo de razonamiento que lleve a cabo la inferencia (por ejemplo, el razonamiento aproximado mencionado anteriormente).

Las entradas a este tipo de sistemas pueden ser tanto números concisos como conjuntos borrosos. Cuando se trata de un número conciso hay que o bien considerarlo como un conjunto borroso particular, esto es, con pertenencia 1 para un solo valor de su universo y 0 en el resto, o bien convertirlo a número borroso en un proceso que denominaremos *borrosificación* (en inglés, *fuzzyfication*). La salida del sistema es un conjunto borroso que se obtiene a partir de las entradas, las reglas y el mecanismo de razonamiento elegido. Sin embargo, en muchas aplicaciones es necesario que la salida sea un valor numérico y no un conjunto borroso. En esos casos se realiza un proceso de *desborrosificación* mediante el cual, el conjunto borroso se convierte a un valor del universo de salida representativo de la conclusión obtenida.

El diagrama de bloques de un sistema de inferencia borroso (de ahora en adelante SIB) puede verse en la figura 1.14. El proceso básico puede resumirse en las siguientes etapas: obtención de los grados de compatibilidad de las entradas con los antecedentes de las reglas, obtención de las fuerzas de disparo de las reglas, agregación de los consecuentes cualificados y finalmente, en los casos que se requiera, la desborrosificación.

Una clasificación de los tipos más usuales de SIBs, se basa en el modelo utilizado para representar el consecuente de las reglas y en el método de agregación implicado. A continuación trataremos los más utilizados.

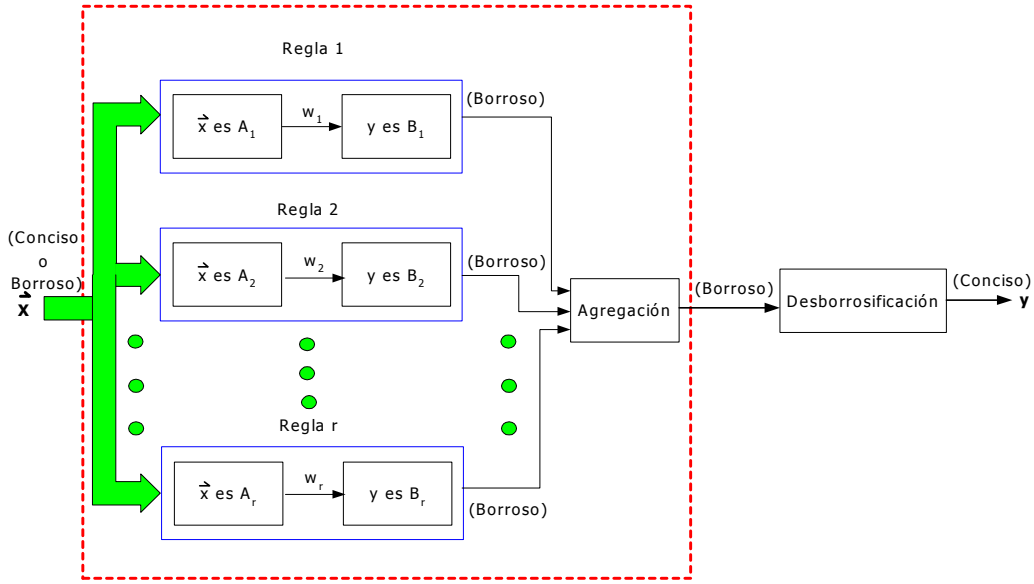


Figura 1.14. Diagrama de bloques de un sistema de inferencia borrosa. El cuadro rojo agrupa los componentes de un sistema de inferencia básico con salida borrosa.

1.5.1 Modelo borroso de Mamdani.

Comencemos por el modelo de Mamdani [Mamdani y Assilian, 1975], que fue propuesto para el control de una máquina de vapor mediante un conjunto de reglas de control lingüísticas obtenidas de la experiencia de los operadores de la máquina.

La estructura genérica de un modelo Mamdani se muestran en la figura 1.15. La *base de conocimiento* almacena la información acerca del problema en forma de reglas borrosas si-entonces. Los otros tres componentes constituyen el motor de inferencia borrosa. La *interfaz de borrosificación* establece un mapeo entre los valores concisos del dominio de entrada y los conjuntos borrosos definidos en el mismo universo de discurso. La *interfaz de desborrosificación* realiza la operación contraria: establece un mapeo entre los conjuntos borrosos definidos en el dominio de salida y los valores concisos definidos en el mismo universo.

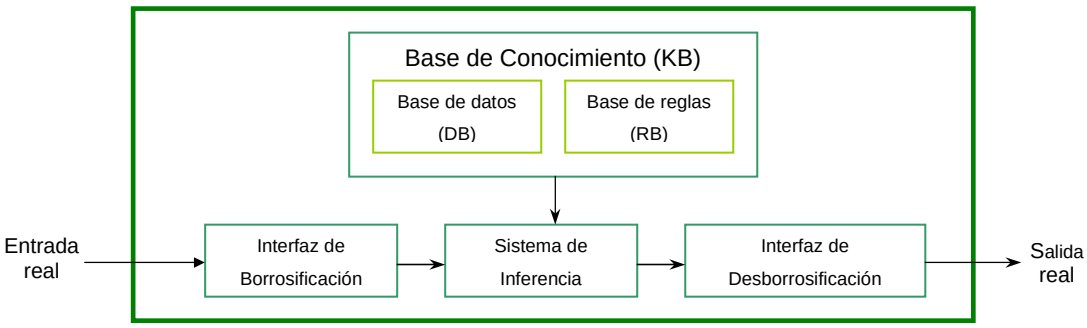


Figura 1.15. Estructura básica de un modelo Mamdani.

La base de conocimiento (en inglés, *knowledge base - KB*) es una parte fundamental de los sistemas tipo Mamdani. Almacena el conocimiento que modela las relaciones existentes entre las entradas y salidas del sistema. Basándose en este conocimiento, el proceso de inferencia obtendrá una salida asociada a una entrada observada. Esta base contiene dos niveles de información distintos: la semántica de las reglas borrosas (en forma de conjuntos borrosos) y las reglas lingüísticas que representan el conocimiento experto. Esta distinción conceptual se refleja en dos entidades separadas que constituyen la base de conocimiento:

- *Base de Datos o diccionario* (en inglés, *Data Base – DB*): contiene los conjuntos de términos lingüísticos considerados en las reglas lingüísticas y las funciones de pertenencia que definen la semántica de las etiquetas lingüísticas. Cada variable lingüística del problema tendrá asociada una partición borrosa con un conjunto borroso asociado a cada uno de sus términos lingüísticos. Además, contiene los factores de escala o funciones de escalado que se usan para realizar las transformaciones entre el universo de discurso donde los conjuntos borrosos están definidos y el dominio de las variables de entrada y salida del sistema.

- *Base de Reglas* (en inglés, *Rule Base – RB*): compuesta por la colección de reglas lingüísticas. Esta base puede presentarse en distintas estructuras. La más común es la lista de reglas aunque también es posible usar una tabla o matriz de decisión, que es una representación equivalente y más compacta en el caso de que tengamos pocas variables de entrada.

La interfaz de borrosificación permite a los modelos Mamdani manejar valores de entrada concisos. La borrosificación establece una correspondencia entre los valores de entrada concisos y los conjuntos borrosos definidos en el universo de discurso de esa entrada. La función de pertenencia del conjunto borroso A' definido sobre el universo de discurso X asociado al valor de entrada conciso x_0 se computa como:

$$A' = F(x_0) \tag{1.52}$$

donde F es un operador de borrosificación. El operador de borrosificación más comúnmente utilizado es el *singleton* con base en x_0 que presenta la siguiente función de pertenencia:

$$A'(x) = \begin{cases} 1, & si \quad x = x_0 \\ 0, & en \quad otro \quad caso \end{cases} \tag{1.53}$$

El modelo Mamdani toma como representación de los consecuentes, conjuntos borrosos que representan las valoraciones de una variable lingüística. Se admiten múltiples reglas con múltiples antecedentes. Cada regla debe tener el mismo conjunto de variables lingüísticas representadas en sus antecedentes. El método de inferencia borrosa, como se ha indicado anteriormente, admite un número de variantes.

Concretamente, en este modelo, estas variantes se refieren al operador Y-lógico donde normalmente se toma una T-norma, el operador O-lógico, que se suele implementar con una S-norma, el operador implicación, normalmente implementado con una T-norma para calcular el consecuente cualificado y el operador agregación corrientemente definido por una S-norma. En las figuras 1.16 y 1.17 se observan diferentes resultados según el método de inferencia utilizado. El modelo de Mamdani se suele utilizar con entradas representadas por números concisos. Entonces el procedimiento para obtener el grado de compatibilidad con los antecedentes es simple. El grado de compatibilidad de una entrada concisa en una regla es el grado de pertenencia de la entrada concisa medido con la función de pertenencia del conjunto borroso que representa el antecedente correspondiente a la variable lingüística a la que corresponde la entrada.

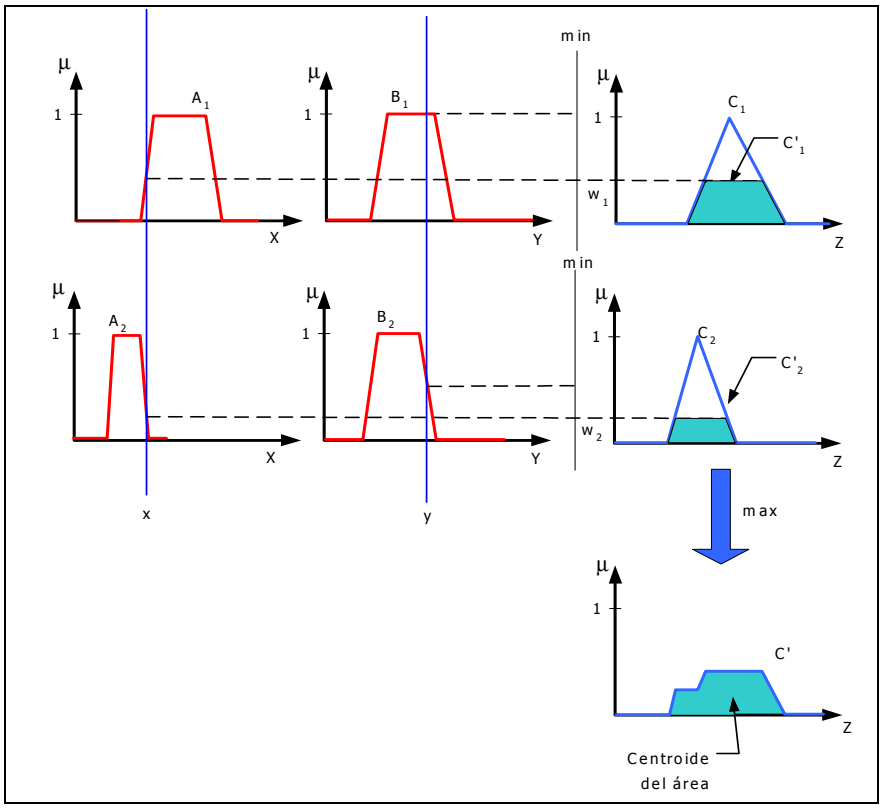


Figura 1.16. Sistema de inferencia borroso de Mamdani utilizando el mínimo y máximo para la T-norma y S-norma, respectivamente.

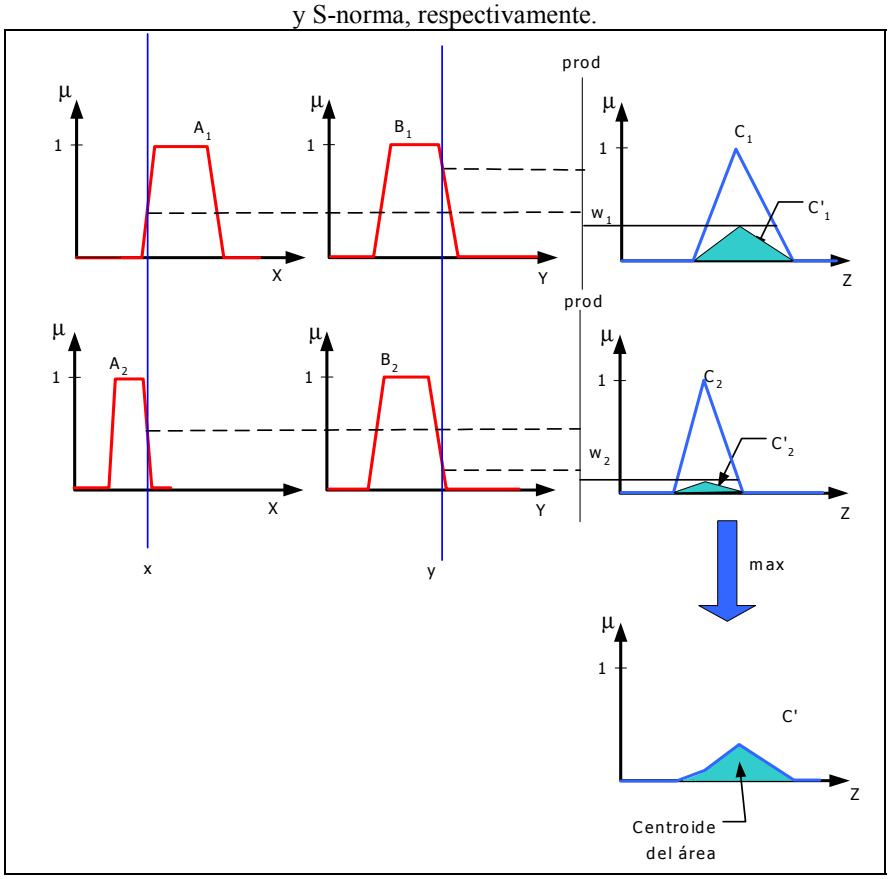


Figura 1.17. Sistema de inferencia borroso de Mamdani utilizando el producto y máximo para la T-norma y S-norma, respectivamente.

La interfaz de desborrosificación establece una correspondencia entre los conjuntos borrosos definidos en el dominio de salida y los valores concisos definidos en el mismo universo. Existen diversos métodos para realizar este proceso (gráficamente representados en la figura 1.18):

- *Desborrosificación basada en el centroide del área*. Se trata de calcular el centroide del conjunto borroso resultado de la agregación de todos los consecuentes cualificados.

$$z_{CDA} = \frac{\int_Z \mu_{C'}(z) z dz}{\int_Z \mu_{C'}(z) dz} \tag{1.54}$$

  En las figuras 1.16 y 1.17 ha sido el método utilizado. En la gráfica más a la derecha de la fila inferior se marca un punto en el interior de la zona sombreada, que proyectado sobre el eje horizontal es el número conciso asociado al conjunto borroso resultado de la inferencia.

- *Bisector del área.* Se trata de encontrar el valor numérico del elemento del universo que separa el área del conjunto borroso en dos mitades iguales.

- *Media de los máximos*. Se buscan aquellos elementos del universo donde la función de pertenencia del conjunto borroso alcanza un valor máximo y se calcula la media de estos puntos.

- *Mínimo de los máximos.* El procedimiento es igual al anterior, sólo que esta vez se toma el menor de los puntos del universo en lugar de la media.

- *Mayor de los máximos.* Lo mismo que en el apartado anterior pero esta vez se toma el mayor de los puntos.

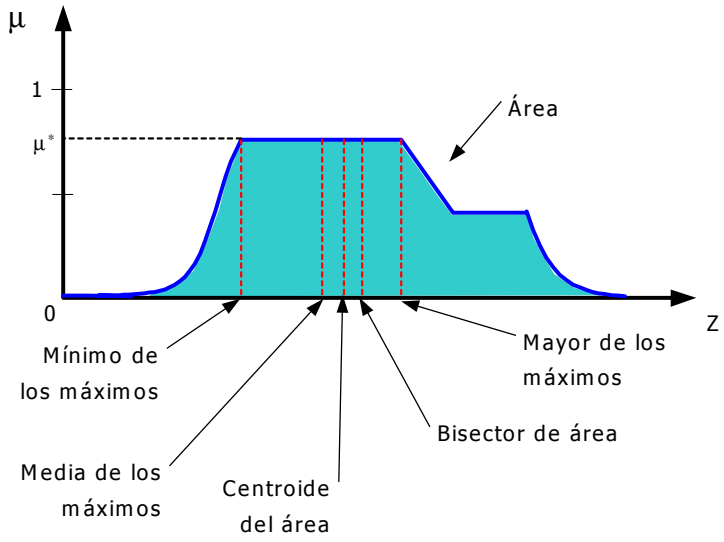


Figura 1.18. Diferentes métodos de desborrosificación para obtener una valor conciso a la salida.

Básicamente las tareas de inferencia y desborrosificación se pueden realizar de dos modos distintos: el modo *A-FATI* (en inglés, *first aggregate, then* infer) y el modo *B-FITA* (en inglés, *first infer, then aggregate*):

- Modo *A-FATI* (first aggregate, then infer): Mamdani sugirió originalmente el modo *A-FATI* [Mamdani, 1974]. En este modo, primero se agregan los conjuntos borrosos individuales resultantes de la inferencia de cada regla para obtener un conjunto borroso total. Luego se transforma este conjunto en un valor conciso mediante la aplicación de algún método de desborrosificación. Los métodos de desborrosificación más utilizados en esta aproximación son los métodos del centroide del área y de la media de los máximos.

- Modo *B-FITA* (first infer, then aggregate): en los últimos años el modo *B-FITA* se ha hecho popular [Sugeno y Yasukawa, 1993], [Cordón et al., 1997]. En este método, la contribución de los conjuntos borrosos resultantes de la inferencia de cada regla se considera separadamente. Se obtiene un número conciso de cada conjunto borroso por medio de una desborrosificación. Sobre estos números se aplicará un operador de selección para elegir el número conciso final. Esta aproximación evita hacer la agregación de las salidas de las reglas para formar el conjunto borroso final, lo que reduce el gasto computacional en comparación

con el modo A-FATI. Se suele emplear en aplicaciones que requieren una respuesta rápida en tiempo (aplicaciones en tiempo real con altas frecuencias de muestreo).

Cuando la agregación de los consecuentes cualificados se realiza mediante la "suma ponderada" de los mismos el sistema se denomina "sistema borroso aditivo". Una propiedad muy importante de esta clase de sistemas es que son *aproximadores universales de funciones*. Una regla borrosa define un "parche borroso", esto es, un subconjunto del espacio de entrada-salida del sistema. En ese subconjunto la regla alcanza una fuerza de disparo relevante produciendo diferentes valores de salida. En un sistema borroso aditivo, los "parches" producen un recubrimiento del espacio de entrada-salida de forma que estos subconjuntos no están completamente aislados entre sí, sino que dan lugar en las zonas de solape a valores de salida obtenidos como sumas ponderadas de los "parches" relevantes en esas zonas. Un sistema aditivo borroso puede aproximar de forma continua y uniforme cualquier función sobre un dominio compacto con cualquier grado de precisión que se requiera [Kosko, 1994].

La mejora en la aproximación se obtiene mediante la utilización de un mayor número de parches más pequeños en el sistema borroso. Esto permite aproximar más exactamente las zonas donde la función sufre cambios de tendencias (cambios en su primera derivada). Sin embargo, cuando la dimensionalidad del espacio entrada-salida es grande, el número de reglas necesario para producir un recubrimiento suficientemente fino se incrementa exponencialmente. Esto es lo que se denomina el *problema de la dimensionalidad de los sistemas borrosos.* Una forma de atacar este problema consiste en establecer procedimientos automáticos para la selección de reglas que aproximen de forma más precisa los datos [Kosko y Dickerson, 1995].

Además, las particiones en los espacios de entrada y salida son rígidas, lo que da lugar a una falta de flexibilidad del sistema, y si las variables de entrada son dependientes entre sí, es difícil encontrar una partición adecuada del espacio de entrada.

Entre las numerosas ventajas que el modelo Mamdani presenta, podemos citar las siguientes:

- En primer lugar, permite combinar el conocimiento con reglas generadas a partir de conjuntos de datos que describen las relaciones entre las entradas y salidas del sistema.

- Posee un alto grado de libertad para seleccionar las interfaces de borrosificación y desborrosificación así como el método de inferencia en sí mismo. Esta propiedad permite adaptar el motor de inferencia a las características específicas del problema considerado.

- Permite una formulación del conocimiento muy flexible a la vez de interpretable. Cada regla si-entonces es una descripción de una condición–acción de clara interpretación para un humano. Por esta razón, este tipo de sistemas son conocidos como *sistemas Mamdani descriptivos o lingüísticos*. Debido a esta propiedad son muy usados en aplicaciones donde la interpretabilidad del modelo es importante, como por ejemplo, en control borroso y en modelado lingüístico.

Existen diversas variantes del modelo de Mamdani cuyo objetivo es aprovechar las ventajas del modelo e intentar paliar las desventajas. Para lograr esto, estas variantes intentan hacer la estructura de las reglas lingüísticas más flexible. A continuación, se presentan las dos variantes más comunes.

1.5.1.1 Modelo borroso Mamdani DNF.

La primera extensión del modelo de Mamdani introduce una nueva estructura para las reglas llamada *forma disyuntiva normal* (en inglés, *disjunctive normal form* - *DNF*) que tiene la siguiente sitaxis:

$$\text{Si } x_1 \text{ es } \widetilde{A}_1 \text{ y ... y } x_n \text{ es } \widetilde{A}_n \text{ entonces } y \text{ es } B \tag{1.55}$$

donde cada variable de entrada x_i toma como valor un conjunto de términos lingüísticos $\widetilde{A}_i$, cuyos miembros están unidos por un operador disyuntivo, mientras que a la variable

de salida se le asigna una única etiqueta lingüística. Por lo tanto, la sintaxis completa para el antecedente de la regla es el siguiente:

$$x_1 \text{ es } \widetilde{A}_1 = \{A_{11} \text{ o } ... \text{ o } A_{1l_1}\} \text{ y } ... \text{ y } x_n \text{ es } \widetilde{A}_n = \{A_{n1} \text{ o } ... \text{ o } A_{nl_n}\} \qquad (1.56)$$

Por ejemplo, supongamos que tenemos tres variables de entrada, x_1, x_2 y x_3 y una variable de salida y, de modo que los conjuntos de términos lingüísticos D_i, F asociados a cada una son:

$$D_1 = \{A_{11}, A_{12}, A_{13}\} \quad D_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}\} \quad D_3 = \{A_{31}, A_{32}\} \quad F = \{B_1, B_2, B_3\}$$

En este caso, una posible regla tipo DNF podría ser la siguiente:

Si x_1 es $\{A_{11}$ o $A_{13}\}$ y x_2 es $\{A_{23}$ o $A_{25}\}$ y x_3 es $\{A_{31}$ o $A_{32}\}$ entonces y es B_2

Esta aproximación se puede consultar con más detalle en [González et al., 1993], [Magdalena, 1997].

1.5.1.2 Modelo Mamdani aproximado.

Mientras que la anterior variación del modelo Mamdani no implicaba una importante pérdida de interpretabilidad, esta segunda extensión consigue aumentar la exactitud del modelo a costa de reducir la interpretabilidad del mismo. Por este motivo, son denominados *modelos Mamdani aproximados*, para diferenciarlos de los *modelos Mamdani descriptivos o lingüísticos* convencionales.

La estructura del modelo aproximado es similar a la del descriptivo (figura 1.15). La diferencia está en que cada regla del modelo aproximado define sus propios conjuntos borrosos mientras que las reglas del modelo descriptivo tienen asociadas etiquetas lingüísticas que señalan a conjuntos borrosos particulares de una partición lingüística de la variable lingüística. Por lo tanto, una regla del modelo aproximado tiene la siguiente forma:

$$\text{Si } x_1 \text{ es } A_1 \text{ y } \ldots \text{ y } x_n \text{ es } A_n \text{ entonces } y \text{ es } B \tag{1.57}$$

En esta regla las variables de entrada x_i y la variable de salida y son variables borrosas en vez de variables lingüísticas, y por lo tanto, A_i y B son conjuntos borrosos definidos independientemente y sin una interpretación lingüística intuitiva. En otras palabras, las reglas de naturaleza aproximada son de *semántica libre* mientras que las reglas descriptivas operan en el contexto formulado por la semántica lingüística.

Por lo tanto, los modelos aproximados no contienen una *base de datos (DB)* que define un contexto semántico con variables y términos lingüísticos. Las entidades separadas que existen en el modelo descriptivo (base de datos y base de reglas) desaparecen en el modelo aproximado. En su lugar aparece una *base de reglas borrosas* (en inglés, *fuzzy rule base – FRB*) donde cada regla es como se muestra en la figura 1.19.

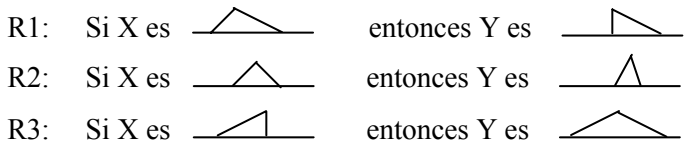


Figura 1.19. Ejemplo de base de reglas borrosas.

Un ejemplo de utilización de un modelo aproximado se puede encontrar en [Carse et al., 1996]. Los modelos aproximados tienen algunas ventajas sobre los modelos lingüísticos que los hace especialmente útiles para algunos tipos de aplicaciones:

- La principal ventaja es que cada regla emplea sus propios conjuntos borrosos, lo que incorpora grados de libertad adicionales y aumenta la expresividad.

- El número de reglas se puede adaptar a la complejidad del problema. Las relaciones entrada-salida se modelan con varias reglas, siendo posible aumentar el número de reglas a medida que aumenta la complejidad del problema. Por lo tanto, los modelos aproximados constituyen una solución potencial al problema de la dimensionalidad.

Estas propiedades permiten al modelo aproximado tener más exactitud que los modelos lingüísticos en dominios de problemas complejos. Sin embargo, los modelos aproximados conllevan una serie de desventajas:

- La principal desventaja en comparación con los modelos descriptivos es la degradación en términos de interpretabilidad del modelo debido a que las variables borrosas que emplea no tienen interpretación lingüística. A diferencia de otros tipos de modelos aproximados, como por ejemplo, las redes neuronales, que almacenan conocimiento implícitamente, el conocimiento en un modelo Mamdani aproximado permanece explícito ya que el comportamiento del sistema está descrito por reglas locales. Por lo tanto, estos modelos pueden ser considerados como un compromiso entre la aparente interpretabilidad del modelo descriptivo y el comportamiento de una caja negra, típico de modelos implícitos.

- La capacidad de aproximar un conjunto de datos de entrenamiento con exactitud puede originar una pobre generalización sobre otros datos de entrada no vistos en el entrenamiento.

A consecuencia de estas propiedades, el modelado borroso [Bardossy y Duckstein, 1995] constituye la mayor aplicación de los modelos Mamdani aproximados, por ser más relevante en estas aplicaciones la exactitud que la habilidad descriptiva del modelo.

1.5.2 Modelo borroso de Takagi-Sugeno-Kang.

Aunque el método de Mamdani ha sido muy utilizado, en este trabajo se toma como referencia más frecuentemente el modelo de Sugeno, también conocido como modelo TSK [Takagi y Sugeno, 1985], [Sugeno y Kang, 1988]. La diferencia fundamental respecto al modelo de Mamdani es que los consecuentes de las reglas dejan de ser conjuntos borrosos para convertirse en funciones, de forma que una regla típica en un modelo TSK vendría dada por:

$$\text{Si } x \text{ es } A \text{ e } y \text{ es } B \text{ entonces } z = f(x,y) \tag{1.58}$$

Normalmente la función utilizada como consecuente suele ser un polinomio (por ejemplo, un polinomio de primer orden):

$$z = f(x_1,x_2,...,x_n) = p_1x_1 + \ldots + p_nx_n + p_0 \tag{1.59}$$

Los más usados con diferencia son el polinomio de orden cero (constante y el polinomio de orden 1 (función lineal de las entradas del sistema). En el primer caso se tienen el denominado modelo de Sugeno de orden 0 y en el segundo, el modelo de Sugeno de orden 1. Nuestra aproximación es de orden 0.

La salida de un modelo TSK compuesto por m reglas se obtiene como una suma pesada de las salidas individuales de cada regla, y_i, $i=1,...,m$, tal y como se muestra a continuación:

$$\frac{\sum_{i=1}^{m} h_i \cdot y_i}{\sum_{i=1}^{m} h_i} \tag{1.60}$$

donde $h_i = T(A_{i1}(x_1),..., A_{in}(x_n))$ es el grado de encaje entre la parte antecedente de la i-ésima regla y las entradas actuales del sistema, $x_0 = (x_1,...,x_n)$. T es un operador conjuntivo modelado por una T-norma. Las T-norma más utilizadas en este tipo de modelos son la T-norma mínimo y la T-norma producto algebraico. En la figura 1.20 se muestra una representación gráfica de la estructura de estos modelos borrosos.

Figura 1.20. Estructura básica de un modelo borroso TSK.

En el modelo de Sugeno no es preciso un método de desborrosificación ya que el consecuente cualificado es un número conciso. En su lugar se utiliza un promedio ponderado o una suma ponderada. La suma ponderada es la suma de los valores

obtenidos de las funciones en los consecuentes de las reglas ponderadas por las fuerzas de disparo de cada regla. El promedio ponderado consiste en un cálculo más: la suma ponderada es además dividida por la suma de las fuerzas de disparo de las reglas, tal y como se puede observar en la figura 1.21.

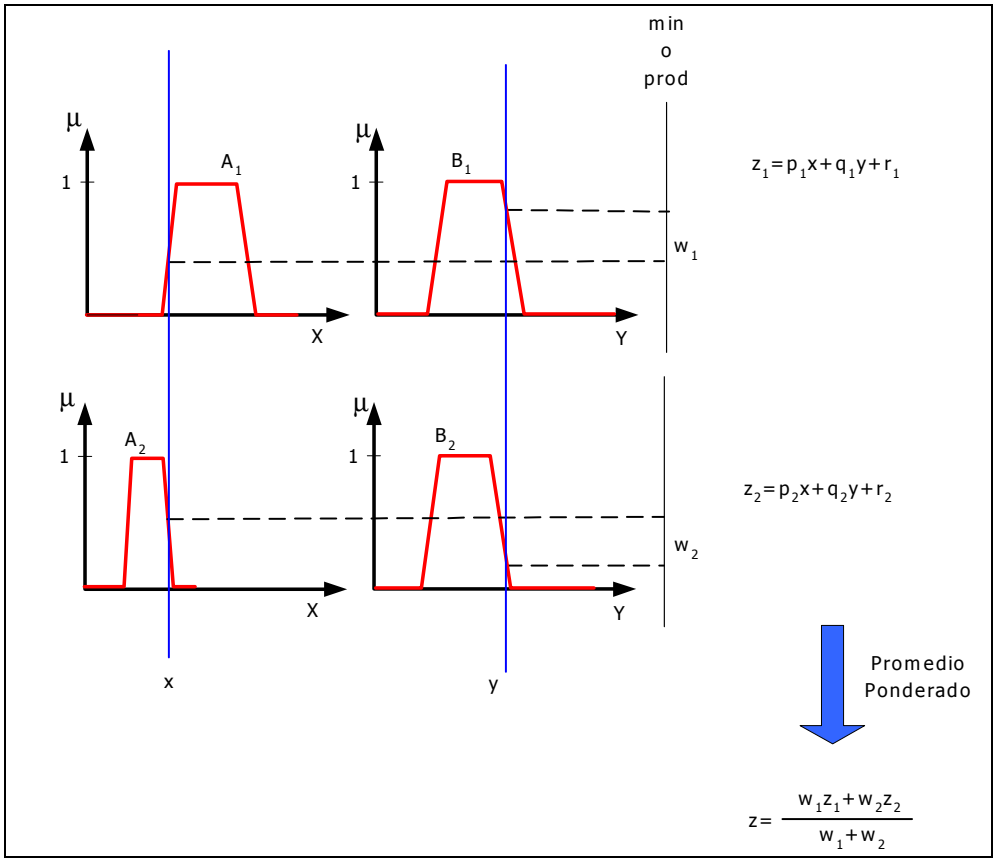


Figura 1.21. Modelo de inferencia borrosa de Takagi-Sugeno-Kang.

Este modelo se ha mostrado como el más efectivo para el procesamiento de datos, ya que elimina el paso de la desborrosificación mediante un método simple matemáticamente tratable. El modelo de Sugeno de orden 0 es asimilable bajo ciertas condiciones a una red de funciones de base radial, con las ventajas que esto le confiere al poder aprovecharse de algoritmos de ajuste de parámetros para el aprendizaje de conjuntos de datos [Jang y Sun, 1993].

La mayor ventaja de estos sistemas es que presentan un conjunto de ecuaciones compactas del sistema que permite estimar los parámetros p_i por medio de métodos clásicos, lo que facilita el proceso de diseño. Sin embargo, el mayor inconveniente de estos modelos es que son más difíciles de interpretar que los modelos borrosos

Mamdani, debido a que la estructura de los consecuentes de las reglas es difícil de entender para un experto humano.

1.5.3 Modelo borroso de Tsukamoto.

El modelo de Tsukamoto es otro tipo de SIB, donde la fuerza de disparo de cada regla es utilizada directamente para obtener un número conciso de la función de pertenencia monótonamente creciente o decreciente que representa el consecuente de la regla. Si $\mu_C(z)$ es la función de pertenencia del consecuente y ω es la fuerza de disparo, el número conciso asociado a la regla (el equivalente al consecuente cualificado) se obtiene como $z = \mu_C^{-1}(\omega)$. Finalmente la salida del sistema se obtiene como el promedio ponderado o la suma ponderada de forma similar al modelo de Sugeno.

1.6 Generación de conjuntos de reglas borrosas.

La exactitud de un modelo borroso depende de dos aspectos: el modo en que se implementa el proceso de inferencia (la elección de los operadores borrosos ya citados anteriormente que se utilizan en la inferencia) y la composición del conjunto de reglas borrosas.

Para generar el conjunto de reglas borrosos se requieren las siguientes tareas de diseño:

1. Selección de las variables de entrada y salida relevantes en el problema entre todas las variables existentes en el sistema. Esto lo suele hacer el experto humano. También se puede hacer por medio de métodos estadísticos basados en analizar la correlación existente entre las variables disponibles o por métodos combinacionales, que analizan la influencia de los subconjuntos compuestos por diferentes combinaciones de variables.

2. Cuando se trabaja con modelos de naturaleza descriptiva (modelos Mamdani lingüísticos o modelos TSK) que emplean variables lingüísticas en los antecedentes de las reglas, hay que definir la estructura de la base de datos (DB) que contiene la semántica de los términos que las variables lingüísticas de entrada

y salida pueden tomar como valor. Esto implica algunas de las siguientes subtareas:

- Definición de los factores de escala.

- Elección de los posibles términos lingüísticos para cada variable lingüística, lo que nos permite determinar la granularidad deseada en el sistema.

- Elección de los tipos de funciones de pertenencia que se emplearán: triangulares, trapezoidales, gaussianas o con forma exponencial principalmente. Las dos últimas tienen la ventaja de presentar una forma más suave y, por tanto, computacionalmente son más simples.

  Se han realizado distintos estudios para analizar la influencia de la forma de estas funciones sobre la exactitud del modelo [Baglio et al., 1993], [Chang et al., 1991]. En [Delgado et al., 1998] se enuncia que las funciones de pertenencia con forma trapezoidal pueden aproximar adecuadamente a las anteriores, presentando la ventaja de su sencillez.

- Definición de las funciones de pertenencia del conjunto borroso específico asociado a cada etiqueta lingüística.

Cuando se trabaja con modelos aproximados, la única tarea que hay que realizar es seleccionar el tipo de funciones de pertenencia que se emplearán en el conjunto de reglas borrosas.

3. Derivación de las reglas lingüísticas o aproximadas que formarán parte del conjunto de reglas borrosas del sistema. Para esta tarea, hay que determinar el número de reglas y su composición, definiendo las partes de antecedentes y consecuentes.

Para generar las reglas borrosas se puede disponer de distintos tipos de información: numérica y lingüística. La primera se obtiene de observar el sistema

estudiado y la segunda, del experto humano. Cuando la información proviene del conocimiento de un experto humano, éste especifica las etiquetas lingüísticas asociadas a cada variable lingüística, la estructura de las reglas de la base de reglas y el significado de cada etiqueta. Este método es el más sencillo si el experto es capaz de expresar su conocimiento en forma de reglas lingüísticas, lo cual no siempre es posible. En los casos en que esta tarea es complicada, se pueden usar métodos de aprendizaje inductivos para encontrar la base de reglas, como por ejemplo, variantes del método de los mínimos cuadrados [Bardossy y Duckstein, 1995], [Takagi y Sugeno, 1985], métodos descendentes [Nomura et al.,1991], métodos híbridos de los dos anteriores [Jang, 1993], redes neuronales [Takagi y Hayashi, 1991], [Takagi et al., 1992], técnicas de clustering [Delgado et al., 1997], [Yoshinari et al., 1993], y algoritmos evolutivos [Cordón et al., 2001], entre otros. Los sistemas en los que un algoritmo evolutivo aprende el conjunto de reglas borrosas son llamados *sistemas borrosos genéticos*, y son el tema central del trabajo presentado en esta tesis.

1.7 Propiedades básicas del conjunto de reglas borrosas.

Algunas propiedades de los conjuntos borrosos son beneficiosas para aumentar la exactitud del modelo. A continuación, se citan las más importantes:

Completitud. Un sistema borroso debe cumplir esta propiedad. Para una entrada arbitraria del sistema, x_0, al menos una de las reglas borrosas de la base se debe disparar. La salida global del sistema, es decir, el conjunto borroso obtenido de combinar las salidas de todas las reglas no debe estar vacío [Bardossy y Duckstein, 1995]. Como se puede apreciar, esta definición se refiere a aquellos modelos borrosos cuya inferencia y desborrosificación se realizan con el método *A-FATI* ya que estos tipos de sistemas son los únicos donde la agregación de conjuntos individuales borrosos no vacíos puede dar como resultado una salida global vacía. Por lo tanto, la propiedad de completitud se expresa como se muestra a continuación:

$$\forall x_0 \in X, \quad Altura(S(x_0)) \geq \sigma \tag{1.61}$$

donde $S(x_0)$ es el conjunto borroso global de salida, *Altura*() devuelve la altura del conjunto borroso y $\sigma \in (0,1]$ es una cota mínima de altura (mayor que cero). Esta propiedad también recibe el nombre de σ*-completitud* y es más restrictiva que la definición inicial de completitud, ya que incluye un grado de satisfacción en la completitud.

Esta propiedad es muy útil en la práctica debido al hecho que puede ser considerada en el proceso de generación del conjunto de reglas borrosas. En el caso en que las reglas borrosas existentes en la base no se disparen con un grado mayor que σ para una entrada determinada al sistema, será necesario añadir reglas nuevas o modificar las ya existentes.

Consistencia. Un conjunto de reglas borrosas si-entonces es consistente si no contiene reglas contradictorias. Este concepto es fácil de entender cuando las reglas son clásicas, pero tiene diversas interpretaciones en el caso en que las reglas sean borrosas.

En principio, un conjunto de reglas borrosas es inconsistente si tiene reglas con los mismos antecedentes y distintos consecuentes, pero esta definición conlleva algunas incoherencias. Distintos autores han estudiado esta definición y han llegado a distintas conclusiones.

Por ejemplo, en [Driankov et al., 1993] se llega a la conclusión de que dos reglas se consideran inconsistentes cuando tienen el mismo antecedente y los consecuentes son mutuamente excluyentes.

En [Harris et al., 1993] se consideran que las reglas consistentes tienen variaciones similares entre los conjuntos borrosos que definen los antecedentes y los que definen los consecuentes, y se propone un *índice de inconsistencia* para medir este aspecto.

En [González y Pérez, 1998] la propiedad de consistencia está basada en los conceptos de *ejemplos negativos y positivos.* Un ejemplo se considera positivo para una regla borrosa si encaja con el antecedente y consecuente de la regla, y será considerado negativo cuando hay encaje con el antecedente pero no con el consecuente. Una regla borrosa se considera inconsistente cuando tiene ejemplos negativos asociados a ella. Sin embargo, una regla borrosa que cubre un número pequeño de ejemplos negativos y un número muy grande de ejemplos positivos podría no ser considerada inconsistente. En

[González y Pérez, 1998] se introduce la propiedad de la *k-consistencia* que es menos estricta que la definición previa, ya que sólo considera la cardinalidad de los conjuntos de ejemplos positivos y negativos de una regla y establece que una regla borrosa es k-consistente si el número de ejemplos negativos es menor o igual que un porcentaje 100**k* del número de ejemplos positivos, siendo $k \in [0,1]$ un parámetro.

Baja complejidad. Esta propiedad hace referencia al número de reglas borrosas que compone la base [Lee, 1990]. Es preferible tener el menor número de reglas posible en la base, ya que esto aumenta la comprensibilidad del modelo y disminuye el gasto computacional del proceso de inferencia. Esta propiedad es importante en aplicaciones donde la rapidez del proceso y la simplicidad de la base son importantes, como por ejemplo, en aplicaciones de control, o en aplicaciones donde pese más la interpretabilidad del sistema, como por ejemplo, en aplicaciones de modelado lingüístico.

Redundancia. Un punto del espacio de las entradas podría estar cubierto por más de una regla borrosa si los antecedentes de las reglas de la base se solapasen [Bardossy y Duckstein, 1995]. La existencia de reglas redundantes puede degradar la eficiencia del sistema global, por lo tanto, es importante analizar la redundancia del conjunto de reglas borrosas para poder eliminar las reglas innecesarias.

La simplificación de la base de reglas de un sistema borroso es beneficiosa desde el punto de vista del modelado de sistemas no lineales por tres motivos: se mejora la interpretabilidad del sistema [Setnes et al., 1998], aumenta su capacidad de generalización [Yen y Wang, 1999] y se reduce la complejidad computacional y de espacio necesario para el almacenamiento [Yam et al., 1999]. Para simplificar la base de reglas, se debe evaluar la utilidad de cada regla analizando su impacto sobre el comportamiento global del sistema.

1.8 La doble utilidad de la lógica borrosa. Principales campos de aplicación.

Para terminar esta breve introducción a la lógica borrosa recordemos las dos capacidades fundamentales de los sistemas basados en lógica borrosa. Estas reflexiones se deben a [Yen, 1999].

Tradicionalmente la lógica borrosa se contempló desde la perspectiva del manejo de la incertidumbre y la vaguedad en los términos. De hecho, el manejo de la indefinición entre clases es presentada por Lofti A. Zadeh como la principal motivación al introducir la noción de conjunto borroso en su artículo de 1965 en *"Information and Control"* [Zadeh, 1965].

Sin embargo, el relanzamiento de la lógica borrosa a finales de los ochenta y durante toda la década de los noventa se debe en buena medida al enfoque de los sistemas borrosos como aproximadores de funciones. Es más, la gran ventaja que es explotada se refiere al hecho de poder controlar la dicotomía entre precisión de la representación y el coste en el desarrollo del modelo aproximado.

En la figura 1.22 podemos ver dos gráficas. En la gráfica de la izquierda el eje horizontal representa la *precisión* y el eje vertical representa el *coste*. Por otra parte, en la gráfica de la derecha, el eje horizontal sigue representando la precisión, mientras el eje vertical es ahora la *utilidad*. La curva de la gráfica coste-precisión muestra como normalmente el coste asociado a alcanzar una precisión determinada suele seguir un comportamiento exponencial, mientras que la utilidad real no se incrementa de la misma manera sino que tiende a saturarse o a crecer de forma muy lenta. La clave de la aplicación exitosa de los sistemas borrosos está en el aprovechamiento de la zona intermedia sombreada en ambas gráficas.

Las principales aplicaciones de los sistemas borrosos se encuentran en los campos del modelado lingüístico, del control borroso y de la clasificación borrosa.

- Dentro del modelado borroso podríamos citar como relevantes las siguientes contribuciones: aplicaciones en el campo de la economía [Yuize et al., 1991], en el modelado de la media de temperatura diaria [Bardossy y Duckstein, 1995], en la medicina [Bardossy y Duckstein, 1995], [Lee y Takagi, 1996] y en la ingeniería eléctrica [Cordón et al., 1998b].

- La primera aplicación del control borroso fue propuesta en [Umbers y King,1980]. Existen muchas aplicaciones del control borroso en la automatización industrial, ejemplos representativos son: control de plantas de tratamiento de agua y plantas incineradoras, ascensores, reactores nucleares, lavadoras automáticas y robótica, entre otras. Algunos de estos trabajos se pueden consultar en [Lee, 1990], [Bardossey y Duckstein, 1995], [Berenji, 1992], [Hirota, 1993], [Saffiotti et al., 1995], [Lazzerini et al., 1999].

- En el campo de la clasificación borrosa, dentro del que se enmarca el presente trabajo, destacan las siguientes aplicaciones: segmentación de imágenes geográficas y de satélites [Chi et al., 1996], [Binaghi et al., 1996], reconocimiento de caracteres [Chi et al., 1996], aplicaciones en el diagnóstico médico [González et al., 1995], clasificación de variables meteorológicas [Bardossy y Durkstein, 1995] y monitorización de señales médicas, donde podemos citar los trabajos de [Steimann, 1996], [Steimann, 1997], [Steimann, 2001], [Barro et al., 2001], [Moreno et al., 2001a], [Moreno et al., 1997].

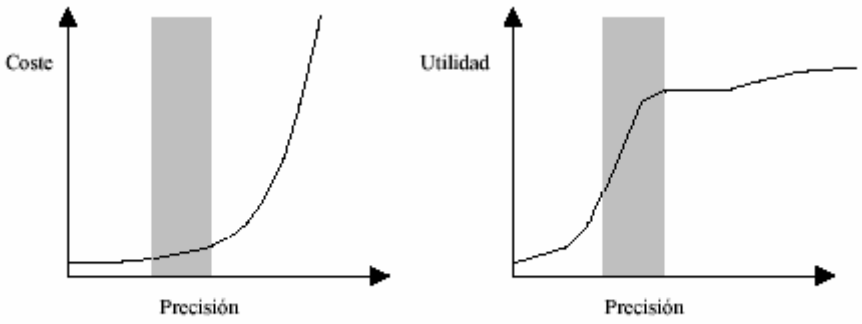


Figura 1.22. Representación de relaciones típicas entre coste y precisión y utilidad y precisión en los sistemas que aproximan modelos. Los sistemas basados en lógica borrosa están capacitados para explotar la zona sombreada.

Capítulo 2

Las máquinas de estados finitas borrosas.

2.1 Introducción.

La máquina finita de estados borrosa (en inglés, *fuzzy finite state machine – FFSM*) es una extensión del autómata finito determinista (en inglés, *finite deterministic automaton – FDA*). Un autómata finito determinista es un sistema con estados internos. Estos estados internos cambian en función de la activación que tenían previamente y en función de la entrada externa.

Este concepto se traslada a la lógica borrosa de distintos modos, dependiendo de la aplicación. En todos los casos se conserva la naturaleza del autómata: se obtiene una nueva situación interna de los estados en función de la situación anterior de los estados y del valor de la entrada externa. Una diferencia importante entre el autómata borroso y el autómata finito es que en la implementación del autómata borroso no se activa un único estado, sino que se activan todos simultáneamente pero con distintos grados de activación.

2.2 Ejemplos de aplicaciones.

El modelo de una máquina de estados finita borrosa se caracteriza principalmente por dos propiedades: los eventos externos producen transiciones graduales entre los estados internos del sistema y todos los estados del sistema ocurren a la misma vez pero con niveles de activación distintos. Para entender esto mejor, comentamos a continuación algunas aplicaciones.

Este algoritmo tiene utilidad en varios campos, principalmente en el reconocimiento de patrones, modelado de agentes inteligentes, interfaces hombre-máquina y en la ingeniería de control.

En [Reyneri, 1997], la aplicación de la máquina de estados finita borrosa surge de modo natural por el hecho de necesitarse un autómata finito determinista en el algoritmo de control. La lógica borrosa permite introducir conocimiento humano en el diseño del controlador, además de permitir el entrenamiento del controlador mediante "ejemplos", pero la mayoría de los sistemas borrosos son sistemas sin realimentación, y por tanto, sin memoria. Esto limita la aplicabilidad de estos sistemas en el ámbito del control, ya que algunos controladores deben tener memoria. En este sentido, las máquinas finitas de estados borrosas constituyen una de las opciones más beneficiosas, por ser sistemas borrosos recurrentes (con memoria).

Las plantas que operan en un conjunto bien definido de estados pueden ser controladas por un único controlador o por un conjunto de controladores más sencillos que interactúan con un autómata finito determinista (FDA). En este segundo enfoque, el FDA se usa normalmente como un interruptor para conmutar entre distintos controladores, dependiendo de la región del espacio de estados en donde se encuentre la planta. Esta estrategia tiene varias ventajas:

- El controlador global se divide en un conjunto de controladores más sencillos, y cada uno de estos controladores es a menudo tan sencillo como un controlador lineal.

- Los procesos directos de diseño de los controladores también se simplifican, ya que cada controlador será entrenado sólo sobre un subconjunto limitado del espacio de estados.

- Cada controlador tiene un tamaño reducido y se puede implementar de un modo óptimo.

- Los controladores pueden ser entrenados de un modo independiente, por lo tanto, el entrenamiento de uno de ellos no afecta a ninguno de los otros.

Pero también cuenta con alguna desventaja:

- Las transiciones entre controladores podrían originar cambios abruptos en el comportamiento de la planta, lo que puede provocar grandes aceleraciones, picos de corriente, vibraciones, etc.

- Para reducir estas discontinuidades, se necesita un subsistema adicional de suavizado (lo que a menudo empeora la eficiencia global del sistema) o los controladores individuales se deben diseñar de modo que se cuide más la transición entre estados (pero esto aumenta la complejidad y la duración del entrenamiento de los controladores).

Para solventar este problema se usa la generalización del FDA a la FFSM. Con una FFSM la característica del controlador es más suave y más fácil de entrenar o ajustar [Reyneri, 1997]. La mayor diferencia entre la FFSM y el FDA es que las transiciones entre los estados de la FFSM son disparadas por variables borrosas en vez de por eventos concisos, y estas transiciones entre estados también son borrosas. De esto se deduce inmediatamente que, en cualquier instante, el sistema completo no se encuentra en un estado bien definido, sino que se encuentra en varios estados a la misma vez, cada uno con un nivel de activación diferente. Las transiciones entre estados son, por lo tanto, más suaves y lentas, incluso si los controladores activados no están diseñados para suavizar las transiciones. Como consecuencia de esto, los controladores individuales son tan sencillos como el tradicional PID, pero cada uno diseñado para distintos objetivos. La FFSM cuida la suavidad de las discontinuidades y el sistema global es mucho más sencillo y fácil de diseñar y ajustar que un controlador tradicional con las mismas características.

Pero esta estrategia tiene una gran desventaja. Como la FFSM suele estar en más de un estado a la vez, es necesario procesar más de un controlador al mismo tiempo. Esto aumenta el tiempo de cómputo y, en algunos casos, la hace más lenta que la FDA.

Las acciones de control que lleva a cabo un controlador borroso se establecen por medio de una colección de reglas de control borrosas, que expresan una dependencia cualitativa entre las salidas *Y* (por ejemplo, las variables de control) y las entradas *X* (por ejemplo las variables de estado).

Existen distintas definiciones posibles de controladores borrosos. En [Lazzerini et al., 1999], se presenta un modo de diseñar controladores para plantas no lineales basado en sistemas en tiempo real de alta eficiencia con controladores adaptativos. Las distintas metodologías del control inteligente (control borroso, control con redes neuronales y control genético) ofrecen soluciones para problemas de control no lineales o, al menos, soluciones alternativas para problemas clásicos. Cada una de estas aproximaciones tiene sus propias ventajas y desventajas, lo que explica por qué han sido aplicadas solamente en campos muy específicos. Por este motivo, en el trabajo referenciado, se presenta una aproximación híbrida que combina algoritmos de control borroso, redes neuronales, control lineal, algoritmos de optimización (optimización genética) y FFSM.

En el modelo de controlador borroso utilizado en [Lazzerini et al., 1999], se define una cuantización del dominio para cada posible entrada x_i. Un nivel de cuantización corresponde a un conjunto borroso caracterizado por su etiqueta (por ejemplo, *cero*, *positivo, negativo,* etc) y su función de pertenencia. En este modelo concreto, las funciones de pertenencia son sigmoides.

Se usan reglas de control borrosas con el siguiente formato:

$$R^k\text{: si } x_1 \text{ es } A_1^k \text{ y } \ldots \text{ y } x_n \text{ es } A_n^k \text{ entonces } g^j = f_k(...) \qquad (2.1)$$

donde $x_1, \ldots, x_n$ son las variables de entrada del controlador borroso, $A_1^k, \ldots, A_n^k$ son los conjuntos borrosos que aparecen en la regla R^k, g^j es la variable de control, y $f_k(...)$ es la función de control asociada a la regla R^k. $f_k(...)$ podría ser una función de $x_1, \ldots, x_n$, o de los bloques de entrada/salida, por ejemplo, la función de transferencia de un controlador lineal.

Dados los valores concisos de las variables de entrada, el controlador borroso calcula la activación a_k de la regla k interpretando el conectivo "y" como el operador mínimo:

$$a_k = min(\mu_{A_1^k}(x_1), \mu_{A_2^k}(x_2), ..., \mu_{A_n^k}(x_n)) \tag{2.2}$$

donde $\mu_{A_{1i}^k}(x_i)$ es la función de pertenencia del conjunto borroso A_i^k.

Finalmente, el valor asignado a la variable de salida se calcula como la media de los valores $f_k(...)$ pesados por las activaciones de las reglas:

$$g^j = \frac{\sum_k a_k f_k(...)}{\sum_k a_k} \tag{2.3}$$

Bajo este planteamiento, se puede entender el diagrama de la figura 2.1. La acción de control se obtiene por medio de una combinación lineal de las salidas de varios controladores. Los pesos aplicados son los niveles de activación de cada estado de la máquina de estados finita borrosa. La entrada externa de la máquina se obtiene preprocesando las salidas de los sensores de la planta.

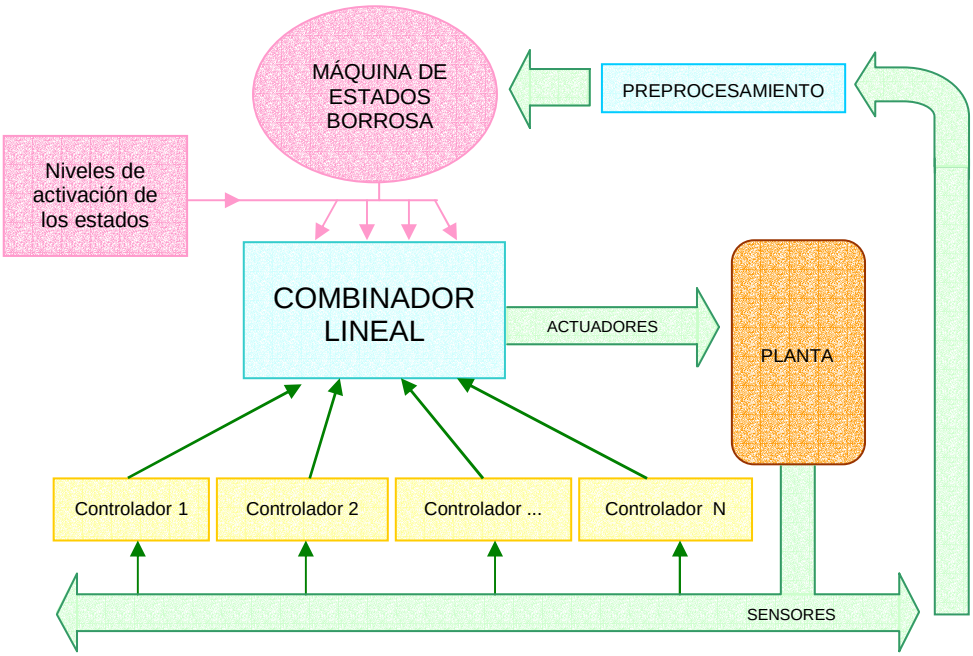


Figura 2.1. Uso de una máquina de estados finita borrosa en lugar de un autómata finito determinista en una aplicación de control.

Otro problema interesante que se puede aproximar por medio de máquinas de estados borrosas es el control y la estabilización de sistemas no holónomos. En la robótica móvil se han empleado para seguir trayectorias, control de brazos de robots, etc.

En otro ámbito de aplicación, [Surmann y Maniadakis, 2001] utilizan un sistema borroso recurrente basado en reglas dentro de un sistema predictivo, que se aplica en la predicción de series temporales. En concreto, se aplica el sistema predictivo sobre los conocidos datos de referencia de Box y Jenkins del comportamiento de un gas en un horno y la concentración de CO_2. La tarea que se proponen es construir un modelo de la base de reglas a partir de un conjunto de datos de referencia que identifica el proceso. La entrada al sistema es el flujo de gas dentro del horno y la salida es la concentración de CO_2 en el gas agotado. Para realizar esta tarea de predicción, se propone un método que aprende sistemas borrosos recurrentes con variables borrosas ocultas. Los sistemas borrosos recurrentes utilizados aproximan la relación que existe en este proceso dinámico de orden desconocido.

En el campo del análisis de señales biomédicas, también podemos encontrar importantes contribuciones. En [Hunstein et al., 1986] se realiza un análisis automático del EEG (electroencefalograma) del sueño. En este artículo se obtiene como resultado más significativo que la utilización de un parámetro continuo para representar el EEG es más ventajoso que un modelo discreto. Desde este punto de vista, el objetivo de la máquina de estados borrosa es el de aprovechar estas ventajas atribuidas al parámetro continuo, manteniendo su estructura simbólica subyacente. Una de estas ventajas, en el caso del parámetro continuo del sueño, es la conservación de los cambios graduales que son de importancia en este fenómeno, como se pone de manifiesto en [Salzarulo et al, 1991]. En general, los parámetros continuos contienen información simbólica al tiempo que representan cambios y tendencias.

Como ejemplo de la clase de procesamiento que se puede realizar con este modelo se presenta en la figura 2.2 el FDA y el resultado del procesamiento sobre un fragmento de EEG que contiene dos grafoelementos de epilepsia formados por una "punta rápida" y una "punta lenta". Este estudio está recogido en [Moreno et al., 2001a]. El FDA fue diseñado para reconocer un patrón consistente en alta activación de símbolo de entrada con un mantenimiento más sostenido. La señal es procesada para obtener el valor absoluto de la pendiente normalizado y constituye la entrada del

sistema extendido y borrosificado. En el lado derecho de la figura 2.2 se muestran los niveles de activación de diferentes estados. El estado q_3 detecta la punta rápida, el q_4 detecta el mantenimiento en nivel bajo del símbolo de entrada y el q_5 determina el final de la morfología.

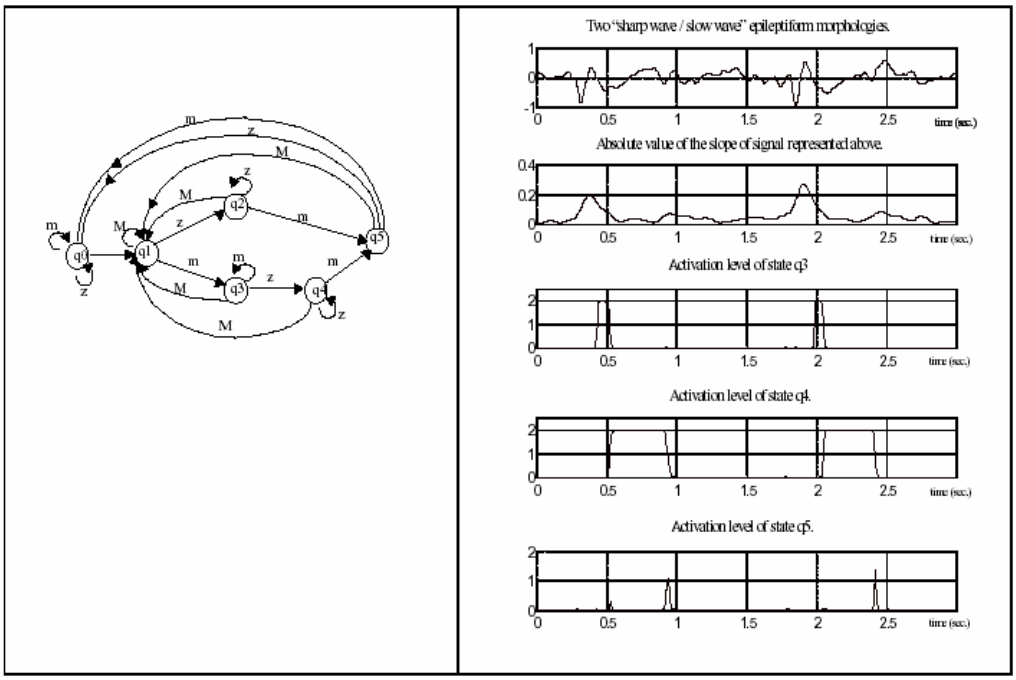


Figura 2.2. A la izquierda, autómata precursor utilizado para detectar el grafoelemento "punta rápida", "punta lenta" en el EEG. A la derecha y de arriba abajo: la señal, el valor absoluto de la pendiente y los niveles de activación de los estados q_3, q_4 y q_5. Figura extraída de [Moreno et al., 2001a].

Existen otras aproximaciones al análisis de las tendencias de una señal que hacen uso de la lógica borrosa. Tal es el caso del sistema propuesto por Steimann [Steimann, 1996], que también se basa en una máquina de estados.

2.3 Autómatas finitos.

Recordamos en esta sección el concepto de autómata finito no determinista y determinista, ya que son los precursores de la FFSM. El autómata no determinista se introduce aquí por completitud de esta introducción de conceptos previos, ya que en el resto del trabajo se usará fundamentalmente el concepto de autómata determinista.

2.3.1 Autómata finito no determinista.

Un autómata finito no determinista (AFN) es un modelo matemático formado por:

- Un conjunto de estados S.
- Un conjunto de símbolos de entrada Σ, denominado alfabeto de símbolos de entrada.
- Una función de transición δ que transforma pares estado-símbolo en conjuntos de estados.
- Un estado s_0 que se considera el estado inicial.
- Un conjunto de estados *F* considerados como estados de aceptación o finales.

Un AFN se puede representar mediante un grafo dirigido etiquetado, denominado grafo de transiciones, en el que los nodos son los estados y las conexiones dirigidas y etiquetadas representan a la función de transición. La figura 2.3 muestra en su parte superior el grafo de un AFN.

El AFN acepta una cadena de entrada *x* si y sólo si hay algún camino en el grafo de transiciones desde el estado de inicio a algún estado de aceptación de forma que las etiquetas de las conexiones a lo largo de dicho camino deletreen a la cadena de entrada *x*. El lenguaje definido por un AFN es el conjunto de cadenas de entrada que acepta.

2.3.2 Autómata finito determinista.

Un autómata finito determinista (AFD) es un caso especial de un autómata finito no determinista en el cual para cada estado *s* y cada símbolo de entrada *a*, hay a lo sumo una conexión etiquetada como *a* sale de *s*. La figura 2.3 en su parte inferior muestra el grafo de un AFD.

Dado un AFN se puede obtener un AFD que acepte el mismo lenguaje. Para ello se emplea un algoritmo que se denomina “construcción de subconjuntos” [Aho et al., 1986].

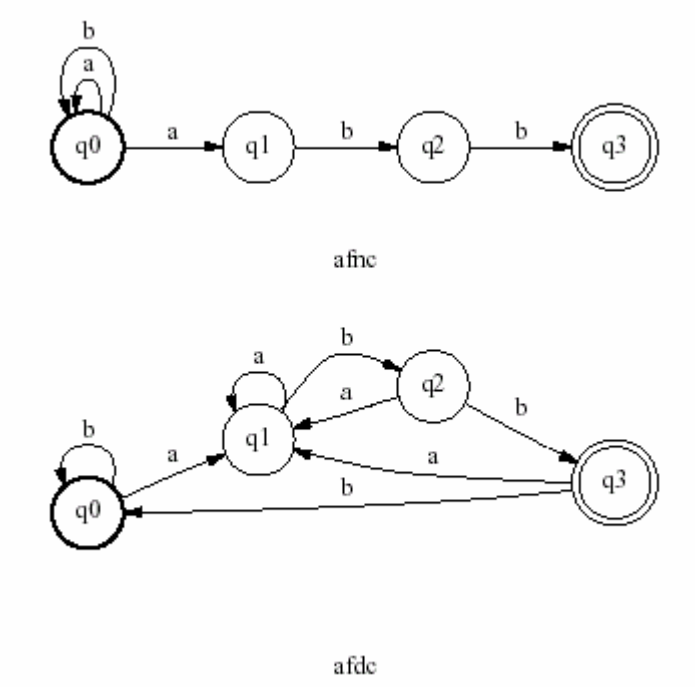


Figura 2.3. Grafos de dos clases de autómatas. En la parte superior se muestra un autómata no determinista (AFN) y en la parte inferior se muestra un autómata determinista (AFD).

2.4 El autómata borroso clásico.

En 1969 Wee y Fu introdujeron el concepto de autómata borroso [Wee y Fu, 1969]. Un autómata finito borroso es un conjunto formado por cinco elementos *(I, V, Q, f, g)* donde:

- *I* es un conjunto no vacío y finito de objetos que corresponde a los símbolos de entrada.

- *V* es un conjunto no vacío y finito de objetos que corresponde a los símbolos de salida.

- *Q* es un conjunto finito de objetos que corresponde a los estados internos.

- f es la función de pertenencia de un conjunto borroso definido sobre $Q \times I \times Q$, esto es, $f : Q \times I \times Q \to [0,1]$.

- g es la función de pertenencia de un conjunto borroso definido sobre $V \times I \times Q$, esto es, $g : V \times I \times Q \to [0,1]$.

Las transiciones vienen dadas por la función f que es llamada *función de transición borrosa*, mientras que las salidas del sistema vienen dadas por g que es llamada *función de salida borrosa*.

Sea $I = \{i_1, i_2, ..., i_p\}$, $V = \{v_1, v_2, ..., v_r\}$ y $Q = \{q_1, q_2, ..., q_n\}$, entonces $f_A(q_l, i_j, q_m)$ es el grado de transición desde el estado q_l hasta el estado q_m cuando la entrada es i_j. Por lo tanto:

$$f_A(q_l, i_j, q_m) = f\{q(k) = q_l, i(k) = i_j, q(k+1) = q_m\} \tag{2.4}$$

Como se aprecia, para cada símbolo de entrada i_j y para cada par de estados internos del autómata se tiene un grado de transición. Por lo tanto, para cada símbolo de entrada se podrá definir una matriz A que incluya el grado de transición de todos los posibles cambios de estado. Análogamente para cada símbolo de entrada tendremos un grado de transición de salida, para cada estado interno q_i y para cada estado de salida v_j que es $g_A(v_j, a, q_i)$. Esto define para cada símbolo de entrada una matriz de salida A_0.

Si consideramos que estas son matrices borrosas que representan relaciones borrosas binarias, una secuencia de entradas producirá una relación borrosa de grado n. Sea una secuencia de símbolos de entrada $I_p(k) = \{i_1(k), i_2(k+1), ..., i_p(k+p-1)\}$. La matriz de transición borrosa para esta secuencia de símbolos de entrada la obtendremos a partir de una ley de composición de relaciones borrosas. Esta ley de composición de relaciones borrosas para dos relaciones borrosas es:

$$f_{A \circ A}(x, \{i_1, i_2\}, y) = \vee_v (f_A(x, i_1, y) \wedge f_A(x, i_2, y)) \tag{2.5}$$

Luego para el caso general de una secuencia de símbolos de entrada:

$$f_A(q_l, I_j(k), q_m) = \vee_{q_0, q_p, \ldots, q_s} (\wedge\{f_A(q_1, i_1, q_0), f_A(q_0, i_2, q_p), \ldots, f_A(q_s, i_j, q_m)\}) \quad (2.6)$$

Cuando el operador $\vee$ es sustituido por la función máximo y $\wedge$ es sustituido por la función mínimo tenemos el autómata borroso pesimista. Si se utiliza al contrario tendremos la versión optimista del autómata.

En [Wee y Fu, 1969] se utiliza al autómata borroso como un modelo de aprendizaje. Una de las aplicaciones presentadas se refiere al problema de la clasificación. En la figura 2.4 se muestra el esquema del clasificador con aprendizaje en línea propuesto por Wee y Fu.

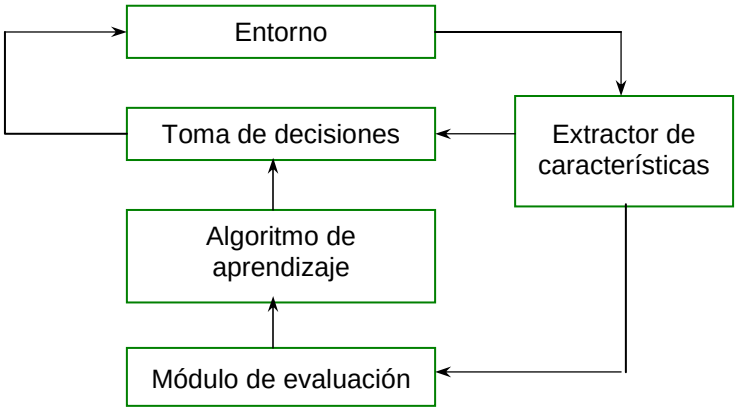


Figura 2.4. Esquema de clasificador con aprendizaje basado en el autómata borroso de Wee.

El *entorno desconocido* recibe como entrada las decisiones del módulo de toma de decisiones y produce una salida. Esta salida es un conjunto de características que pasa al módulo de extracción de características donde existen diferentes conjuntos de funciones de discriminación.

En el módulo de toma de decisiones se debe elegir el conjunto de funciones de discriminación que sea más adecuado para mejorar la bondad calculada en el módulo de evaluación. En la aplicación concreta de la clasificación de patrones el extractor de características posee un número N de diferentes conjuntos con R funciones de discriminación $g_i(X)$ cada uno. El módulo de evaluación calcula la bondad del conjunto de funciones de discriminación seleccionado en la toma de decisiones en cada paso el aprendizaje.

La elección del conjunto de funciones discriminantes se realiza tomando como base al autómata borroso donde los estados de entrada son las *R* clases posibles que pueden obtenerse con las funciones de discriminación y los *N* estados internos son los *N* conjuntos de funciones de discriminación. Para determinar en cada momento qué conjunto de funciones de discriminación va a ser el próximo en ser utilizado (q_j) se necesita la clase actual que actúa como estado de entrada (i_l) y el conjunto actual (q_i). Aquel conjunto que obtenga el valor máximo para

$$f_{ij}^{l}(k) = f\{q(k) = q_l, i(k) = i_l, q(k+1) = q_j\} \tag{2.7}$$

será el próximo conjunto de funciones de discriminación en ser utilizado. Por lo tanto, la función del módulo de aprendizaje debe ser la de adaptar las matrices de transición borrosa para mejorar la bondad de las evaluaciones en cada paso.

2.5 El autómata borroso a partir de un relieve borroso (Virant y Zimic).

Virant y Zimic introducen en 1995 [Virant y Zimic, 1995] una variante más de un autómata borroso. Su objetivo no es otro que llevar el proceso de borrosificación no sólo a las transiciones sino a la definición de las propias entradas, estados y salidas del autómata. Gracias a esto pretenden integrar la utilización de autómatas borrosos en las técnicas estándar de construcción de sistemas de inferencia borrosos aprovechándose de conceptos como el de variable lingüística, o los métodos de borrosificación, inferencia borrosa y desborrosificación.

De esta manera, se va a asociar un conjunto borroso *I* a las entradas, un conjunto borroso *O* a las salidas y un conjunto borroso para los estados *S*, de modo que el autómata queda especificado mediante:

$$\overline{A} = (I, S, O, R, \delta, \lambda) \tag{2.8}$$

donde *R* es la llamada *función de relieve*. A continuación veremos el papel que juega cada uno de estos elementos. El símbolo de entrada es un conjunto borroso *I* sobre el

espacio X = [0,1]. El estado interno del autómata es otro conjunto borroso S sobre el intervalo Y = [0,1]. El producto $Y \times X$ nos da el conjunto de estados-entradas completo del autómata borroso. Sobre el espacio $Y \times X$ definimos el estado completo borroso como un conjunto borroso:

$$P = \{(y,x), \mu_P(y,x)\} \tag{2.9}$$

Veamos cómo se define $\mu_P(y,x)$. En primer lugar, se definen los estados del autómata como conjuntos borrosos con funciones de pertenencia $\mu_P(y,x)$, con forma piramidal con valor máximo de pertenencia en el pico $p_i = (y_{p_i}, x_{p_i})$, siendo $\mu_P(y_{p_i}, x_{p_i}) = \mu_P(p_i) = 1$.

En el plano $Y \times X$ es posible definir más de un estado pico, p_i, con $i = \{1,2,...,n\}$. Para tener en cuenta de forma conjunta todas las pirámides que definen estos estados, se define la función relieve:

$$R(\mu_{P_1}, \mu_{P_2}, ..., \mu_{P_n}) = max(\mu_{P_1}, \mu_{P_2}, ..., \mu_{P_{n1}}) \tag{2.10}$$

y entonces:

$$\mu_P(p) = R(\mu_{P_1}(p), \mu_{P_2}(p), ..., \mu_{P_n}(p)) \tag{2.11}$$

Usando el concepto de estado completo y el conjunto borroso P asociado, podemos redefinir el autómata borroso como:

$$\begin{aligned} &\overline{A} = (S, O, p_0, R, \delta, \lambda) \\ &P = \{p, \mu_P(p)\}, p \in Y \times X \\ &O = \{z, \mu_O(z)\}, z \in Z\} \end{aligned} \tag{2.12}$$

O es un conjunto borroso definido sobre el intervalo Z = [0,1], que representa la salida del autómata, y p_0 es el punto estado-entrada inicial (y_0,x_0) del autómata.

La función de transición entre estados viene dada a partir de la función de relieve. Se define la función de transición entre estados, δ, de forma que el nuevo estado *y'* viene determinado por:

$$y' = y + (dyp(\mu_P(y + dy, x) > \mu_P(y, x))) \tag{2.13}$$

donde *dy* es el incremento de *y*, y la función *p*( ) es una función de decisión que vale 0 si la condición señalada es falsa y 1 en caso contrario. La condición será verdadera cuando se produzca un aumento en la pertenencia del estado completo al introducir la variación *dy* señalada. Obsérvese que con la estrategia de cambio del estado completo elegida, la tendencia del autómata es la de alcanzar estados donde existan extremos de la función de pertenencia, como puntos estables.

Para finalizar la descripción del autómata, el conjunto borroso de salida es calculado en función del conjunto borroso de entrada y el estado interno a partir de reglas de inferencia borrosa del tipo:

$$\lambda : \text{si } I = I' \text{ y } S = S' \text{ entonces } O = O' \tag{2.14}$$

Los conjuntos borrosos *I'* y *S'* pueden obtenerse mediante la borrosificación de los valores *(x',y')* correspondientes al estado completo. De esta forma, los autores consiguen su objetivo de construir un autómata borroso donde tanto las entradas como los estados y las salidas estén borrosificados al contrario de lo que ocurre con el autómata borroso clásico de Wee.

2.6 Definición del modelo FFSM.

2.6.1 Formulación del modelo discreto de una máquina de estados.

Vamos a definir un modelo de máquina de estados que se puede entender como un autómata finito determinista (AFD) donde los estados pueden estar *activados* simultáneamente con diferentes niveles de activación.

El modelo discreto de una máquina de estados es un conjunto compuesto por ocho objetos:

$$\Phi = \{S, I, \sigma_M, \eta_L, s_0, T, \Omega_p, W\} \tag{2.15}$$

El conjunto $S = \{S_1, S_2, ..., S_M\}$ está formado por los denominados M aspectos o estados internos del sistema. El conjunto $I = \{I_1, I_2, ..., I_L\}$ es un conjunto cuyos elementos son las entradas externas del sistema. El conjunto σ_M se obtiene como el producto cartesiano de los M conjuntos $\sigma^i : \sigma_M = \sigma^1 \times \sigma^2 \times ... \times \sigma^M$. Cada σ^i es un conjunto totalmente ordenado compuesto por los *N(i)* niveles de activación posibles del estado $S_i : \sigma^i = \{\sigma_1^i, \sigma_2^i, ..., \sigma_{N(i)}^i\}$. Asignaremos los índices de modo que si $i > j$ entonces $\sigma_i^k > \sigma_j^K$.

Por su parte el conjunto η_L juega un papel análogo en lo que respecta a las entradas externas del sistema. η_L es el producto cartesiano de L conjuntos η^i, donde cada η^i se compone de los niveles de activación de la entrada externa I_i. De esta forma, $\eta_L = \eta^1 \times \eta^2 \times ... \times \eta^L$ y $\eta^i = \{\eta_1^i, \eta_2^i, ..., \eta_{R(i)}^i\}$. También existe una relación de orden total en cada η^i y los índices se asignan de forma que si $i > j$ entonces $\eta_i^k > \eta_j^K$.

El estado inicial del sistema es s_0 y queda definido por los niveles de activación de los estados en el instante inicial, por lo que $s_0 \in \sigma_M$. El nivel de activación global de los estados se obtiene mediante la aplicación $T : \sigma_M \rightarrow \Omega$. El conjunto $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_p\}$ incluye los niveles de activación globales del sistema.

La dinámica del sistema viene determinada por la aplicación $W : \eta_L \times \sigma_M \rightarrow \sigma_M$. Esta aplicación se usa para obtener un nuevo estado *F' = W(O,F)* a partir de una pareja *(O,F)*, siendo $O \in \sigma_M$ y $F \in \eta_L$ una descripción de los niveles de activación de los estados y de las entradas del sistema.

Muchos sistemas automáticos diseñados para la detección de patrones encajan en este modelo. Como un ejemplo, podemos citar el sistema de análisis de objetos y escenarios denominado VISOR [Leow y Miikkulaien, 1994], [Leow, 1994]. Está basado en la aplicación combinada de esquemas visuales y redes neuronales para la detección de objetos y el análisis de escenarios. En este sistema se asocian niveles de activación a la percepción de las componentes básicas del objeto así como a la activación global de otros subsistemas. El sistema VISOR evoluciona dinámicamente en base a los niveles de activación registrados en la etapa anterior.

El modelo discreto presentado se corresponde con un autómata finito determinista donde los estados del sistema son cada uno de los elementos de σ_M, los

símbolos de entrada son cada uno de los elementos de η_L, el estado inicial es s_0 y la función de transición es *W*.

2.6.2 Extensión del modelo discreto a un modelo borroso.

Se pretende encontrar un sistema con las mismas propiedades básicas que el modelo discreto que admita como entradas variables continuas. El primer paso es definir los elementos $\sigma_i, \eta_j^i, \Omega_i$ como conjuntos borrosos sobre $\Re$, cuyas funciones de pertenencia serán $\mu_{\sigma_j^i}(x), \mu_{\eta_j^i}(x), \mu_{\Omega_j^i}(x)$. Estas funciones de pertenencia deben cumplir las propiedades de normalidad y convexidad.

El modelo discreto incluye la aplicación $W : \eta_L \times \sigma_M \rightarrow \sigma_M$. Esta aplicación se puede expresar mediante un conjunto de reglas lógicas $R_{\sigma_M \times \eta_L, \sigma_M}$ de la forma: Si $E_1 = \eta_{i1}$ y $E_2 = \eta_{i2}$ y ... y $E_L = \eta_{iL}$ y $S_1 = s_{j1}$ y $S_2 = s_{j2}$ y ... y $S_M = s_{jM}$ entonces $S' = S_c$, con $E = (\eta_{i1}, \eta_{i2}, ..., \eta_{iL}) \in \eta_L$, $S = (s_{j1}, s_{j2}, ..., s_{jM}) \in \sigma_M$ y $S_c \in \sigma_M$.

Una vez hemos expresado el modelo discreto mediante conjuntos borrosos y una base de reglas lógicas, su traslación a un sistema borroso es conceptualmente directa. El sistema borroso que representa la extensión del modelo discreto es *F = {D,R,M}*, donde *D* es la definición parametrizada de los conjuntos borrosos que expresan los niveles de activación de las entradas y estados, $R \in R_{\sigma_M \times \eta_L, \sigma_M}$ es un conjunto de reglas que representa la función *W* y finalmente *M* es el conjunto de operadores lógicos y de implicación así como métodos de borrosificación utilizados en la definición del sistema borroso. De esta forma el sistema construido acepta como entradas externas variables continuas. Los valores reales son borrosificados en conjunto borroso y el estado del sistema queda representado por un producto cartesiano de conjuntos borrosos que son inferidos para cada entrada.

En la aplicación práctica del sistema hay que considerar la desborrosificación de los conjuntos borrosos de salida, esto es, obtener valores numéricos que representen el nivel de activación de los estados. La implementación realizada en este trabajo se basa en:

- El sistema tiene por entradas una variable externa y los niveles de activación de los estados. Como salida devuelve los nuevos niveles de activación de los estados.

- Por razones de índole práctico usaremos como base el modelo de sistema de inferencia borroso de tipo Sugeno.

- La base de reglas lógicas describe el comportamiento dinámico de la máquina de estados borrosa.

- Los niveles de activación de los estados son representados mediante conjuntos borrosos con sus funciones de pertenencia asociadas en los antecedentes de las reglas. Los niveles de activación de los estados son representados por constantes en los consecuentes tal y como el modelo de tipo Sugeno establece.

- La realimentación se realiza con las salidas numéricas que representan los niveles de activación de los estados. Por lo tanto, existe una etapa de borrosificación para los estados al igual que para la variable externa.

En la figura 2.5 se muestra un esquema de la estructura del sistema presentado en este trabajo. Como se puede observar, los elementos más importantes para implementar la máquina de estados finita borrosa son los sistemas de inferencia borrosos (SIBs), de donde se obtienen los niveles de activación de cada uno de los estados. Las entradas para estos sistemas son la señal externa y los niveles de activación de todos los estados de la máquina borrosa. Se usarán sistemas borrosos TSK, de modo que las salidas de los sistemas borrosos de inferencia son números concisos que pueden utilizarse como pesos para una combinación lineal.

Podemos encontrar muchas aplicaciones de los sistemas recurrentes basados en reglas. Gorrini y Bersini introdujeron en 1994 sistemas recurrentes basados en reglas borrosas junto con un algoritmo de aprendizaje basado en el gradiente para adaptar las funciones de pertenencia para modelar procesos dinámicos de alto orden [Gorrini y Bersini, 1994]. Independientemente de esta aproximación, en [Surmann et al., 1995]

utilizaron sistemas recurrentes basados en reglas borrosas de orden σ para modelar el comportamiento de un robot móvil autónomo.

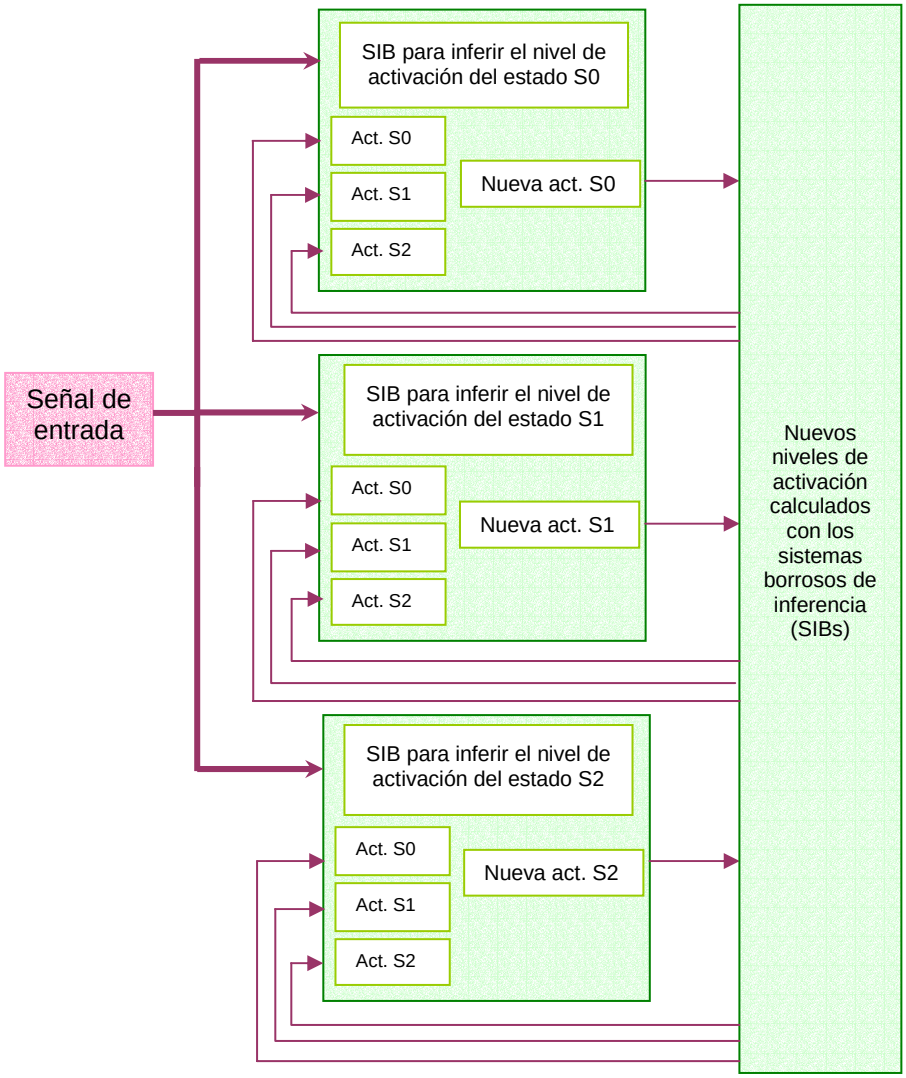


Figura 2.5. Diagrama de bloques representando una máquina de estados finita borrosa de tres estados implementada con sistemas borrosos.

En [Surmann y Maniadakis, 2001] se utilizan sistemas recurrentes basados en reglas borrosas (en inglés, *recurrent fuzzy rule-base systesms – R-FRBS*). Estos sistemas trabajan con reglas en las que una o más variables de estado *s* aparecen tanto en la parte de antecedentes como en la de consecuentes, del siguiente modo:

k: si $s(t)$ es $I_{0,k}$ y x_1 es $I_{1,k}$ y ... y x_n es $I_{n,k}$, entonces $s(t+1)$ es $O_{0,k}$ y y es $O_{1,k}$ (2.16)

donde $s(t)$ representa el estado del sistema en el instante t, x_1, ... x_n y y representan las variables de entrada y salida, y $I_{0,k}$, ..., $I_{n,k}$, $O_{0,k}$, $O_{1,k}$, sus respectivas etiquetas lingüísticas. Estos sistemas recurrentes se usan para modelar procesos de orden superior a uno, por ejemplo:

$$y(t) = f\,(\, y(t\text{-}1), y(t\text{-}2), ..., y(t\text{-}\sigma), x_1(t\text{-}1), x_1(\text{t-}2), ..., x_n(t\text{-}\sigma) \qquad (2.17)$$

En los procesos dinámicos de orden superior a uno, la salida depende de la entrada actual y de las entradas anteriores. Estos procesos son más difíciles de aproximar y controlar que los procesos de primer orden. Las variables de la estructura recurrente, que aparecen en la parte de entrada y salida del sistema, se usan como variables internas y constituyen una memoria a corto plazo (figura 2.6).

Si se presenta la misma entrada x a un R-FRBS, la salida y puede ser diferente dependiendo del estado s de las variables internas en ese momento. Los bucles de histéresis pueden ser modelados mediante el uso de variables borrosas ocultas (figura 2.6). Las diferentes curvas se seleccionan según sea el estado de las variables ocultas. Para el control del comportamiento de robots móviles autónomos [Surmann et al., 1995], las variables borrosas ocultas se utilizan para seleccionar distintos comportamientos y modelar su activación temporal. Las variables internas borrosas tienen su propia dinámica temporal dada por las reglas borrosas recurrentes.

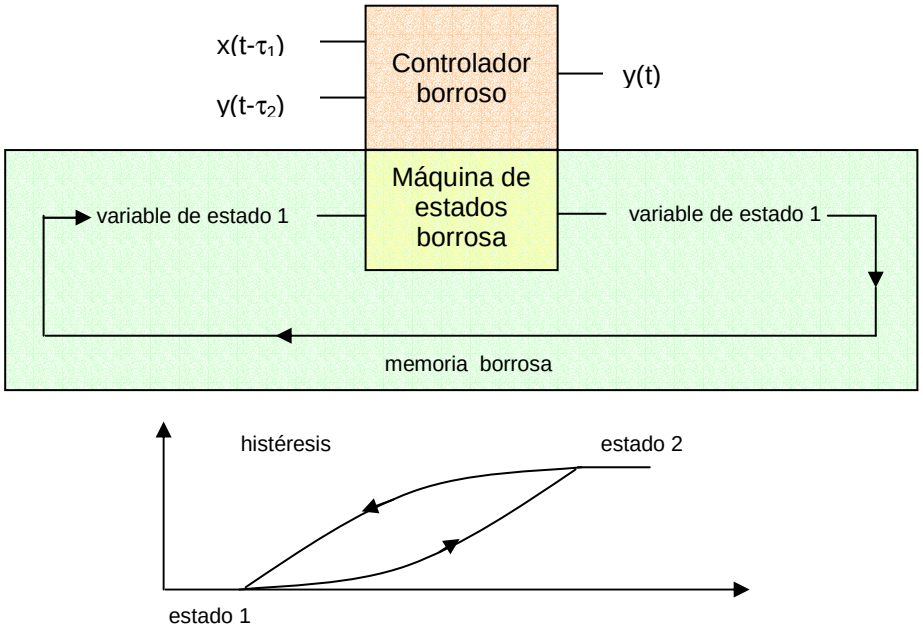


Figura 2.6. Memoria de la máquina de estados borrosa. Con la ayuda de las variables borrosas internas se pueden realizar bucles de histéresis.

2.7 Problemas en el diseño de máquinas finitas de estado borrosas.

El diseño de las FSM se puede abordar como la resolución de tres problemas diferentes:

- La determinación de la base de reglas (estructura simbólica de la FSM).

- La sintonización de los parámetros de las funciones de pertenencia asociadas a los antecedentes y a los consecuentes.

- La simplificación / reestructuración de las reglas (reducción de la base de reglas).

En este trabajo se realiza una investigación sobre la determinación de la base de reglas. En concreto, se trata de generar bases de reglas de un modo automático para abordar un problema de clasificación supervisada, de forma que el modelo subyacente al conjunto de entrenamiento pueda ser extraído.

El método que se ha utilizado para resolver el problema de la determinación de la base de reglas constituye una solución distinta a la planteada en [Estévez, 2001]. En este trabajo al que hacemos referencia se intenta especificar la estructura de la FFSM definiendo su comportamiento básico con un "modelo semilla" y utilizando un algoritmo ("modelo generativo") que sintetiza la máquina de estados borrosa que hereda este comportamiento básico.

El propósito del modelo generativo es producir, a partir de una determinada FDA denominada "autómata semilla", un producto de FDAs donde cada FDA represente el nivel de activación de un estado del FDA semilla. El modelo final debe incluir el comportamiento básico expresado a través del autómata semilla. La propiedad en que se basa este modelo es la de *influencia entre estados*, que a su vez dependerá de la *distancia* entre estados.

En primer lugar, se define el concepto de *alejamiento del estado s_j del estado s_i en un FDA*. Dado el FDA $A = \{X, \sum, \delta, s_0\}$, el alejamiento del estado s_j respecto al estado s_i es la cardinalidad de la cadena de símbolos p de menor longitud que cumple $\delta(s_i,p) = s_j$.

La *proximidad normalizada* entre estados s_i y s_j viene dada por:

$$nprox(s_i, s_j) = \frac{\hat{d} - D(s_i, s_j)}{\hat{d}} \tag{2.18}$$

donde $\hat{d}$ es el mayor alejamiento posible entre dos estados del autómata y $D(s_i,s_j)$ es el alejamiento del estado s_j respecto del estado s_i.

Finalmente, la influencia del estado s_i sobre el estado s_j se define como la función:

$$I(s_i, s_j) = \frac{1}{1 + e^{-\lambda ij\,(nprox(s_i, s_j) - c_{ij})}} \tag{2.19}$$

con $0 \leq c_{ij} \leq 1$ y $\lambda_{ij} > 0$.

En la figura 2.7 se puede apreciar el efecto de los parámetros λ_{ij} y c_{ij}. El aumento de c_{ij} hace necesario un menor alejamiento del estado s_j del s_i para mantener la influencia de s_i sobre s_j. Por otra parte, λ_{ij} regula la suavidad en la transición entre influencia nula y alta influencia.

Se comienza con un FDA $A = \{X, \sum, \delta, s_0\}$, donde $X = \{0,1\}$ es el alfabeto compuesto por dos símbolos (“0” para “no activado” y el “1” para “activado”). El objetivo es generalizar este modelo aumentando el número de los posibles niveles de activación para los estados. En términos del modelo discreto subyacente a la FFSM, la parte de los antecedentes de las reglas (donde se consideran todas las posibles combinaciones de los niveles de activación en la entrada y en los estados) es conocida. Por lo tanto, el modelo generativo debe establecer la parte de los consecuentes de estas reglas examinando la descripción de los estados internos y las entradas externas dadas por cada regla.

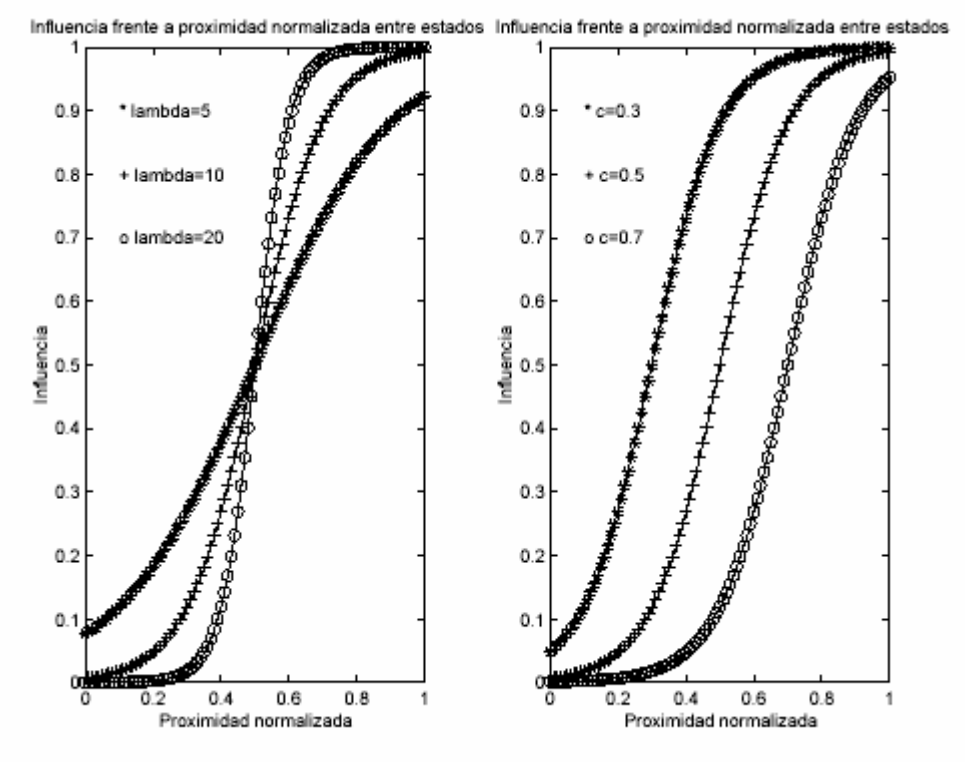


Figura 2.7. Función de influencia y efecto de los parámetros λ y c.

Cada estado del nuevo sistema tiene un número de niveles de activación $\sigma^k = \{\sigma_{k1}, \sigma_{k2}, ..., \sigma_{kN(k)}\}$. Definimos, como parte del modelo generativo, un conjunto de funciones monótonas crecientes (una para cada estado) $g_k : \sigma_k \rightarrow [0,1]$. Hacemos lo mismo para las variables de entrada con niveles de activación $\eta = \{\eta_1, \eta_2, ..., \eta_R\}$, definiendo las funciones monótonas crecientes $h : \eta \rightarrow [0,1]$. Ahora, se calcula la parte de consecuentes para la regla *i* en el modelo extendido. Si α_i es el nivel de activación de la variable de entrada, se calcula la variable auxiliar $s_i = h(\alpha_i)$. Si β_i^j es el nivel de activación del estado *j*, entonces calculamos la variable auxiliar $e_i^j = g_j(\beta_i^j)$. La siguiente fórmula establece el nivel de activación del estado *k* en la parte de consecuentes de la regla *i*:

$$nac_i(k) = nearest_{\sigma^k}(s_i \max_j(e_i^j I(k, \delta(j,1)) + (1 - s_i) \max_j(e_i^j I(k, \delta(j,0))) \quad (2.20)$$

con

$$nearest_{\sigma^k}(x) = \{\sigma_l^k \in \sigma^k / \left| g_k(\sigma_l^k) - x \right| = \min_{\sigma_k} \{\left| g_k(\sigma_l^k) - x \right|\}\} \quad (2.21)$$

Esta expresión está basada en la influencia entre dos estados. Cuando un estado ve aumentado su nivel de activación por efecto de un nivel de activación particular en la entrada, también afecta a sus vecinos. En este caso sólo se considera una variable de entrada que puede estar en diferentes niveles de activación. Las transiciones entre estados dependen del nivel de activación de la variable de entrada. En la expresión se considera que la influencia entre estados se ejerce por niveles de activación altos y bajos en la variable de entrada, ya que ambas situaciones provocan transiciones diferentes en el autómata básico. El nivel de activación asignado al estado *k* se calcula sólo en base a otros dos estados: un estado de alta activación en la regla y que tiende a acercarse a *k* si la entrada tiene una alta activación y otro estado también de alta activación relativa en la regla y que tiende a acercarse a *k* cuando la entrada tiene un nivel de activación bajo. En la figura 2.8 se muestra un esquema del proceso global.

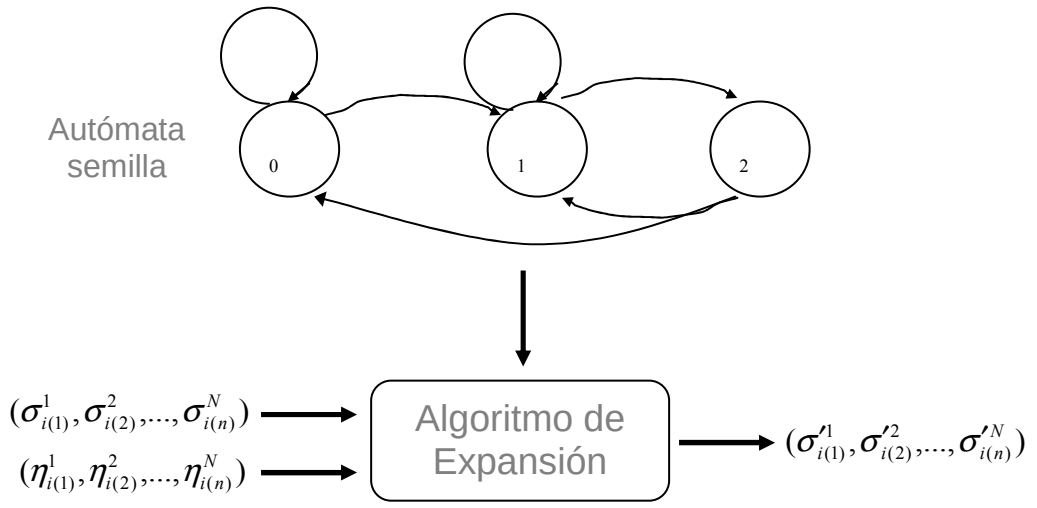


Figura 2.8. Representación esquemática del proceso descrito. El algoritmo de expansión determina las nuevas activaciones de los estados a partir de las activaciones de los estados en el instante anterior y de las entradas externas. Para ello, utiliza información proporcionada por el autómata semilla.

Tras esta explicación queda claro, que en [Estévez, 2001] el conocimiento experto se explicitaba directamente en el modelo semilla y en los parámetros del algoritmo de expansión.

Evidentemente, la determinación de la base de reglas no es el único problema que se plantea en el diseño de FFSM. Otra de las principales dificultades prácticas encontradas es la obtención de valores adecuados para los parámetros involucrados en la definición de las funciones de pertenencia. La especificación de dichas funciones de pertenencia es un problema que puede ser abordado desde múltiples enfoques.

Una primera clase de estrategias se basa en restricciones generales aplicadas a los interfaces de los sistemas borrosos: por ejemplo, un criterio de distinguibilidad entre las funciones de pertenencia [Oliveira, 1995] u otros criterios de carácter heurístico [Chow et al., 1999a], [Chow et al., 1999b].

Una alternativa muy extendida en la práctica es basarse en el conocimiento de un experto, o en la propia estructura sugerida por la estadística de los datos para realizar la sintonización adecuada de los parámetros involucrados en las funciones de pertenencia ([Turksen, 1991], [Chow et al., 1999b]).

En [Moreno et al., 2000] se propone como base para un método de sintonización por grado de cumplimiento de un conjunto de restricciones, la optimización de cierta función de coste cuyo valor es penalizado ante el no cumplimiento de las restricciones. A continuación, se describe brevemente los fundamentos de este método.

El modelo borroso puede verse en general como una función $f(\vec{x},\vec{a})$: $R^n \rightarrow R^m$, donde el vector $\vec{x}$ se corresponde con la medida de los niveles de activación de los estados, mientras que el vector $\vec{a}$ integra el conjunto de parámetros asociados a las funciones de pertenencia del sistema. Las restricciones sobre el comportamiento del sistema pueden ser definidas de múltiples formas. En [Moreno et al., 2000], se toma como referencia un estado del modelo S_i y se le asocian un conjunto de estados relacionados $\vec{S}_i = \{S_{i(1)}, S_{i(2)}, ..., S_{i(K(i))}\}$. Estos estados relacionados son aquellos que participarán en las restricciones que tomen como base el comportamiento del estado S_i. Las restricciones definidas establecen zonas prohibidas en el espacio de salida o de rango de la función f. Si $\vec{v}_0 = f(\vec{x},\vec{a})$ representa el vector con los valores obtenidos de la función f para un $\vec{x}$ de entrada, podemos definir un tipo de restricción como:

Si $\vec{v}_0(i) < \alpha$ entonces es necesario que exista $S_{i(j)} \in \vec{S}_i \,/\, k(j)(\vec{v}_0(S_{i(j)} - \beta(j)) > 0$, con $k(j) \in \{-1, +1\}$ (2.22)

Esta restricción puede modelar la siguiente instrucción sobre el comportamiento del sistema: si un estado particular tiene un bajo nivel de activación, entonces deben existir otros que son complementarios con un alto nivel de aplicación.

Utilizando la función $h : R^m \rightarrow \{0, 1\}$, que toma el valor 0 cuando su argumento no cumple la restricción impuesta y 1 cuando la cumple, y empleando el volumen de la

región en el espacio de las entradas del sistema V_I donde la restricción se cumple, se puede definir un índice a minimizar:

$$J = \left\| \int_I h(f(\vec{x}, \vec{a}))d\vec{x} - V_I \right\|, \qquad \vec{a} > \vec{0} \tag{2.23}$$

Este es un problema de optimización complejo ya que la función a optimizar presenta discontinuidades. Debido a las características propias del problema, existen zonas del espacio de parámetros de la función de coste con gradiente cercano a cero y además muchos mínimos locales. Además, el cálculo de la función a optimizar puede ser computacionalmente costoso ya que se trata de una integral sobre un espacio multidimensional. La función a evaluar es el modelo borroso. El número de evaluaciones crece con la dimensionalidad (número de estados) del modelo. Aunque este problema se puede paliar en cierto grado mediante la aplicación de técnicas heurísticas, la dependencia con el número de estados es inherente al método por lo que no puede ser eliminada.

En [Estévez, 2001] se aborda la definición de las funciones de pertenencia a partir de la inyección de conocimiento experto explicitado mediante restricciones sobre el comportamiento del sistema. La metodología utilizada se basa en la aplicación de un modelo generativo junto con una técnica de recombinación de antecedentes. Esta técnica permite incorporar el conocimiento de las restricciones del sistema a las funciones de pertenencia del sistema borroso. La idea inicial es usar un modelo generador sobre un sistema básico para modificar el espacio de búsqueda. Con esta estrategia se modifica el conjunto de parámetros sobre los que realizar variaciones encaminadas a mejorar el cumplimiento de restricciones impuestas, de modo que a costa de aumentar el número de valoraciones lingüísticas, el nuevo conjunto de parámetros tenga un mejor comportamiento desde el punto de vista de los algoritmos de búsqueda. Este aumento del número de funciones de pertenencia asociadas a los antecedentes de las reglas se trata de compensar en la segunda parte del método, mediante un proceso de recombinación de las nuevas valoraciones que permite llegar a un modelo reducido.

En el presente trabajo se crea un diccionario de posibles valores para los parámetros de las funciones de pertenencia y se utiliza un algoritmo genético en el espacio definido para estos parámetros. Este método soslaya el problema de la

discontinuidad, ya que no se utilizan métodos basados en el gradiente como métodos de búsqueda.

El tercer problema que aparece en el diseño de una FFSM es el de la simplificación/ reestructuración de las reglas (reducción de la base de reglas). El uso de un gran número de reglas asegura el mejor recubrimiento del espacio de entrada. Sin embargo, la capacidad de generalización del sistema es pequeña cuando se establece una partición excesivamente fina. Es decir, la respuesta del sistema ante situaciones que no se habían considerado explícitamente en el diseño puede ser muy diferente a lo esperado. Además, un gran número de reglas podría suponer un coste computacional elevado, la complicación del proceso de ajuste y mayor dificultad en la interpretación del sistema por parte del especialista.

Siguiendo a [Yen y Wang, 1999], una regla puede ser *redundante* o *poco importante*. Será redundante cuando su activación pueda obtenerse aproximadamente como el resultado de la combinación lineal de activaciones de otras reglas, para cualquier valor del espacio de entrada. Esto incluye el caso de varias reglas con la misma activación para el espacio de entrada completo. Por otra parte, se considerará poco importante cuando su activación sea cercana a cero para cualquier punto del espacio de entrada. Estas definiciones de redundancia e importancia pequeña, establecidas así en [Yen y Wang, 1999], tienen algunas desventajas. Por ejemplo, no se ha considerado la probabilidad de ocurrencia de los puntos del espacio de entrada. Esto es, si una regla tiene un valor de activación destacable sólo en una región del espacio de entrada altamente improbable también se podría considerar como poco importante para determinadas aplicaciones.

Las definiciones utilizadas en el trabajo anteriormente referenciado pueden ser poco operativas cuando no es posible un muestreo suficientemente detallado del espacio de entradas del sistema borroso. Este problema se acentúa cuando la dimensionalidad de dicho espacio crece. En estos casos puede ser más práctica la consideración de redundancia o poca importancia en base a la propia estructura conocida de la regla más que por cómo es su activación en el espacio de las entradas. La discriminación de reglas redundantes en base al análisis de la propia estructura de las reglas y considerando la distribución de probabilidad de activación de diferentes zonas del espacio de entrada es el método seguido en [Estévez, 2001].

El objetivo de los métodos de reducción de reglas es eliminar *M* reglas, dado un sistema de inferencia borroso con *N* reglas. Se trata de encontrar el subconjunto de *M* reglas tal que una medida de error entre el sistema original y el sistema reducido alcance un valor mínimo.

Una vez establecida una medida de error, existen diferentes métodos para alcanzar el objetivo propuesto. Uno de estos métodos consiste en ir midiendo el error tras eliminar cada uno de los posibles subconjuntos de reglas del sistema original. Este método en general es inabordable, ya que el número de combinaciones de reglas seleccionadas en un sistema con un número elevado de reglas es enorme.

En [Estévez, 2001] se aborda la selección de reglas bajo la premisa de que se dispone de un modelo básico aproximado de la dinámica del sistema recurrente que se pretende reducir. Esta aproximación es diferente a la presentada en [Yen y Wang, 1999], ya que allí se considera la reducción del sistema en base a la matriz de activación de las reglas. Cada fila de la matriz de activación se corresponde con un dato de entrada al sistema. Cada columna se corresponde con una regla. De esta manera, el elemento (i, j) de la matriz consiste en la evaluación de la parte de los antecedentes en la regla *j-ésima* cuando se tiene como entrada el dato *i-ésimo* de un conjunto de datos representativo del espacio de entradas del sistema. En [Yen y Wang, 1999], las características estructurales del sistema, como por ejemplo, la similaridad de reglas o la distribución estadística de las muestras de entrada, aparecen reflejadas en esta matriz de activación. En cambio, en [Estévez, 2001], se desarrolla un método de selección de reglas que no se basa en un conjunto de datos de entrenamiento sino más bien en un conocimiento a priori sobre el sistema y sobre las entradas del mismo. Por lo tanto, se consideran tres aspectos: un modelo probabilístico para las entradas externas al sistema, un modelo básico de la dinámica del sistema que junto con el modelo para la entrada externa servirá para una estimación de la distribución estadística de los valores de entrada del sistema, y una medida de similaridad entre las reglas.

Es importante destacar el carácter recurrente del sistema sobre el que se aplica la técnica. Esta característica no se da en el estudio realizado en [Yen y Wang, 1999], y es la motivación de incluir en el método de selección de reglas un modelo básico sobre la dinámica del sistema, ya que parte de las entradas del mismo provienen de las propias salidas del sistema. Con estas consideraciones la estrategia que se emplea se basa en dos ideas: conservar más reglas en aquellas zonas del espacio de entrada en las que existe

más probabilidad de que aparezca una muestra de entrada, y explotar la posible similaridad entre reglas de la base.

Como continuación del trabajo que se presenta en esta tesis, sería deseable utilizar algún método de reducción de reglas, pero teniendo en cuenta que las FFSM con las que trabajamos son sistemas recurrentes. Por lo tanto, lo más adecuado sería aplicar un método como el propuesto en [Estévez et al., 2002c], [Estévez, 2001], ya que este método considera la recurrencia del sistema a la hora de decidir la importancia relativa de las reglas.

Capítulo 3

Fundamentos del Reconocimiento de Patrones.

3.1 Introducción.

Este capítulo presenta un resumen de la teoría básica relacionada con el reconocimiento de patrones y la evaluación de clasificadores. Se divide en tres partes. En la primera se introduce el problema de la clasificación y el concepto de clasificador desde el punto de vista de la teoría bayesiana de la decisión. A continuación se presenta una taxonomía de los clasificadores, dividiéndolos en parámetricos y no paramétricos. En la segunda parte del capítulo se discute la estructura y diseño de un sistema de reconocimiento de patrones. Esta parte finaliza con la discusión de las técnicas clásicas de evaluación de clasificadores, enlazando con la tercera parte del capítulo relativo a la evaluación de pruebas diagnósticas mediante curvas ROC.

3.2 Definición del problema.

La clasificación es la asignación de objetos a clases predeterminadas, en base a sus características. Estas características pueden ser de tipo cuantitativo o cualitativo. Al conjunto de estas características se le suele denominar *patrón*. Por lo tanto, la dificultad principal del problema estriba en la capacidad discriminatoria entre clases del conjunto de características elegido.

Habitualmente, la naturaleza del problema es aleatoria, ante la imposibilidad de definir un modelo determinista de las clases. Para dar cuenta de esta aleatoriedad, las clases se caracterizan por medio de las densidades de probabilidad multivariantes de las características observadas. Este enfoque del problema de clasificación permite abordarlo desde la perspectiva de la Teoría Estadística de la Decisión [Sigut, 2001].

Teniendo en cuenta lo anterior, podemos definir más formalmente el problema de la clasificación de patrones de la siguiente manera: el objetivo de la clasificación de patrones es asignar una clase o categoría C_j contenida en un conjunto de clasificaciones alternativas $C = \{C_1, C_2, ..., C_L\}$ a un objeto basándose en su vector de características $\{x_1, x_2, ..., x_n\} \in \Re^n$. El problema de diseñar un clasificador se resume en encontrar un mapa D: $\Re^n \rightarrow C$ que sea óptimo en el sentido que maximice alguna medida de eficiencia deseada $\delta(D)$. La clasificación del objeto para el valor observado $X = x$, supone optar entre $L + 2$ decisiones. Estas decisiones pueden ser, o asignarlo a una de las L clases, o clasificarlo como "dudoso", posponiendo la decisión hasta tener mayor seguridad, o bien clasificarlo como *outlier*, esto es, que no pertenece a ninguna de las clases.

El algoritmo de aprendizaje normalmente comienza con un conjunto de ejemplos correctamente clasificados (ejemplos de entrenamiento) con el objetivo de encontrar un clasificador que asigne etiquetas de clases de modo que se minimice el error de clasificación en el espacio de características total. La eficiencia del clasificador se evalúa con un conjunto de ejemplos no vistos previamente (conjunto de test) para obtener una estimación del error de clasificación real.

Hay que aclarar que esta definición de clasificación se refiere a lo que se conoce como *clasificación supervisada*, en la que las clases están predefinidas y se dispone de patrones previamente clasificados (conjunto de entrenamiento) que son la base para el diseño del clasificador. En la *clasificación no supervisada* no se conocen las clases a

priori y, en esencia, se trata de descubrir agrupamientos de patrones (nuevas clases) de acuerdo con algún criterio establecido. En este trabajo nos centraremos exclusivamente en el entrenamiento supervisado de clasificadores.

Los sistemas de clasificación se pueden dividir en dos tipos dependiendo del modo en que se usan: los clasificadores que trabajan autónomamente y los que constituyen una herramienta de ayuda a la decisión para un usuario humano. En el caso de clasificadores autónomos, el objetivo básico del proceso de diseño es la eficiencia medida, por ejemplo, en el porcentaje de clasificaciones correctas. Otros criterios como la comprensibilidad, la robustez y la versatilidad son secundarios aunque también relevantes.

Si la densidad de probabilidad condicionada a las clases del vector de características es conocida, el problema de la clasificación de patrones se convierte en un problema de contraste de hipótesis estadístico que suele tratarse en el contexto de la Teoría Bayesiana de la Decisión. Si bien el supuesto de densidades de probabilidad conocidas es muy improbable en la práctica, resulta interesante su consideración, ya que proporciona un marco para tratar el problema y establece límites a las prestaciones de un clasificador.

3.3 Teoría Bayesiana de la Decisión.

La Teoría Bayesiana de la Decisión proporciona un marco adecuado para tratar el problema de la clasificación. El resultado fundamental en el que se basa el planteamiento posterior es la conocida Regla de Bayes, que se enuncia a continuación.

Regla de Bayes: La probabilidad condicional $p(a|b)$ se define como la probabilidad del suceso a si se ha dado el suceso b. Podemos expresar la probabilidad conjunta $p(a \cap b)$ de dos sucesos a y b (probabilidad de que se den el suceso a y el suceso b) como la probabilidad de que se verifique el suceso a por la probabilidad de que se verifique b condicionado a a: $p(a \cap b) = p(a)p(b|a)$, análogamente también se puede escribir como $p(a \cap b) = p(b)p(a|b)$. Si eliminamos $p(a \cap b)$ de las dos expresiones obtenemos la *Regla de Bayes*:

$$p(a \mid b) = \frac{p(a)\,p(b \mid a)}{p(b)} \tag{3.1}$$

Podemos aplicar esta regla al problema de determinar la pertenencia de un objeto a una de dos clases C_1 y C_2. Inicialmente podemos suponer conocidas las probabilidades de pertenencia del objeto a las clases con independencia del valor de la observación de sus propiedades. Con estas probabilidades a priori $P(C_1)$ y $P(C_2)$, podemos tomar como regla de decisión el elegir la clase de mayor probabilidad.

Alternativamente, podemos usar la información de las propiedades observadas del objeto para obtener una decisión más fiable a través de la regla de Bayes, junto con el criterio de máxima verosimilitud. Esto es, conocidos $p(x|C_1)$ y $p(x|C_2)$, se aplicaría la regla de Bayes para obtener:

$$p(C_i \mid x) = \frac{P(C_i)p(x \mid C_i)}{p(x)}, i = 1,2 \tag{3.2}$$

donde $p(x)$ es la probabilidad total (independientemente de las clases) de una observación de valores x y $P(C_i|x)$ es la probabilidad a posteriori de la clase C_i. Ahora la regla de decisión quedaría como sigue:

$$x \in C_1 \text{ si } P(C_1|x) > P(C_2|x) \tag{3.3}$$

$$x \in C_2 \text{ si } P(C_2|x) > P(C_1|x) \tag{3.4}$$

Una vez establecida la regla de decisión sería interesante disponer de algún criterio que diera cuenta de la eficiencia de la misma. En este sentido, el criterio más usado en el contexto de la clasificación de patrones es la probabilidad promedio de error definida como:

$$P(error) = \int P(error \mid x)p(x)dx \tag{3.5}$$

Con la regla de decisión anterior tenemos que $P(error|x)$ será igual a $P(C_1|x)$ si se decide C_2, o $P(C_2|\, x)$, si se decide C_1. Es muy fácil comprobar que de esta manera se minimiza $P(error)$ y, por lo tanto, se toma la mejor decisión posible. Esto supone que si tomamos este criterio de error como referencia, la aplicación de la regla de Bayes

establece un límite en la eficiencia alcanzable por un clasificador para unas características X dadas.

En la expresión anterior de la probabilidad promedio de error se ha asumido que todos los errores son igualmente costosos, es decir, que equivocarse al asignar a la clase 1 un objeto que pertenece a la clase 2, es equivalente a equivocarse en el otro sentido. En la práctica, a veces, resulta conveniente pesar estos errores de forma distinta. En estos casos, se plantea como criterio de bondad en la clasificación una función de riesgo más general que tendría a la probabilidad promedio de error como caso particular.

A partir de lo expuesto es posible introducir también el concepto de *función discriminante*, que resulta de gran utilidad como base para muchos de los métodos de diseño de clasificadores. Las funciones discriminantes se definen como un conjunto de funciones $D_i(x)$, cada una asociada a una clase C_i, de manera que el clasificador asigna el objeto a la clase i que verifica:

$$D_i(x) > D_j(x) \quad \forall\, j \neq I \tag{3.6}$$

Las funciones $D_i(x)$ son totalmente generales, no tienen que estar basadas necesariamente en argumentos probabilísticos. Desde el punto de vista geométrico, estas funciones dividen el espacio de las características en dos tipos de regiones. Por un lado, las regiones en las que todos los puntos verifican la condición anterior y, por tanto, pertenecen a la misma clase. Por otro lado, las regiones denominadas indeterminadas, en las que no se verifica la regla y por ello no es posible hacerles corresponder una única clase. Los límites de las regiones vendrán determinados por las fronteras o superficies de decisión entre cada dos clases i y j:

$$S_{ij} = D_i(x) - D_j(x) = 0 \tag{3.7}$$

En el caso de una decisión entre dos clases sólo hay una superficie de decisión $S_{ij} = 0$ y por tanto no quedan regiones indeterminadas: un punto pertenece a una clase u otra según quede a un lado u otro de la superficie de decisión.

La probabilidad a posteriori, antes introducida, puede ser considerada como una función discriminante, ya que:

$$D_1(x) = P(C_1|\, x) \tag{3.8}$$

$$D_2(x) = P(C_2|\, x) \tag{3.9}$$

La frontera de decisión quedaría entonces como:

$$S_{12} = 0 \tag{3.10}$$

$$P(C_1|\, x) - P(C_2|\, x) = 0 \tag{3.11}$$

$$\frac{P(C_1)p(x\,|\,C_1)}{p(x)} - \frac{P(C_2)p(x\,|\,C_2)}{p(x)} = 0 \tag{3.12}$$

$$P(C_1)P(x|C_1) - P(C_2)P(x|C_2) = 0 \tag{3.13}$$

expresión que se puede simplificar aún más en el caso de que las probabilidades a priori de las dos clases sean iguales.

La forma analítica de la superficie de decisión depende de las distribuciones asumidas en cada clase $P(x|C_1)$. El caso más común consiste en aproximar las distribuciones reales de cada clase por distribuciones normales multivariantes. En el caso de dimensión *n* esta distribución sería la siguiente:

$$p(x\,|\,C_i) = (2\pi)^{-n/2}\left|\Sigma_i\right|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right] \tag{3.14}$$

donde μ_i es el vector de medias y Σ_i es la matriz de covarianza.

Cada clase vendrá caracterizada en la distribución por su vector de medias y su matriz de covarianza. En el caso de que las matrices de covarianza sean iguales en ambas clases, la superficie de decisión resulta tener una dependencia lineal con las componentes del vector característico *x*, siendo este caso conocido como discriminante lineal. La frontera de discriminación vendría dada entonces por:

$$(\mu_2 - \mu_1)^T \Sigma^{-1} x + \frac{1}{2}(\mu_1{}^T \Sigma^{-1}\mu_1 - \mu_2{}^T \Sigma^{-1}\mu_2) - \log\frac{P(C_1)}{P(C_2)} = 0 \tag{3.15}$$

donde μ_1 y μ_2 son los vectores de medias de las dos clases y $\Sigma = \Sigma_1 = \Sigma_2$ es la matriz de covarianza.

En el caso en que estas matrices sean distintas, la dependencia con las componentes de *x* es cuadrática. Esta es la superficie de decisión más compleja a la que se puede llegar, asumiendo normalidad. Este tipo de fronteras se conoce como *discriminante cuadrático* y su ecuación es:

$$\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)+\frac{1}{2}\log\frac{|\Sigma_1|}{\Sigma_2}-\log\frac{P(C_1)}{P(C_2)}=0 \quad (3.16)$$

En la figura 3.1 se muestra un ejemplo de estas dos situaciones.

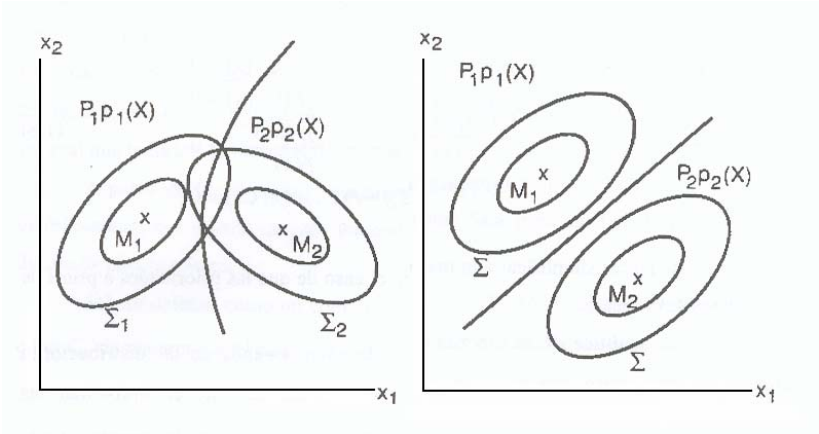


Figura 3.1. Superficies discriminantes para el caso cuadrático (izquierda) y el caso lineal (derecha), con distribuciones de clase normales (figura extraída de [Fukunaga, 1990].

3.4 Tipos de clasificadores.

En la discusión anterior se hizo referencia a dos tipos de discriminantes obtenidos a partir de la suposición de normalidad de las distribuciones de probabilidad entre clases: el discriminante lineal y el discriminante cuadrático.

Existen otros tipos de clasificadores, cada uno con sus especificaciones y problemática particular. Aunque, como ya se ha expuesto anteriormente, calculando las probabilidades a posteriori se puede conseguir una clasificación "ideal", en la práctica diversos factores complican mucho este cálculo y se hace preciso abordar el problema desde otra perspectiva. Lo más habitual es ir probando diferentes modelos de clasificadores hasta dar con el más adecuado. En la figura 3.2 se muestra un esquema con algunos de los clasificadores más comunes.

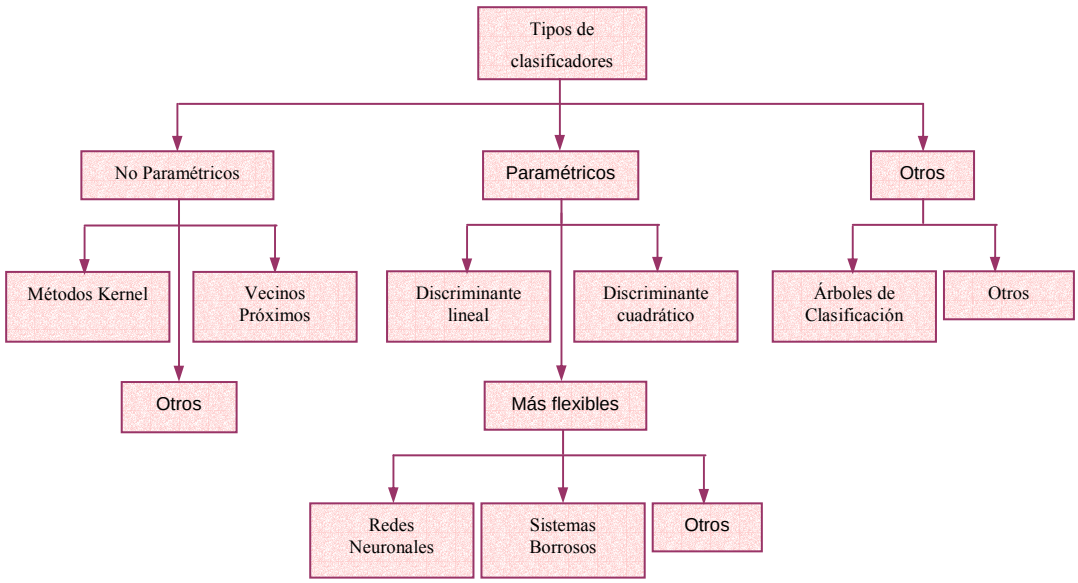


Figura 3.2. Tipos de clasificadores más comunes.

A continuación se hará una breve descripción de los diferentes tipos de clasificadores, insistiendo en los que se han utilizado en este trabajo.

3.4.1 Clasificadores paramétricos.

Los clasificadores paramétricos son aquellos cuyo diseño supone dar valores a una serie de parámetros. Aunque esta definición podría aplicarse a cualquier clasificador, dado que siempre hay algún parámetro que ajustar, se refiere en concreto a los casos en los que se asume una determinada estructura dependiente de cierto número de parámetros. Otra cuestión es si esta estructura paramétrica se asume para las densidades de probabilidad de las clases o para las probabilidades a posteriori. Ambas formas de plantear el problema se han mostrado efectivas en la práctica y se debe optar por la que se crea más conveniente para un problema dado.

3.4.1.1 Discriminantes lineal y cuadrático.

Los discriminantes lineal y cuadrático constituyen la opción más simple, pero también menos flexible. A pesar de esta falta de flexibilidad son muy usados en la práctica debido a la seria limitación que supone disponer de un número, normalmente escaso, de patrones de entrenamiento. Se definen, como se ha mencionado anteriormente,

independientemente de la suposición de normalidad. En el caso particular del discriminante lineal para dos clases, la definición es la siguiente:

$$y = w^T x + w_0 \tag{3.17}$$

es decir, una combinación lineal de las componentes de *x*. Dependiendo de la dirección del vector *w* obtendremos una mejor o peor separación de las clases. Se trata entonces de imponer ciertos criterios que permitan encontrar el valor óptimo de *w*. El umbral w_0 vendrá determinado por el criterio escogido.

Existen diferentes criterios para fijar *w*. El más clásico da lugar a lo que se conoce como el discriminante de Fisher [Fisher, 1936]. Así como para el discriminante lineal existen varios criterios que permiten fijar el valor de *w* [Fukunaga, 1990], para el discriminante cuadrático resulta mucho más complicado maximizar cualquier función de *w*, debido al gran número de parámetros implicados.

3.4.1.2 Redes neuronales.

En muchos problemas es más conveniente disponer de discriminantes paramétricos más flexibles que los anteriores. Entre las posibles opciones destacan por su gran popularidad las redes neuronales. Bajo este término se recogen una gran cantidad de sistemas computacionales que han experimentado un gran desarrollo en los últimos años. La característica común principal de estos sistemas es el estar inspirados en las redes neuronales que constituyen los sistemas nerviosos de los seres vivos más evolucionados.

Las redes neuronales implementan funciones complejas a base de interconectar elementos neuronales con funciones mucho más sencillas. La estructura de la neurona artificial depende en general de cada modelo, pero usualmente disponen de múltiples entradas con pesos asociados. Estos pesos son los parámetros a determinar, y representan la fuerza de las conexiones con otras neuronas. En los valores de los pesos queda almacenada la información del sistema. En las fases de entrenamiento o aprendizaje estos pesos se hacen variar de una forma determinada para lograr el funcionamiento deseado.

De especial interés en el campo del reconocimiento de patrones resulta la red conocida como perceptrón, creada por Frank Rosenblatt [Rosenblatt, 1962], inicialmente como modelo de retina artificial. Esencialmente es una red formada por una única capa, con múltiples entradas (cada una asociada a un elemento de visión, por ejemplo) y una salida binaria que indica si se ha detectado un patrón determinado a la entrada o no. En la figura 3.3 aparece la estructura de un perceptrón.

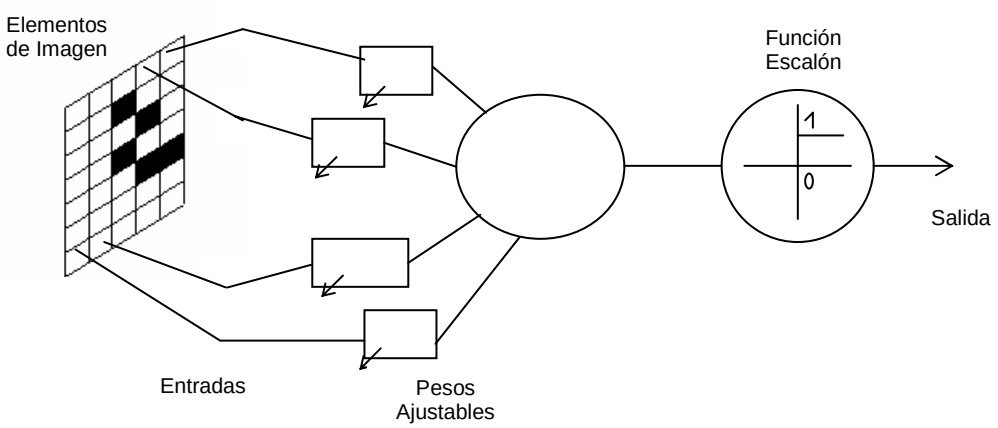


Figura 3.3. Esquema de la estructura del perceptrón.

El perceptrón implementa la función:

$$y = g\left(\sum_{i=0}^{n} w_i x_i\right) \tag{3.18}$$

siendo x_i las entradas y g la función de activación escalón o salto unitario. El peso w_0 se denomina *sesgo* o *umbral*, está conectado a una entrada 1 constante y tiene como misión aumentar la capacidad de representación del perceptrón, introduciendo una constante aditiva que da mayor flexibilidad, al poder controlar el umbral de disparo de la función escalón con independencia de las entradas actuales. Es, en definitiva, otra forma de discriminante lineal, aunque obtenido por procedimientos diferentes a los planteados en la sección anterior.

Los problemas de capacidad de representación del perceptrón individual se resuelven al usar estructuras con múltiples capas en alimentación hacia delante. La estructura típica contiene varias capas conectadas totalmente (la salida de una se propaga a todas las neuronas de la capa siguiente) como aparece en la figura 3.4.

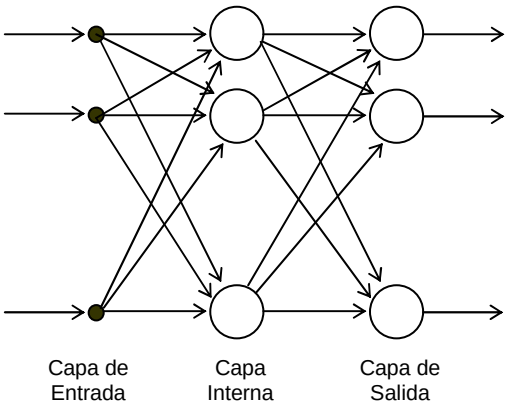


Figura 3.4. Estructura de un perceptrón multicapa con una capa interna.

A esta estructura se le denomina *perceptrón multicapa* (*MLP*) con una capa interna. En general, se compone de una capa de entrada con *n* unidades, *M* unidades en la cama interna y *m* unidades de salida. La relación entre las salidas y_k y las entradas x_i queda ahora de la siguiente manera:

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{n} w_{ji}^{(1)} x_i\right)\right) \tag{3.19}$$

donde $w_{ji}^{(1)}$ denota un peso en la capa de entrada, que va de la entrada *i* a la unidad *j* de la capa interna y $w_{kj}^{(2)}$ representa un peso en la capa interna que va de la unidad *j* a la salida y_k. Se ha distinguido también entre la función de activación para la capa interna *g* y la función de activación para la capa de salida $\tilde{g}$.

La capa de entrada sólo ejerce la función de proporcionar el *fan-out* necesario para que se propaguen las entradas a la red a todos los elementos de la siguiente capa y por tanto, no contiene unidades activas. La capa final de salida es la responsable de entregar los resultados finales. El resto de capas (internas) se denominan, tradicionalmente, ocultas, ya que no están en contacto directo con el exterior. Las unidades activas que componen la red son perceptrones con la modificación de tener a su salida, en lugar de la función salto, una función continua y acotada como la sigmoide o la tangente hiperbólica:

$$\text{Función sigmoide: } g(\alpha) = \frac{1}{1+\exp(-\alpha)} \tag{3.20}$$

$$\text{Función tangente hiperbólica: } g(\alpha) = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}} \tag{3.21}$$

De esta manera la función del perceptrón es diferenciable, así como la red en conjunto, y se pueden emplear métodos de gradiente en su entrenamiento.

Una generalización de la función sigmoide que es usada con frecuencia como función de activación en la capa de salida es la *función softmax* [Bridle, 1990]:

$$y_k = \frac{\exp(\alpha_k)}{\sum_{k'=1}^{L} \exp(\alpha_{k'})} \tag{3.22}$$

Con esta función de activación se normalizan las salidas de forma que estén en el rango (0,1) y sumen la unidad, lo cual es fundamental si éstas van a ser interpretadas como probabilidades.

El MLP de una sola capa interna con el número suficiente de unidades en ella es capaz de representar cualquier función continua con precisión arbitraria. Con dos capas internas se supera la restricción de continuidad. Contando con el número suficiente de unidades se aproxima, con la precisión requerida, cualquier función. La capacidad de representación no implica que esté resuelto el problema de determinar el conjunto adecuado de pesos para implementar la función deseada.

En las redes multicapa, el error se define normalmente como la suma de las diferencias al cuadrado entre las salidas que produce la red y las que debería producir, acumulada para cada patrón, debido a que ahora las salidas varían en un intervalo continuo. El objeto del entrenamiento es reducir dicho error para los patrones o parejas entrada/salida usados en el mismo.

3.4.1.3 Clasificadores basados en sistemas borrosos.

Siguiendo con la aproximación clásica de la clasificación por medio de funciones discriminantes, recordemos que dadas M clases, se requieren M funciones $d_1(\mathbf{x}), d_2(\mathbf{x}), ..., d_M(\mathbf{x})$ para decidir en qué clase está incluido el vector *x*. Decimos que el vector *x* está incluido en la clase *i* si $d_i(\mathbf{x}) > d_j(\mathbf{x}) \; \forall j \neq i$. Encontrar las funciones

$d_m(\mathbf{x})$ es el objetivo principal del proceso de diseño del clasificador. Las funciones de discriminación aíslan regiones en el espacio de características. Sin embargo, la forma de estas regiones no es siempre tan simple como en el caso de los discriminantes lineales, puede llegar a ser muy complicada, especialmente cuando la definición de las clases oculta relaciones entre las componentes de *x*. En los clasificadores basados en sistemas borrosos la función discriminante es más compleja y su forma está definida por las características del sistema borroso.

Los humanos poseen una importante habilidad para reconocer objetos a pesar de la presencia de información incompleta o con incertidumbre. Los sistemas borrosos son una herramienta capaz de manejar este tipo de ruido (imprecisión e incertidumbre de la información) que aparece en muchos problemas de clasificación. En concreto, los clasificadores implementados con sistemas basados en reglas borrosas (en inglés, *fuzzy rule-based classification system –FRBCS*) utilizan reglas borrosas para asignar etiquetas de clases a objetos [Bardossy y Duckstein, 1995], [Bezdek y Pal, 1992], [Chi et al., 1996] y tratan de hacer el proceso de clasificación transparente e interpretable.

Los componentes básicos de los FRBCS's son los siguientes:

1. Una base de conocimiento (en inglés, *Knowledge Base – KB*) compuesta por:

   - Una base de datos (en inglés, *data base-DB*), que contiene información sobre las variables de entrada.
   - Una base de reglas (en inglés, *rule base-RB*), que contiene las reglas borrosas de clasificación para el problema de clasificación específico.

   La granularidad y forma de la partición del espacio de entrada tendrá una gran influencia sobre la capacidad de clasificación del sistema.

2. Un método de razonamiento borroso (en inglés, *fuzzy reasoning method – FRM*) que clasifica patrones nuevos, es decir, determina que clase está asociada a ellos usando la información de la base de conocimiento.

Para implementar un FRBCS se debe empezar a partir de un conjunto de ejemplos preclasificados y se debe elegir el método para aprender o encontrar el

conjunto de reglas borrosas para el problema de clasificación específico y el método de razonamiento borroso que se usa para clasificar nuevos patrones. La estructura de un FRBCS y el proceso de diseño se muestran en la figura 3.5.

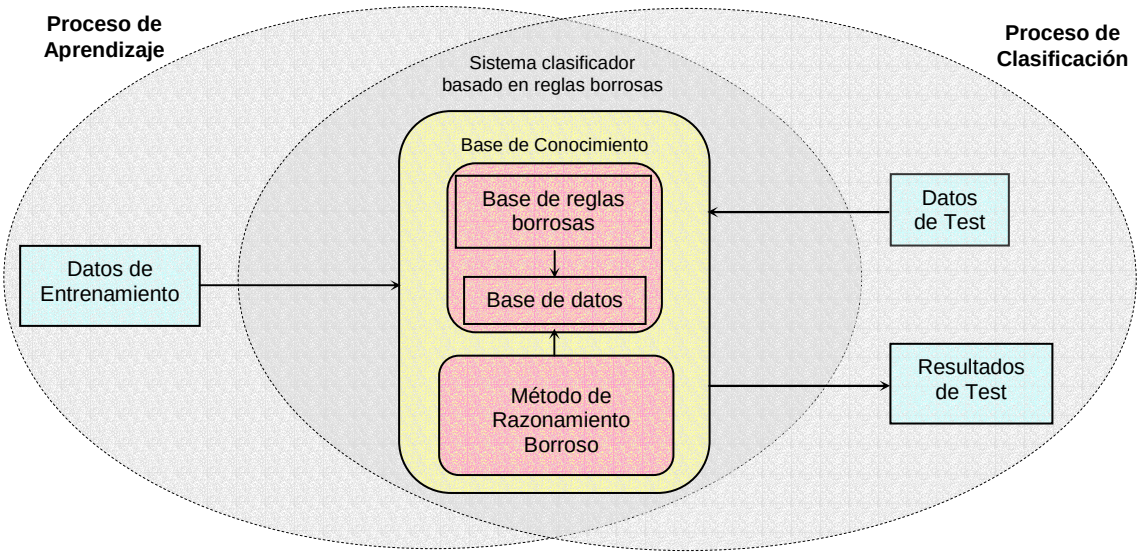


Figura 3.5. Estructura básica de un sistema clasificador basado en reglas borrosas.

Durante los últimos años se han propuesto varios métodos para generar reglas borrosas de clasificación a partir de pares de datos numéricos basándose en distintas técnicas [Chi et al., 1996], [Cordón et al., 1998a]. Además, se pueden encontrar en la literatura distintas propuestas sobre la selección del método de razonamiento borroso [Bardossy y Duckstein, 1995]. A continuación se analiza la composición de la base de conocimiento de un sistema clasificador basado en reglas borrosas y los métodos de razonamiento borroso más comunes empleados en estos clasificadores.

Dentro de la base de conocimiento, podemos encontrar tres tipos distintos de reglas borrosas de clasificación en la base de reglas:

1. Reglas borrosas con una clase en el consecuente [Abe y Thawonmas, 1997]. Este tipo de reglas tienen la siguiente estructura:

$$\text{Si } X_1 \text{ es } A_1 \text{ y .. y } X_n \text{ es } A_n \text{ entonces } Y \text{ es } C_i$$

donde $X_1,...,X_n$ son las características, $A_1,...,A_n$ son las etiquetas lingüísticas que particionan el universo del discurso y C_i $(i = 1,...,k)$ es la etiqueta de la clase que se asigna al objeto.

2. Reglas borrosas con una clase y un grado de certeza en el consecuente [Ishibuchi et al., 1992]:

$$\text{Si } X_1 \text{ es } A_1 \text{ y .. y } X_n \text{ es } A_n \text{ entonces } Y \text{ es } C_i \text{ con } r$$

donde r es el grado de certeza de que un objeto que encaje con el antecedente de la regla pertenezca a la clase C_i. Este grado de certeza se puede calcular por la relación S_i/S, que es el número de objetos S_i en el subespacio definido por el antecedente de la regla que pertenece a la clase C_i entre el número total de objetos S en esa región.

3. Reglas borrosas con un grado de certeza para todas las clases en el consecuente [Mandal et al., 1992]:

$$\text{Si } X_1 \text{ es } A_1 \text{ y .. y } X_n \text{ es } A_n \text{ entonces } (r_1,...,r_k)$$

donde r_i $(i = 1,...,k)$ son los grados de certeza de que el objeto de la región descrita por el antecedente de la regla pertenezca a las distintas clases C_i. Estos grados de certeza se pueden calcular con la misma relación expuesta en el punto anterior.

Los métodos de razonamiento borrosos infieren una etiqueta de clase para un objeto dado un vector de características y un conjunto de reglas borrosas del tipo *si-entonces*. Una de las ventajas del razonamiento borroso es que se puede obtener una clasificación incluso cuando sólo existe un encaje aproximado entre el vector de características y el antecedente de la regla.

El método más común de inferencia borrosa para los problemas de clasificación borrosa es el método del *encaje máximo* [Ishibuchi et al., 1992] que selecciona la etiqueta de clase de la regla cuyo antecedente encaje mejor con el vector de

características. Para reglas borrosas del tercer tipo (reglas borrosas con un grado de certeza para todas las clases en el consecuente) esta operación se realiza como se expone a continuación.

Supongamos que el sistema clasificador basado en reglas borrosas contiene las siguientes reglas borrosas: $R = \{R^1,..., R^n\}$. Para un patrón $E^t = (e_1^t,...,e_n^t)$ el método de razonamiento borroso selecciona la etiqueta de clase de la regla que mejor encaja con el vector de características con el máximo valor de certeza r_i. Para determinar qué regla es ésta, se utiliza el siguiente algoritmo:

- $R^i(E^t)$ es el *grado de activación* de la regla R^i. Normalmente $R^i(E^t)$ se obtiene al aplicar cualquier operador conjuntivo, como por ejemplo, una t-norma, a los grados de encaje de las cláusulas individuales (*"X_j es A^i_j"*):

$$R^i(E^t) = \mathrm{T}(\mu_{A_1^i}(e_1^t),...,\mu_{A_n^i}(e_n^t)) \tag{3.23}$$

- $d(R^i(E^t),r^i_j)$ denota el grado de asociación del patrón E^t con la clase C_j según la regla R^i. Este grado se obtiene aplicando un operador de combinación, como por ejemplo, el mínimo, el producto o la media aritmética, sobre $R^i(E^t)$ y r^k_j.

- El grado de asociación del patrón E^t con la clase C_j, $Y_j(E^t)$, se calcula para cada clase ($j = 1,...,k$):

$$Y_j(E^t) = \max_i d(R^i(E^t), r_j^i), i = 1,..., m \tag{3.24}$$

- Finalmente, el vector de características E^t se clasifica según la clase C_h que tiene el máximo grado de asociación.

$$Y_h = \max_j Y_j, j = 1,..., k \tag{3.25}$$

Uno de los inconvenientes del método de encaje máximo es que la regla ganadora toma toda la decisión y el método de razonamiento borroso no tiene en cuenta

las clasificaciones dadas por otras reglas activas. Existen métodos de razonamiento más sofisticados que añaden la información de múltiples reglas activas aumentando con ello la capacidad de generalización del clasificador [Bardossy y Duckstein, 1995], [Chi et al., 1996].

En este trabajo se trata de diseñar un sistema clasificador basado en reglas borrosas con una máquina finita de estados borrosa para clasificar series de datos. La naturaleza recurrente de la máquina (por ejemplo, la memoria introducida en el sistema al considerar el concepto de estados y activaciones de los estados) hace de este modelo un buen candidato para capturar información relevante concerniente a la dependencia entre datos sucesivos ordenados por una variable independiente. Por lo tanto, nuestro propósito es utilizar la máquina finita de estados borrosa como parte de una función discriminante usada para clasificar series de datos.

3.4.2 Clasificadores no paramétricos.

En este tipo de clasificadores no se asume ninguna forma paramétrica, ni de las densidades de probabilidad de las clases ni de las probabilidades a posteriori. Los procedimientos que se siguen están precisamente orientados a la estimación de las densidades de probabilidad a partir de los patrones disponibles, o bien, a la estimación directa de las probabilidades a posteriori.

En el primer caso, tenemos los métodos que se denominan, genéricamente, de *kernel,* porque se basan precisamente en funciones *kernel* (por ejemplo, gaussianas) que de forma local tratan de aproximar la forma de la densidad de probabilidad de la población de la que proceden los patrones [Hand, 1982].

Por otro lado, están los métodos que intentan estimar directamente las probabilidades a posteriori de las clases. Como ejemplo representativo de este tipo de clasificadores podemos citar *los vecinos próximos* [Dasarathy, 1991], que aproximan las probabilidades a posteriori para cada valor de x, asignándole la clase C_i más representada entre los k ejemplares de entrenamiento más cercanos.

Un resultado bastante significativo relacionado con los clasificadores de vecinos próximos es el que demuestra que el error asintótico (infinitos datos) es menor que dos veces el error de Bayes [Fukunaga, 1990]. Hay que tener en cuenta que este

procedimiento de clasificación no usa ninguna información acerca de la estructura probabilística del problema.

En ambos casos, y especialmente en el primero, se necesitan muchos patrones para que estos métodos resulten eficientes, por lo que en ocasiones, su aplicabilidad práctica puede llegar a ser un tanto limitada.

3.4.3 Otros clasificadores.

Dentro de la categoría de otros clasificadores, se incluyen aquellos que tratan el problema desde una perspectiva bastante diferente a los anteriores. Por ejemplo, métodos que consisten en particionar el espacio de características en regiones y asignar una clase a cada región.

Dentro de esta filosofía se encuentran los *árboles de clasificación* [Breiman et al., 1984], los cuales tienen la virtud de su fácil interpretabilidad, pero no resultan tan eficientes como discriminadores.

3.5 Componentes de un sistema de reconocimiento de patrones.

Los componentes básicos de un sistema típico de reconocimiento de patrones son los siguientes: interfaz de entrada, segmentación, extracción de características, clasificación y post-procesamiento. El proceso es el siguiente: un sensor convierte imágenes o sonidos u otras entradas físicas en una señal de datos, el módulo de segmentación aísla los objetos de interés del resto del entorno y de otros objetos, el extractor de características mide propiedades del objeto útiles para su clasificación, el módulo de clasificación es el encargado de asignar a los objetos una categoría utilizando la información de las características del mismo y, finalmente, el post-procesamiento se realiza para incluir otras consideraciones como, por ejemplo, los errores de coste, con el objetivo de decidir qué acciones son las más apropiadas.

En la figura 3.6 se muestra un diagrama de estos componentes. Aunque esta descripción refleja un flujo de datos descendente, algunos sistemas emplean realimentación para volver a niveles anteriores. En este apartado se presentan estos

componentes detalladamente, resaltando las dificultades más comunes que aparecen en su implementación.

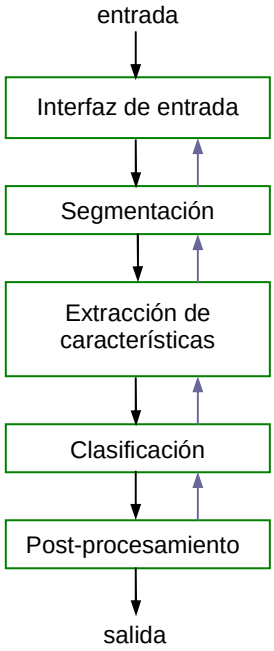


Figura 3.6. Diagrama de los componentes básicos de un sistema de reconocimiento de patrones.

Interfaz de entrada. La entrada del sistema de reconocimiento de patrones proviene normalmente de algún transductor (como por ejemplo una cámara o un array de micrófonos). Las dificultades que surgen en esta etapa inicial están relacionados con las características y limitaciones del transductor: su ancho de banda, resolución, sensitividad, distorsión, razón señal-ruido, latencia, etc.

Segmentación. Los patrones individuales deben ser segmentados. Los objetos de interés, por ejemplo en una imagen, pueden estar solapados, y el sistema debe ser capaz de realizar un pre-procesamiento de la imagen para determinar donde terminan unos objetos y empiezan otros y cuáles son los objetos que interesa aislar para una clasificación posterior. La segmentación es uno de los problemas más complicados en el reconocimiento de patrones.

Extracción de características. La barrera conceptual entre la extracción de características y la clasificación es de algún modo arbitraria. Un extractor ideal obtendría una representación tan buena que haría trivial el trabajo del clasificador, y

viceversa, un clasificador muy bueno no necesitaría la ayuda de un extractor de características muy sofisticado. La distinción entre estos dos conceptos es debida a motivos prácticos más que a razones teóricas.

La tarea de la extracción de características, al igual que la tarea de segmentación, requiere información del dominio ya que es un problema dependiente del mismo. Es muy importante seleccionar las características más adecuadas entre todas las posibles, tarea para la que es imprescindible manejar conocimiento del dominio. Además, es posible utilizar técnicas que, aunque no pueden sustituir el conocimiento del dominio, son capaces de hacer que los valores de las características sean menos sensibles al ruido.

Un aspecto central en cualquier problema de clasificación de patrones es obtener una buena representación en la que las relaciones estructurales entre los componentes sean simples y relevantes y sobre la cual se pueda expresar el modelo de los patrones. Los patrones se pueden representar como vectores de números reales, como una lista de atributos, etc. Se busca una representación donde los patrones correspondientes a una determinada naturaleza estén de algún modo cercanos entre sí y lejanos de los que tienen una naturaleza distinta. El modo en que se crea esta representación y cómo se cuantifica la cercanía y lejanía de los patrones determina en gran medida el éxito del clasificador de patrones. Es importante encontrar un número adecuado de características, lo más favorable es elegir un número pequeño para obtener regiones de decisión pequeñas y clasificadores fáciles de entrenar, así como características robustas (como ya se ha mencionado antes, características relativamente no sensibles a ruido u otros errores).

El objetivo tradicional del extractor de características es caracterizar un objeto para que pueda ser reconocido por medidas cuyos valores sean muy similares a las de otros objetos de la misma categoría y muy distintas a las de objetos de categorías diferentes. Estas características deben ser invariantes frente a transformaciones irrelevantes de la entrada. Por ejemplo, en la extracción de características de objetos de una imagen, suele ser deseable que las características extraídas sean invariantes a la traslación y a la rotación del objeto en esa imagen. Además, las características deben ser invariantes a la escala (el tamaño del objeto no debe ser relevante). Normalmente, las características que describen propiedades como la forma, el color y distintos tipos de texturas son invariantes a la traslación, rotación y la escala.

En este trabajo se pretende clasificar los núcleos celulares que aparecen en imágenes médicas de citologías en dos clases: células normales (sanas) y células anormales (patológicas). Las características que normalmente se extraen de los núcleos en este tipo de imágenes para realizar dicha clasificación son las siguientes: radio, varianza del radio, perímetro, área, características relacionadas con la irregularidad de la forma del núcleo (relación perímetro/área, suavidad de la forma, número de concavidades, simetría), tamaño, textura, etc [Street et al., 1993], [Wolberg et al., 1994].

Los sistemas clasificadores convencionales existentes que se emplean para clasificar en estos tipos de problemas suelen trabajar con la información combinada que se obtiene al extraer las características citadas. Nuestro sistema clasificador está diseñado para clasificar a partir de la información proporcionada solamente por la extracción de una característica: la textura de los núcleos. Se ha seleccionado esta característica por ser una de las características más invariantes con las que se puede trabajar en este tipo de imágenes. La medida de la textura del núcleo ofrece información de cómo se distribuye la cromatina en el mismo, que suele ser un buen indicador de la benignidad o malignidad de la célula. En este trabajo se extrae esta característica por medio de un procedimiento distinto a los habitualmente empleados. En el capítulo 8 comentaremos en detalle el modo en que normalmente se suelen extraer las características que se utilizan para realizar una clasificación en este tipo de imágenes y haremos una descripción completa del procedimiento de extracción de características seguido en el presente trabajo.

Clasificación. La tarea propia del clasificador es asignar una categoría a un objeto utilizando el vector de características de ese objeto proporcionado por el extractor de características. La clasificación perfecta es a menudo imposible, por lo tanto, existe una tarea más general que es determinar la probabilidad de cada una de las posibles categorías. La abstracción de la representación por medio de vectores de características de los datos de entrada posibilita el desarrollo de teorías de clasificación independientes del dominio.

El grado de dificultad del problema de clasificación depende de la variabilidad de los valores de las características para objetos de la misma categoría en relación con la diferencia de los valores de las características de objetos de distintas categorías. La variabilidad de los valores de las características de objetos de la misma categoría puede

ser debida a la complejidad y al ruido, considerando "ruido" en su concepto más general: cualquier propiedad del patrón recogido que no es debida al modelo real sino a alguna aleatoriedad del sensor utilizado. Los problemas no triviales de clasificación y reconocimiento de patrones incluyen ruido de algún modo, por lo tanto, es importante intentar diseñar un clasificador capaz de trabajar con esta variabilidad. Otro de los inconvenientes más comunes en la práctica es que no siempre es posible determinar los valores de todas las características para una entrada dada.

Post-procesamiento. Esta etapa utiliza la salida del clasificador para recomendar acciones que mejoren la eficiencia del sistema. Conceptualmente, la medida más simple de la eficiencia del clasificador es la razón del error de clasificación, esto es, el porcentaje de nuevos patrones a los que se han asignado categorías erróneas. Es muy común buscar la razón mínima del error de clasificación. Sin embargo, también se pueden recomendar acciones que traten de minimizar el coste total esperado, llamado *riesgo*. Incorporar conocimiento sobre los costes afectará la decisión del clasificador.

Este post-procesamiento debe ser capaz de explotar el *contexto* (información dependiente de la entrada distinta a la información recogida por el patrón) para mejorar la eficiencia del sistema, aunque este contexto puede llegar a ser altamente complejo y abstracto.

Mediante la utilización de varios clasificadores, cada uno operando en distintos aspectos de la entrada, se podría mejorar los resultados de la clasificación. La dificultad aparece en los casos en los que los clasificadores no están de acuerdo en la clasificación de un patrón dado. Es necesario determinar cuál es la mejor decisión (qué clasificador es más fiable).

3.6 Diseño de un sistema de reconocimiento de patrones.

El diseño de un clasificador de patrones es una tarea muy compleja. Es infrecuente encontrar problemas en los que se conozcan reglas específicas que puedan ser usadas en el diseño. En la gran mayoría de aplicaciones no se hacen suposiciones estructurales y toda la estructura del clasificador se aprende a partir de los datos disponibles. Es lo que

se conoce como reconocimiento de patrones estadístico. Es fundamental, por lo tanto, obtener la máxima información de estos datos, a menudo escasos. En este sentido cobran especial importancia las técnicas de análisis orientadas a averiguar, en la medida de lo posible, las distribuciones de probabilidad de los patrones.

El diseño de un clasificador o sistema de reconocimiento de patrones se lleva a cabo en una serie de etapas: recogida de datos, elección de características, elección del modelo, entrenamiento y evaluación. En la figura 3.7 se presenta el diagrama de este ciclo de diseño de los sistemas de reconocimiento de patrones. Los datos recogidos serán utilizados tanto en la etapa de entrenamiento como en la etapa de evaluación o test del sistema. Las características de estos datos influyen en la elección de características de discriminación adecuadas y en la elección de los modelos correspondientes a distintas categorías. El proceso de entrenamiento utiliza parte de los datos para determinar los parámetros del sistema. Tras el análisis de los resultados de la etapa de evaluación se puede determinar si es necesario repetir alguno de los pasos del proceso total para conseguir resultados más satisfactorios. A continuación se describen brevemente estas etapas y los problemas que frecuentemente surgen en su desarrollo.

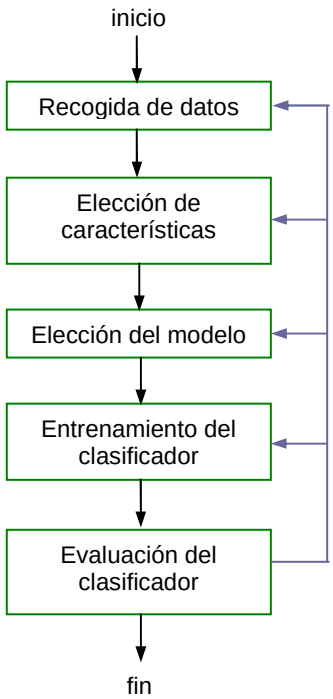


Figura 3.7. Diagrama de las etapas involucradas en el proceso de diseño de los sistemas de reconocimiento de patrones.

Recogida de datos. Es necesario recopilar datos del problema específico para entrenar el sistema y para evaluarlo. Esta recogida de datos puede llegar a ser muy

costosa. Es posible realizar un estudio preliminar del sistema utilizando un conjunto pequeño de "ejemplos" típicos, pero se deben recoger muchos más datos para asegurar una buena eficiencia del sistema. El principal problema de esta etapa es saber qué cantidad y qué tipo de datos forman un conjunto representativo de ejemplos para realizar adecuadamente el entrenamiento y la evaluación del sistema.

Elección de características. La elección de las características es un paso crítico y depende de las características del dominio del problema. El conocimiento previo del problema es indispensable en este proceso, pero no es siempre fácil incorporarlo para simplificar la obtención de características fáciles de extraer, invariantes a transformaciones irrelevantes e insensibles al ruido.

Elección del modelo. En esta parte del proceso se eligen los modelos para las distintas categorías. La dificultad está en la incapacidad de saber si un modelo seleccionado difiere significativamente del modelo real del que proceden los patrones que se desea clasificar, y por lo tanto, si es necesario buscar un modelo nuevo.

Entrenamiento. No existen métodos universales que resuelvan todos los problemas que surgen en la etapa de ajuste de los parámetros de un clasificador. Sin embargo, experimentalmente se ha comprobado que los métodos más efectivos son los que involucran aprendizaje de patrones ejemplos. Existen muchos métodos de entrenamiento para el clasificador. En el presente trabajo se desarrollan distintos algoritmos de aprendizaje basados en algoritmos genéticos.

Evaluación. La evaluación del sistema es importante para medir la eficiencia del sistema de reconocimiento de patrones y para determinar qué componentes se deben cambiar o mejorar para aumentar dicha eficiencia. Es muy común que un sistema que realiza una clasificación perfecta sobre las muestras de entrenamiento, no funcione con la misma exactitud sobre los nuevos patrones que componen el conjunto de test. Esta situación es conocida como "sobre-ajustamiento" (en inglés, *overfitting*) y ocurre cuando el sistema no ha aprendido de las muestras del conjunto de entrenamiento el patrón general que siguen sino particularidades que no comparten todos los patrones del mismo problema. Esta incapacidad de generalización sólo puede detectarse en la etapa

de evaluación y manifiesta la necesidad de repetir los pasos previos para mejorar estos resultados de clasificación sobre patrones distintos a los empleados en el aprendizaje.

La complejidad de la tarea de diseñar un clasificador para un problema particular viene dada por diversos factores, entre los que destacan:

- Preprocesamiento de los datos.
- Elección del modelo del clasificador. El problema de la generalización.
- Determinación de los parámetros del clasificador.
- El problema de la dimensionalidad.
- Evaluación de las prestaciones de un clasificador.

Estos factores están interrelacionados, lo que complica más el problema. Dada la importancia que tienen, se desarrollarán en las secciones siguientes.

3.6.1 Preprocesamiento de los datos.

En la práctica es frecuente aplicar a los datos algún tipo de transformación, antes de ser utilizados en el proceso de diseño.

Una de las formas más comunes de preprocesamiento consiste en un simple reescalado lineal de las variables de entrada. Esto es importante porque a menudo diferentes variables o características difieren notablemente a causa de las unidades en que han sido medidas y no por su importancia relativa discriminatoria en la clasificación. Un ejemplo de escalado basado en la media y la varianza podría ser el siguiente:

$$\begin{aligned} \overline{X} &= \frac{1}{N}\sum_{i=1}^{N} X_i \\ \sigma^2 &= \frac{1}{N-1}\sum_{i=1}^{N} (X_i - \overline{X})^2 \\ \widetilde{X}_i &= \frac{X_i - \overline{X}}{\sigma} \end{aligned} \tag{3.26}$$

donde la primera expresión es el cálculo de la media de todos los valores, la segunda expresión calcula la desviación, y la tercera ecuación representa el escalado de cada valor en función de la media y la desviación calculadas previamente.

Las técnicas de pre-procesamiento destinadas a reducir la dimensionalidad de los datos de entrada son de gran importancia. La discusión sobre estas técnicas se expone en la sección 3.6.4.

El problema de ausencia de algunos datos (*missing values* [Little y Rubin, 1987]) ha sido ignorado mucho tiempo en la literatura acerca del reconocimiento de patrones. En dominios como el diagnóstico médico es frecuente que falten valores de algunas características, por ejemplo, si un médico decide no realizar un test cuyo resultado parece seguro o no es relevante para el diagnóstico. También podría ocurrir que se tratara de una característica muy difícil de medir. En otros dominios, en los que los datos se obtienen por algún procedimiento automático, es más raro que se produzca este hecho. Se han planteado diferentes soluciones a este problema, algunas tan simples como reemplazar los valores que faltan por valores "típicos" como el promedio sobre los valores observados, y otras más sofisticadas [Gharahmani y Jordan, 1994].

3.6.2 Elección del modelo del clasificador. El problema de la generalización.

La primera decisión a tomar en el diseño de un clasificador será la elección del modelo con el que se va a trabajar. Como ya se expuso anteriormente, existe una amplia variedad de modelos de clasificadores (modelos paramétricos lineales y cuadráticos, más flexibles como redes neuronales o sistemas borrosos, no paramétricos como los clasificadores de vecinos próximos, árboles de decisión, etc).

Una de las cuestiones claves relacionadas con la elección del modelo es, sin duda, su complejidad. El hecho de disponer de un conjunto finito de datos de entrenamiento condiciona en gran manera el proceso de diseño, ya que no se trata de conseguir que el sistema memorice estos datos, sino que sea capaz de hacer buenas predicciones con otros datos no presentes en el entrenamiento, es decir, que generalice adecuadamente.

Para expresar, más formalmente, cómo un número de datos finito *N* condiciona la complejidad que debe tener un modelo, elegiremos el criterio del error cuadrático para medir la bondad de un clasificador, esto es,

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k}\left\{y_k(x_n;\theta) - t_k^n\right\}^2 \tag{3.27}$$

donde t_k^n es la etiqueta de la clase correspondiente a cada dato x_n, de modo que si el dato x_n pertenece a la clase *k*, $t_k^n = 1$, y si no pertenece a la clase *k*, $t_k^n = 0$, y $y_k(X_n;\theta)$ representa la salida de un modelo clasificador de parámetros θ [Bishop, 1995].

En el caso ideal $N = \infty$, esta función de error (ecuación 3.27) puede expresarse como:

$$E = \lim_{N\to\infty}\frac{1}{2N}\sum_{n=1}^{N}\sum_{k}\left\{y_k(x_n;\theta) - t_k^n\right\}^2 \tag{3.28}$$

$$E = \frac{1}{2}\sum_{k}\iint\left\{y_k(x;\theta) - t_k\right\}^2 p(t_k, x)dt_k dx \tag{3.29}$$

Haciendo uso de la regla de Bayes, $p(t_k,x) = p(x)\ p(t_k|x)$, donde $p(x)$ es la densidad incondicional de los datos de entrada, la ecuación 3.29 se puede expresar de la siguiente manera:

$$E = \frac{1}{2}\iint\left\{y_k(x;\theta) - t_k\right\}^2 p(t_k \mid x)p(x)dt_k dx \tag{3.30}$$

A continuación, consideramos los siguientes promedios condicionados:

$$\left\langle t_k \,\middle|\, x\right\rangle = \int t_k\, p(t_k \mid x)dt_k \tag{3.31}$$

$$\left\langle t_k^{\,2} \,\middle|\, x\right\rangle = \int t_k^{\,2}\, p(t_k \mid x)dt_k \tag{3.32}$$

El promedio condicionado $\langle t_k \,|\, x \rangle$ es, de hecho, la probabilidad a posteriori de las clases $P(C_k \,|\, x)$.

Expresamos el término entre llaves $\{y_k - t_k\}^2$, contenido en la ecuación del error 3.30, como:

$$\begin{aligned} \{y_k - t_k\}^2 &= \{y_k - \langle t_k \,|\, x \rangle + \langle t_k \,|\, x \rangle - t_k\}^2 = \{y_k - \langle t_k \,|\, x \rangle\}^2 + \\ &+ 2\{y_k - \langle t_k \,|\, x \rangle\}\{\langle t_k \,|\, x \rangle - t_k\} + \{\langle t_k \,|\, x \rangle - t_k\}^2 \end{aligned} \tag{3.33}$$

Sustituyendo este término así expresado en la ecuación del error obtenemos:

$$E = \frac{1}{2}\sum_k \iint \left[\{y_k - \langle t_k \,|\, x \rangle\}^2 + 2\{y_k - \langle t_k \,|\, x \rangle\}\{\langle t_k \,|\, x \rangle - t_k\} + \{\langle t_k \,|\, x \rangle - t_k\}^2\right] p(t_k \,|\, x)\, p(x)\, dt_k\, dx \tag{3.34}$$

Esta integral se puede separar en dos integrales del siguiente modo:

$$\begin{aligned} E &= \frac{1}{2}\sum_k \iint \left[\{y_k - \langle t_k \,|\, x \rangle\}^2\right]\ p(t_k \,|\, x)\, p(x)\, dt_k\, dx + \\ &+ \frac{1}{2}\sum_k \iint \left[2\{y_k - \langle t_k \,|\, x \rangle\}\{\langle t_k \,|\, x \rangle - t_k\} + \{\langle t_k \,|\, x \rangle - t_k\}^2\right]\ p(t_k \,|\, x)\, p(x)\, dt_k\, dx \end{aligned} \tag{3.35}$$

La primera integral se transforma directamente en:

$$\frac{1}{2}\sum_k \int \{y_k - \langle t_k \,|\, x \rangle\}^2\, p(x)\, dx \tag{3.36}$$

El término entre corchetes de la segunda integral se puede desarrollar y simplificar, de modo que se puede obtener lo siguiente:

$$\begin{aligned} &\frac{1}{2}\sum_k \iint \left[{t_k}^2 - \langle t_k \,|\, x \rangle^2\right]\ p(t_k \,|\, x)\, p(x)\, dt_k\, dx = \\ &= \frac{1}{2}\sum_k \iint \left[{t_k}^2\, p(t_k \,|\, x)\, dt_k - \langle t_k \,|\, x \rangle^2\, p(t_k \,|\, x)\, dt_k\right]\ p(x)\, dx = \\ &= \frac{1}{2}\sum_k \int \left[\langle {t_k}^2 \,|\, x \rangle - \langle t_k \,|\, x \rangle^2\right]\ p(x)\, dx \end{aligned} \tag{3.37}$$

Por lo tanto, la ecuación del error expresada en función de estas dos integrales, con los cambios que hemos hecho sobre cada una, quedaría de la siguiente manera:

$$E = \frac{1}{2}\sum_k \int \left\{ y_k(x;\theta) - \langle t_k \,|\, x \rangle \right\}^2 p(x)dx + \frac{1}{2}\sum_k \int \left[\langle t_k{}^2 \,|\, x \rangle - \langle t_k \,|\, x \rangle^2 \right] \; p(x)dx \quad (3.38)$$

De la expresión se deduce que el clasificador óptimo será aquel que haga $y_i(x;\theta) = \langle t_i \,|\, x \rangle = P(C_i \mid x)$. En general, para conseguirlo, bastará con considerar un modelo lo suficientemente flexible que sea capaz de generar cualquier superficie discriminatoria. A pesar de todo, el error, en general, no será cero ya que el segundo término de la expresión no depende de $y_i(x;\theta)$ y puede tomarse como la dificultad intrínseca del problema. Esto ocurre, por ejemplo, cuando las características que definen a un objeto pueden darse en varias clases, no sólo en una.

En la práctica el problema es más complicado ya que *N* es finito y, casi siempre, reducido. Para estudiar lo que supone esto en relación a la complejidad partimos de la expresión:

$$\left\{ y_k(x;\theta) - \langle t_k \,|\, x \rangle \right\}^2 \quad (3.39)$$

que como acabamos de ver, indica lo bueno que es el modelo seleccionado. Esta medida ahora va a depender del conjunto particular de datos de entrenamiento que se utilice. Para eliminar esta dependencia promediaremos sobre todos los conjuntos de entrenamiento *D* de tamaño N_k procedentes de la misma distribución $p(x,t_k)$ para calcular:

$$\overline{E}_D \left[\left\{ y_k(x;\theta) - \langle t_k \,|\, x \rangle \right\}^2 \right] \quad (3.40)$$

donde $\overline{E}_D[\;]$ denota el valor promedio. El objetivo será elegir un modelo que haga cero esta diferencia promedio. En general existen dos motivos por los que esta diferencia va a ser diferente de cero. Para ahondar un poco más en esta cuestión resulta conveniente expandir la expresión 3.40 de la siguiente manera [Bishop, 1995]:

$$\overline{E}_D\left[\left\{y_k(x;\theta)-\langle t_k\,|\,x\rangle\right\}^2\right]=\left\{\overline{E}_D\left[y_i(x;\theta_D)-\langle t_i\,|\,x\rangle\right]\right\}^2+\overline{E}_D\left[\left\{y_i(x;\theta_D)-\overline{E}_D\left[y_i(x;\theta_D)\right]\right\}^2\right] \tag{3.41}$$

con lo que tenemos dos términos, al primero de los cuales se le denomina *sesgo* o *bias* y al segundo, *varianza* [Geman et al., 1992]. Así, el término bias refleja hasta qué punto el promedio de las salidas del clasificador difiere de la función deseada $\langle t_k\,|\,x\rangle$, mientras que el término varianza da cuenta de la sensibilidad de $y_k(x;\theta_D)$ a un conjunto de datos particular.

De este planteamiento se deduce que la clave, en lo que se refiere a la complejidad del modelo, consiste en buscar el equilibrio óptimo entre bias y varianza. Si el modelo es demasiado flexible, se "pegará" mucho a los datos particulares del entrenamiento y la varianza tenderá a ser alta. Por otro lado, si es muy poco flexible, el sesgo puede hacerse igualmente grande. Esta problemática lleva a que los modelos simples y restringidos como los lineales cobren un protagonismo importante, ya que, sobre todo en aplicaciones en las que se dispone de pocos datos, pueden dar un mejor rendimiento que clasificadores no lineales aún cuando los primeros sean un caso particular de éstos.

Esta circunstancia queda claramente reflejada en el conocido fenómeno de Hugues [Hugues, 1968], que se ilustra a través de un ejemplo en la figura 3.8, en la que se muestra como el error promedio de un clasificador cuadrático, entrenado con un número fijo de datos, crece a medida que aumenta la dimensionalidad.

Existen diversas técnicas para abordar el problema de la complejidad. Los dos enfoques principales son:

- Seleccionar iterativamente un modelo tomando como referencia alguna medida indicativa de la bondad del mismo, de tal manera que se aumente o disminuya su complejidad de acuerdo con los resultados encontrados.

- Utilizar un criterio de selección que incluya un término de penalización de la complejidad del modelo. Como ejemplos de criterios, podemos citar el criterio AIC de Akaike [Akaike, 1973], [Akaike, 1974], el enfoque de Vapnik [Vapnik, 1982] o métodos basados en validación cruzada.

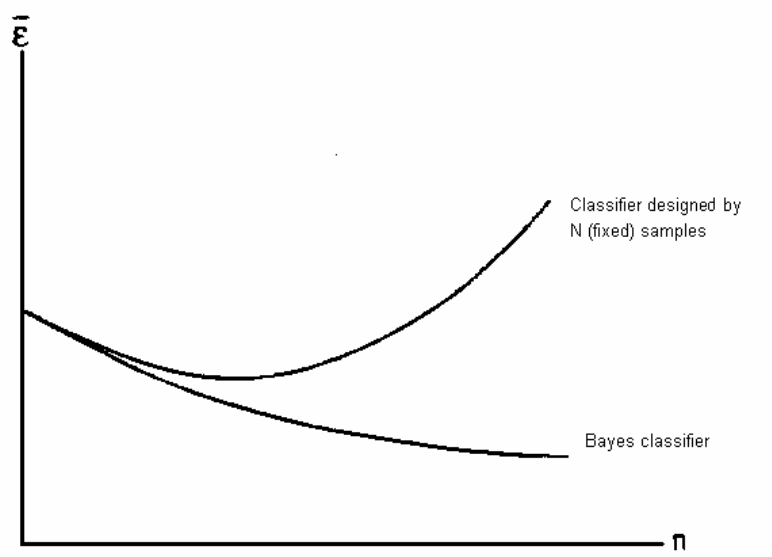


Figura 3.8. Error promedio de un clasificador cuadrático, entrenado con un número fijo de datos (figura extraída de [Fukunaga, 1990]).

3.6.3 Determinación de los parámetros de un clasificador.

En la sección anterior consideramos que teníamos una función discriminante $y_k(x;\theta)$, dependiente de un conjunto de parámetros θ. Estos parámetros pueden ser medias y matrices de covarianza, en discriminantes lineales y cuadráticos, pesos de una red neuronal, etc. En cualquier caso, se debe encontrar un procedimiento que permita calcular estos parámetros a partir de un conjunto de datos de entrenamiento. Existen diversos métodos para dar valores a θ.

En este trabajo, los clasificadores diseñados, basados en una máquina finita de estados borrosa, dependen de distintos parámetros. Algunos de ellos serán fijados a priori y otros serán determinados por medio de procesos de búsqueda basados en estrategias evolutivas. En el capítulo 6 se expondrá cuáles son los parámetros de estos clasificadores así como los algoritmos empleados para ajustar sus valores.

3.6.4 El problema de la dimensionalidad.

La dimensionalidad viene dada por el número de características con las que se trabaja. Añadir características significa añadir información, o en el peor de los casos, quedarnos como estábamos. Esto es así teóricamente, porque en la práctica, a menudo se observa

que cuando se aumenta el número de características el rendimiento del clasificador empeora. A este sorprendente efecto, debido a una excesiva dimensionalidad, se le conoce como *The Curse of Dimensionality*, y se produce por el hecho de trabajar con un conjunto finito de datos. Este efecto es mayor cuando se utilizan modelos de clasificadores no paramétricos.

Es por eso que la dimensión del espacio de características es un factor muy importante a tener en cuenta, especialmente cuando hablamos de dimensionalidades muy altas. En estos casos incluso se pierde la intuición geométrica y estadística que nos da la experiencia en espacios tridimensionales [Jiménez y Landgrebe, 1998], influyendo también en las estrategias a emplear para diseñar el clasificador más adecuado.

Por estos motivos suele resultar ventajoso la utilización de técnicas que permitan la reducción de la dimensionalidad. Éstas se dividen en dos grandes grupos:

- Técnicas que consisten en hacer combinaciones (normalmente lineales) de características que tengan un buen poder discriminatorio entre clases. De hecho, esto es equivalente a utilizar un discriminante lineal y por eso el discriminante de Fisher constituye un ejemplo de este tipo de técnicas. Aunque sin duda la técnica más extendida es el análisis de componentes principales [Watanabe, 1969], [Devijver y Kittler, 1982]. Esta técnica consiste en diagonalizar la matriz de covarianza de las características y elegir aquellos elementos con mayor valor, ya que representan las características más importantes. Esta técnica da muy buenos resultados en diversas aplicaciones, pero en el caso concreto de la discriminación entre clases no siempre es así, tal y como se puede observar en el ejemplo de la figura 3.9. En este ejemplo se pasa de dos a una sola característica mostrándose dos posibles transformaciones para realizar este cambio. Se puede apreciar que la transformación realizada en función de las componentes principales es la peor. Es muy importante elegir adecuadamente la transformación a aplicar, ya que en un caso se obtiene una buena separabilidad entre las clases y en el otro no.

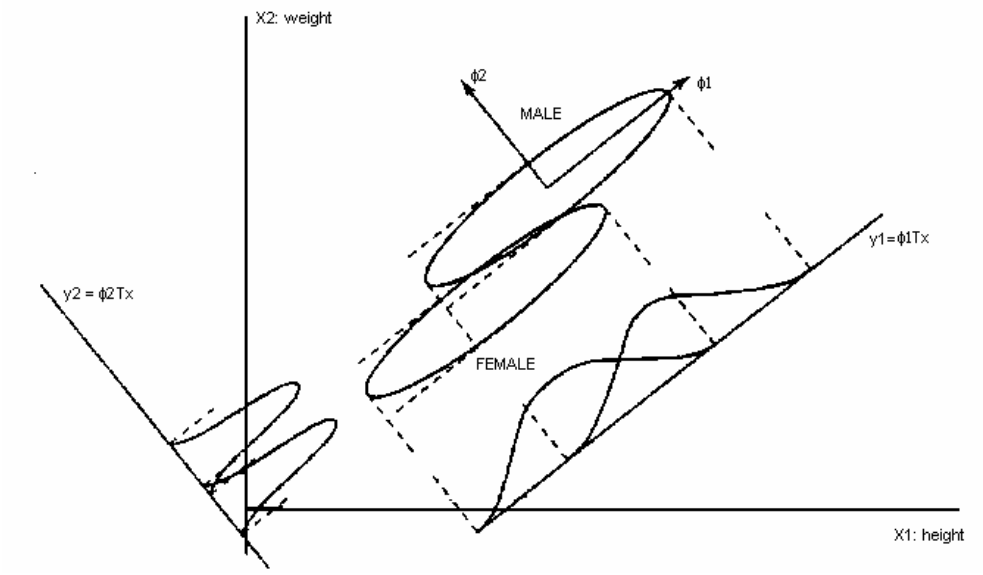


Figura 3.9. Un ejemplo de extracción de características (figura extraída de [Fukunaga, 1990]).

- Técnicas que consisten en eliminar del conjunto original características que, a través de algún procedimiento, dejen evidencia de su pobre contribución a la discriminación entre clases [Dash y Liu, 1997]. Normalmente se trata de técnicas iterativas que añaden o quitan una característica de cada vez. Entre ellas destacan, por ejemplo, los métodos del tipo *Branch and Bound*, que considera el problema de la selección de subconjuntos de características como un problema de optimización combinatoria y el subconjunto que selecciona es el mejor existente entre todas las posibles combinaciones de características, y la selección hacia delante y hacia detrás (en inglés, *forward and backward selection*).

  En el método de *selección hacia delante* (*forward selection*), se empieza evaluando todas las características disponibles por separado. Supongamos que tenemos cuatro características, tal como se observa en el ejemplo de la figura 3.10, y deseamos quedarnos con tres. Se elige de las cuatro la que consiga mejores resultados de separabilidad entre clases según algún criterio $J(*)$. En el ejemplo, es la característica número 2, es decir, $J(2)$ es el mayor valor de separabilidad. A continuación, se evalúa la capacidad de separar entre clases de los posibles subconjuntos de dos características que se pueden formar una vez elegida inicialmente la característica 2, $J(2,*)$. De nuevo, se elige el subconjunto que obtiene la separabilidad mayor, en el ejemplo este valor será $J(2,3)$, y se forman los nuevos subconjuntos de tres características posibles para evaluar la

separabilidad entre clases que tienen $J(2,3,*)$. El mejor subconjunto será el que consiga una mayor separabilidad entre clases. En este ejemplo, si el valor $J(1,2,3)$ es mayor que el valor $J(2,3,4)$, la característica eliminada será la 4.

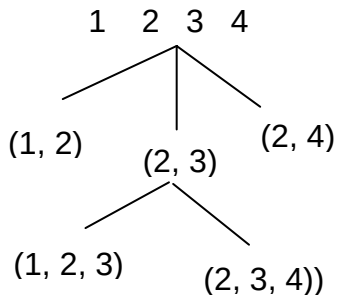


Figura 3.10. Ejemplo del método de reducción de características "selección hacia delante".

En el método de *selección hacia detrás* (*backward selection*), se comienza considerando todas las características conjuntamente. Supongamos que tenemos un conjunto de cuatro características, tal como se observa en el ejemplo de la figura 3.11, (1,2,3,4), y deseamos quedarnos con dos. Formamos todos los posibles subconjuntos de tres elementos con las características de partida y evaluamos según algún criterio $J(*)$ la separabilidad entre clases conseguida con cada subconjunto. Si $J(1,3,4)$ es el mayor valor, se selecciona ese subconjunto (con lo que ya hemos descartado la característica número 2) y repetimos la formación de subconjuntos, esta vez de dos elementos. Se evalúan de nuevo estos subconjuntos y se elige el que presente una mayor separabilidad. Por ejemplo, si $J(1,4)$ es el mayor valor, nos quedaríamos con el subconjunto (1,4) descartando las características 2 y 3.

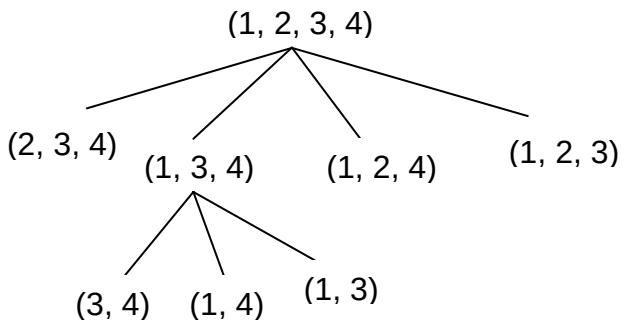


Figura 3.11. Ejemplo del método de reducción de características "selección hacia detrás".

Para la aplicación de estas técnicas es necesario utilizar algún criterio que permita evaluar la capacidad discriminatoria del conjunto reducido de características. Con este fin, y a efectos de simplificación, se suelen usar medidas de separabilidad entre clases, entre las cuales se encuentra la distancia de Bhattacharyya [Fukunaga, 1990], que es una de las más usadas. Se define para distribuciones normales como:

$$J_B = \frac{1}{8}(\mu_2 - \mu_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \log \left(\frac{\left| \frac{\Sigma_1 + \Sigma_2}{2} \right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \right) \qquad (3.42)$$

donde los μ_i representan los vectores de medias y los Σ_i las matrices de covarianza.

Como puede verse, consta de dos términos, el primero da la separabilidad entre clases debido a la diferencia de las medias y el segundo refleja la separación introducida por la diferencia de covarianzas. Aunque se haya definido para distribuciones normales, es una medida razonable aplicable a otro tipo de distribuciones.

Hay que tener en cuenta que todas estas técnicas suponen normalmente una pérdida de información, que de llegar a ser demasiado importante podría no compensar la reducción lograda en la dimensionalidad.

3.6.5 Evaluación de las prestaciones de un clasificador.

En el apartado 3.5.2 se expresó de forma cuantitativa la influencia que tiene sobre el cálculo del error cuadrático de clasificación el disponer de un conjunto de datos de entrenamiento. Esta influencia quedaba en función de dos términos: bias y varianza. Además, en la práctica, no se dispone de infinitas muestras y se suele contar cuántos de los datos disponibles han sido asignados a cada clase. Por lo tanto, aparte del sesgo y la varianza, hay que tener en cuenta esta circunstancia.

Existen varias formas de organizar los datos para el cálculo del error. La más directa consiste en utilizar los mismos datos que en el entrenamiento. A este procedimiento se le llama método de resustitución (en inglés, *resubstitution method*). El problema de este método es que aparte de añadir más varianza al error, también contribuye al sesgo y da una estimación optimista del error.

Más fiable resulta el procedimiento conocido como método *Holdout* que consiste en dividir el conjunto de datos en dos subconjuntos, uno para el entrenamiento y otro para el test o validación. Como ahora tenemos dos conjuntos de datos independientes, no habrá contribución al sesgo (viniendo enteramente del proceso de entrenamiento) y sí que seguirá habiendo contribución a la varianza. Cuando los datos se generan artificialmente, éste parece un procedimiento ideal, pero en la práctica supone dividir aún más el conjunto de datos, que de por sí es reducido. Además está la cuestión no trivial, de cómo hacer la división, ya que las distribuciones de ambos subconjuntos deberían de parecerse lo más posible, e incluso no está claro cuántos datos deben de recaer en cada uno de ellos.

Por todo esto, existe otro procedimiento denominado método *Leave-One-Out.* En este método se excluye un dato, se entrena el clasificador con los N-1 restantes, y el dato excluido es utilizado para validar el clasificador. Esta operación es repetida N veces. Como cada dato de test se excluye del conjunto de entrenamiento, queda garantizada la independencia. Además se utilizan de forma efectiva todos los datos, tanto para entrenar como para validar, por eso a este tipo de métodos se les denomina *de validación cruzada* (en inglés, *cross-validation*). No hay necesidad de preocuparse por diferencias entre las distribuciones de los datos de entrenamiento y test. Quizás el inconveniente más serio que presenta sea el hecho de tener que entrenar N clasificadores, lo cual, puede suponer un tiempo de computación considerable. Afortunadamente para ciertos clasificadores, como los discriminantes lineal y cuadrático, este tiempo puede ser prácticamente igual al equivalente a entrenar un solo clasificador [Fukunaga, 1990]. En este trabajo, el tiempo que tardamos en entrenar los clasificadores que hemos utilizado (basados en sistemas borrosos) es un factor muy importante debido a que es muy grande, por lo tanto, nos ha resultado imposible aplicar la validación cruzada de este modo. La validación cruzada que se ha aplicado ha consistido en dividir el conjunto de ejemplos disponibles en tres subconjuntos, de modo que se ha entrenado el clasificador tres veces, eligiendo para cada proceso de entrenamiento un subconjunto distinto como conjunto de entrenamiento. Los tres clasificadores obtenidos se han evaluado con los dos subconjuntos que no se usaron en su etapa de entrenamiento. Los resultados de las tres evaluaciones se han promediado para obtener el error medio que presentan estos tipos de clasificadores.

Es importante considerar que, aún cuando el error obtenido por un clasificador sea muy bajo, puede resultar inaceptable viniendo de un sistema clasificador tipo "caja negra". Por eso, ciertos clasificadores, que no resultan tan eficientes en términos del error de clasificación, pueden ser preferibles debido a su capacidad explicativa.

Hasta ahora nos hemos referido a una única medida de las prestaciones de un clasificador. En ciertos dominios, como la Medicina, puede ser de gran utilidad ponderar adecuadamente este error entre las diferentes clases. Así, a modo de ejemplo, no supone lo mismo diagnosticar a un paciente enfermo como sano que a uno sano como enfermo. Especial importancia tiene la clasificación de patrones de este dominio en esta tesis, por lo que se comentará en el apartado 3.7 el método de evaluación de la curva ROC, empleado en este trabajo para medir la bondad del clasificador desde el punto de vista médico.

3.7 Evaluación de pruebas diagnósticas. Curvas ROC.

Todo proceso de medición (proceso mediante el cual se cuantifica una magnitud) está amenazado por diversas fuentes de error, derivadas tanto de las limitaciones del instrumento de medida, como de la naturaleza de la magnitud a medir. Clásicamente, se distingue entre el error debido a la *precisión* limitada del instrumento que atenta a la *reproducibilidad* de la medición introduciendo un error aleatorio en la misma, y el debido a la *validez*, también limitada, que introduce un error sistemático, denominado *sesgo*.

En ciertas situaciones, en la práctica clínica entre ellas, el proceso de control de la precisión y validez de una medida es más complejo, debido a dos fenómenos distintos. Por una parte, las magnitudes a medir son *aleatorias*, es decir presentan diversos grados de variabilidad impredecible propia (por ejemplo, esto ocurre al medir presión, temperatura, etc). Por otra, además de magnitudes tales como presión, temperatura, concentración de hemoglobina en sangre, etc., se trabaja con magnitudes como dolor, mejoría en un proceso patológico, grado pronóstico de una afección, etc., para las cuales no existe un patrón de referencia claro y objetivo ni escala métrica apropiada y que, por tanto, suelen describirse en escalas ordinales o, incluso, nominales, cuya apreciación puede estar muy distorsionada por influencias subjetivas. Estas magnitudes suelen denominarse *variables blandas* y dan lugar a clasificaciones mejor

que a mediciones en sentido estricto (que implica la existencia de una escala métrica). Evidentemente, existen también variables objetivas (*"duras"*) que dan lugar a clasificaciones, por ejemplo muerto/vivo.

Los procesos de clasificación sufren los mismos problemas de validez y precisión que los de medición, pero con ciertas complicaciones añadidas en el caso de las variables blandas. Para controlar su validez, no suelen existir patrones de referencia, o no son tan objetivos o accesibles como en el caso de una magnitud física. En este sentido se suele distinguir entre dos modos de controlar la validez de un instrumento de medida (entendiendo el término *instrumento de medida* en un sentido muy amplio, no es sólo el "aparato" usado para obtener una medida determinada, sino el conjunto formado por el aparato que produce la medida y el observador que la interpreta, siendo, además, éste último más crítico para los errores de medición-clasificación): cuando se hace con patrones objetivos se habla de *exactitud* (*"accuracy"* en la literatura clínico-epidemiológica inglesa), mientras que cuando se controla comparando simplemente con una referencia considerada mejor (*"gold standard"*) se habla de *conformidad*. El *gold standard* es una prueba que actúa como patrón de referencia. Debe estar reconocida como tal por la comunidad científica, se debe aplicar a toda la serie de casos estudiados y no tiene que incorporar información procedente de la prueba diagnóstica que se evalúa (sesgo de incorporación: evitar que el resultado de una de las pruebas pueda influir sobre la interpretación de los resultados de la otra).

En cuanto a la reproducibilidad, sobre todo con métodos de clasificación, se distingue entre la reproducibilidad del mismo instrumento (típicamente un observador en este caso) en dos instantes de tiempo diferentes y se habla de *concordancia* (*"agreement"* en la literatura en inglés) *o consistencia interna o intraobservador*. Por ejemplo un radiólogo puede no clasificar igual la misma radiografía si la estudia en distintos momentos. También se considera la *concordancia o consistencia externa o interobservador.* Como ejemplo, dos radiólogos diferentes pueden no clasificar del mismo modo la misma radiografía.

La toma de decisiones clínicas es un proceso extremadamente complejo en el que se debe valorar la utilidad de cualquier prueba diagnóstica. En este contexto, es imprescindible conocer detalladamente la exactitud de las distintas pruebas, es decir, su capacidad para clasificar correctamente a los pacientes en categorías o estados en

relación con la enfermedad (típicamente dos: estar o no estar enfermo, respuesta positiva o negativa a la terapia, etc).

Generalmente, la exactitud diagnóstica se expresa como *sensibilidad* y *especificidad* diagnósticas. Cuando se utiliza una prueba *dicotómica* (una cuyos resultados se puedan interpretar directamente como positivos o negativos), la **sensibilidad** es la probabilidad de clasificar correctamente a un individuo cuyo estado real sea el definido como positivo respecto a la condición que estudia la prueba, razón por la que también es denominada fracción de verdaderos positivos (FVP). La **especificidad** es la probabilidad de clasificar correctamente a un individuo cuyo estado real sea el definido como negativo. Es igual al resultado de restar a uno la fracción de falsos positivos (FFP).

Cuando los datos de una muestra de pacientes se clasifican en una tabla de contingencia por el resultado de la prueba y su estado respecto a la enfermedad, es fácil estimar a partir de ella la sensibilidad y la especificidad de la prueba, tal y como se puede observar en la tabla 3.1. Es importante destacar que lo que realmente obtenemos son *estimaciones* de los verdaderos valores de sensibilidad y especificidad para una población teórica de la que suponemos que el grupo de pacientes considerado constituye una muestra aleatoria. Por lo tanto, un tratamiento estadístico correcto de cantidades como las calculadas por el método descrito por la tabla 3.1, exigiría incluir medidas de su precisión como estimadores, y, mejor aún, utilizarlas para construir intervalos de confianza para los verdaderos valores de sensibilidad y especificidad.

		Diagnóstico verdadero	
		Enfermo	**Sano**
Resultado de la prueba	**Prueba positiva**	Verdadero Positivo (VP)	Falso Positivo (FP)
	Prueba negativa	Falso Negativo (FN)	Verdadero Negativo (VN)
		VP + FN	FP + VN
Sensibilidad	= VP/(VP + FN) = FVP (fracción de verdaderos positivos)		
Especificidad	= VN/(VN + FP) = FVN (fracción de verdaderos negativos) = 1 - FFP (fracción de falsos positivos)		

Tabla 3.1. Resultado de una prueba y su estado respecto a la enfermedad.

Por lo tanto, siguiendo la nomenclatura de la tabla 3.1, se observa que:

- VP son los *verdaderos positivos*: número de pacientes enfermos en los que la prueba dio positiva (diagnóstico correcto).
- FP son los *falsos positivos*: número de pacientes sanos en los que la prueba dio positiva (diagnóstico incorrecto).
- FN son los *falsos negativos*: número de pacientes enfermos en los que la prueba dio negativa (diagnóstico incorrecto).
- VN son los *verdaderos negativos*: número de pacientes sanos en los que la prueba dio negativa (diagnóstico correcto).

Llamando n = VP + FP + FN + VN al número de pacientes sometidos a la prueba, los datos muestrales anteriores permiten estimar las siguientes probabilidades:

$$p\,(P, E) = \mathrm{VP/n} \tag{3.43}$$

$$p\,(P, no\text{-}E) = \mathrm{FP/n} \tag{3.44}$$

$$p\,(no\text{-}P, E) = \mathrm{FN/n} \tag{3.45}$$

$$p\,(no\text{-}P, no\text{-}E) = \mathrm{VN/n} \tag{3.46}$$

donde que un paciente esté o no enfermo se simboliza por E y *no-E* y la prueba diagnóstica, cuyo resultado puede ser positivo o negativo, se simboliza por P o *no-P*, respectivamente.

A partir de los datos anteriores se estiman estas otras cantidades de interés:

- *Sensibilidad* o *tasa de verdaderos positivos*: es la probabilidad de que a un individuo enfermo (E) la prueba le dé resultado positivo (P), lo que formalmente se reduce a calcular la probabilidad condicionada:

$$p\,(P|\,E) = \mathrm{VP/(VP + FN)} \tag{3.47}$$

- *Especificidad* o *tasa de verdaderos negativos*: es la probabilidad de que a un individuo sano (*no-E*) la prueba le dé resultado negativo (*no-P*), lo que se reduce a calcular:

$$p\,(no\text{-}P|\,no\text{-}E) = \mathrm{VN}/(\mathrm{FP} + \mathrm{VN}) \tag{3.48}$$

- *Prevalencia* o *tasa de incidencia*: es la probabilidad de que un individuo padezca la enfermedad (*E*) en la población bajo estudio, aquella de la cual se extrajo la muestra. Este valor es:

$$p\,(E) = (\mathrm{VP} + \mathrm{FN})/\mathrm{n} \tag{3.49}$$

Como la sensibilidad y especificidad de las pruebas diagnósticas son antagonistas, si una aumenta la otra disminuye, y en la aplicación de una técnica de diagnóstico siempre se tiende a favorecer la detección de positivos o de negativos lo que introduce cierto grado de inseguridad en las técnicas. Esa tendencia a favorecer la detección de positivos o de negativos se puede medir utilizando la denominada *Razón de Verosimilitud* (en inglés, *Likelihood Ratio* - LR) que se calcula como:

$$\mathrm{LR} = \text{Sensibilidad} / \text{1-Especificidad} \tag{3.50}$$

cuya interpretación es:

LR = 1; el resultado de la prueba aplicada no es informativo.
LR >1; la prueba favorece la detección de casos positivos (enfermos).
LR <1; la prueba favorece la detección de casos negativos (sanos).

Esta medida también se conoce con el nombre de *cociente de probabilidad*. Los cocientes de probabilidad ofrecen la ventaja de que relacionan la sensibilidad y la especificidad en un solo índice que, además, no varía con la prevalencia.

Lo que interesa en la práctica es estimar la probabilidad de que el paciente esté sano o enfermo, según la prueba haya resultado negativa o positiva:

- *Valor predictivo de una prueba positiva*: es la probabilidad de que un individuo padezca la enfermedad cuando la prueba diagnóstica ha sido positiva:

$$p\,(E|\,P) = \mathrm{VP}/(\mathrm{VP} + \mathrm{FP}) \tag{3.51}$$

- *Valor predictivo de una prueba negativa*: es la probabilidad de que un individuo esté sano cuando la prueba diagnóstica ha sido negativa:

$$p\,(no\text{-}E|\,no\text{-}P) = \mathrm{VN}/(\mathrm{FN} + \mathrm{VN}) \tag{3.52}$$

Los valores predictivos, tanto el positivo como el negativo dependen de forma muy importante de la prevalencia de la enfermedad, que indica la proporción de personas afectadas en ese momento. Al aumentar la prevalencia crece el valor predictivo positivo para una misma sensibilidad y especificidad, lo cual se debe, fundamentalmente, a que disminuye el número de falsos positivos. Por otra parte, cuando disminuye la prevalencia, se reduce también el valor predictivo positivo y aumenta el negativo, dado que para una misma sensibilidad y especificidad disminuyen los falsos negativos.

En el análisis de tablas de contingencia no debe perderse de vista la técnica muestral utilizada. Tres son las situaciones típicas que se suelen presentar:

- *Muestreo mixto*: los individuos que forman parte de la muestra se seleccionan aleatoriamente de toda la población objeto del análisis, por ejemplo, pacientes con determinados síntomas que los hacen candidatos a padecer cierta enfermedad.

- *Muestreo caso-control*: se estratifica la población en sanos y enfermos, tomando a continuación muestras de las dos subpoblaciones.

- *Muestreo de cohortes*: se estratifica también la población, pero ahora en individuos con prueba diagnóstica positiva o negativa. Dentro de cada grupo se hace un muestreo aleatorio simple.

Por ejemplo, la biopsia por aspiración con aguja fina de glándula salival es un procedimiento fácil, seguro y económico que en muchas circunstancias permite un diagnóstico rápido con molestias mínimas para el paciente y con sensibilidad de 81-100% y especificidad de 94-100% [Pérez et al., 1996].

Los sesgos más frecuentes en la evaluación de pruebas diagnósticas son:

- *Sesgo de confirmación diagnóstica* al limitar el estudio a los pacientes a quienes se les hizo en su día el "gold standard" que suelen ser los que más probablemente tengan la enfermedad, por tanto las pruebas positivas están sobre-representadas (sobreestimación de la sensibilidad) y las negativas infra-representadas (infraestimación de la especificidad). Frecuentemente es imposible evitarlo por razones éticas. Hay técnicas matemáticas complejas para controlarlo.

- *Sesgo de interpretación de las pruebas* si no se hacen independientemente.

- *Sesgo debido a resultados no interpretables* de la prueba problema si dicho problema no tiene la misma frecuencia en ambos grupos.

- *Ausencia de gold standard* definitivo.

3.7.1 La curva ROC.

La limitación principal del enfoque hasta ahora expuesto está en la exigencia de que la respuesta proporcionada por la prueba diagnóstica sea de tipo dicotómico, por lo que en principio quedaría excluida la amplia gama de pruebas diagnósticas cuyos resultados se miden en una escala (nominalmente) continua o, al menos, discreta ordinal. Por ejemplo, una prueba en que los resultados se expresen empleando las categorías "seguramente normal", "probablemente normal", "dudoso", "probablemente anormal" y "seguramente anormal".

La generalización a estas situaciones se consigue mediante la elección de distintos *niveles de decisión* o *valores de corte* que permitan una clasificación dicotómica de los valores de la prueba según sean superiores o inferiores al valor

elegido. La diferencia esencial con el caso más simple es que se cuenta no con un único par de valores de sensibilidad y especificidad que definan la exactitud de la prueba, sino más bien con un conjunto de pares correspondientes cada uno a un distinto nivel de decisión.

Este procedimiento constituye la esencia del análisis ROC (Característica Receptor-Operativa, en inglés, *Receiver Operating Characteristic*) , una metodología desarrollada en el seno de la Teoría de la Decisión en los años 50 y cuya primera aplicación fue motivada por problemas prácticos en la detección de señales por radar (por ejemplo, la equivalencia entre el operador que interpreta los picos en la pantalla del radar para decidir sobre la presencia de un misil y el médico que emplea el resultado de una prueba diagnóstica para decidir sobre la condición clínica del paciente, es completa [Robertson y Zweig, 1981]). La aparición del libro de Swets y Pickett [Swets y Picket, 1982] marcó el comienzo de su difusión en el área de la Biomedicina, inicialmente en Radiología, donde la interpretación subjetiva de los resultados se recoge en una escala de clasificación, pero de modo creciente en relación con cualquier método diagnóstico que genere resultados numéricos.

Supongamos que, tanto para la población sana como para la enferma, la variable de decisión que representa el resultado de la prueba diagnóstica se distribuye normalmente, con media y desviación típica conocidas. En la figura 3.12 se muestran las funciones de densidad de probabilidad para ambas variables, que mostrarán un determinado nivel de solapamiento.

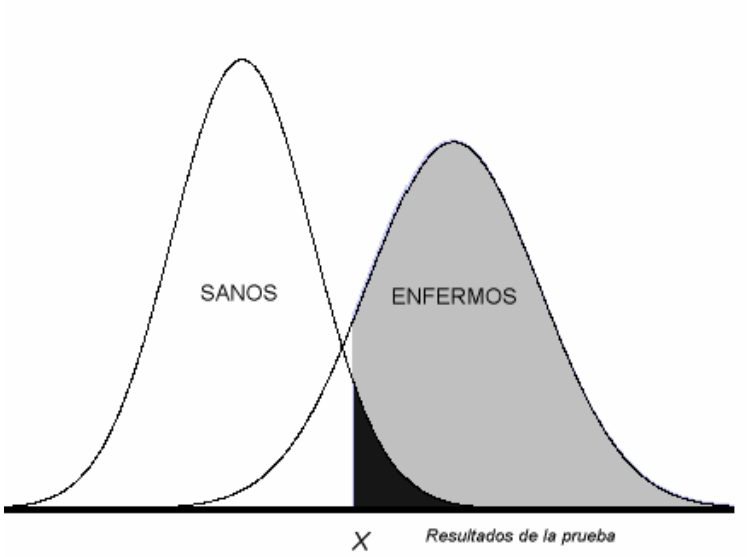


Figura 3.12. Distribución de resultados de una prueba en las poblaciones de pacientes sanos y enfermos.

Si consideramos un valor arbitrario del resultado de la prueba *X*, al que, llamamos *valor de corte*, la FVP (sensibilidad) y la FFP (1-especificidad) se corresponderán respectivamente con el área a la derecha de ese punto bajo la función de densidad de probabilidad de la población enferma (áreas clara y oscura) y de la población sana (área oscura). La curva ROC se obtiene representando, para cada posible elección de valor de corte, la FVP en ordenadas y la FFP en abscisas, como se puede observar en la figura 3.13.

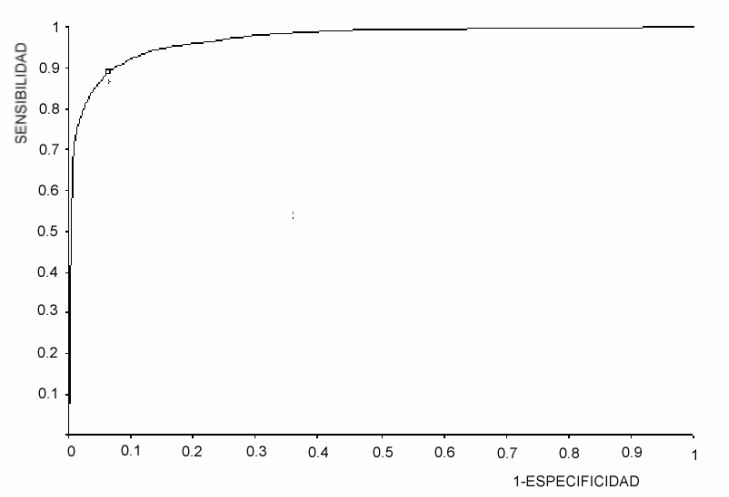


Figura 3.13. Curva ROC correspondiente a la distribución teórica de los resultados de una prueba representada en la figura 3.12.

Mediante esta representación de los pares (1-especificidad, sensibilidad) obtenidos al considerar todos los posibles valores de corte de la prueba, la curva ROC nos proporciona una representación global de la exactitud diagnóstica. La curva ROC es necesariamente creciente, propiedad que refleja el compromiso existente entre sensibilidad y especificidad: si se modifica el valor de corte para obtener mayor sensibilidad, sólo puede hacerse a expensas de disminuir al mismo tiempo la especificidad. Si la prueba no permitiera discriminar entre grupos, la curva ROC sería la diagonal que une los vértices inferior izquierdo y superior derecho. La exactitud de la prueba aumenta a medida que la curva se desplaza desde la diagonal hacia el vértice superior izquierdo. Si la discriminación fuera perfecta (100% de sensibilidad y 100% de especificidad) pasaría por dicho punto. Las pruebas habituales tienen curvas intermedias. En la figura 3.14 se muestran distintos tipos de curvas ROC.

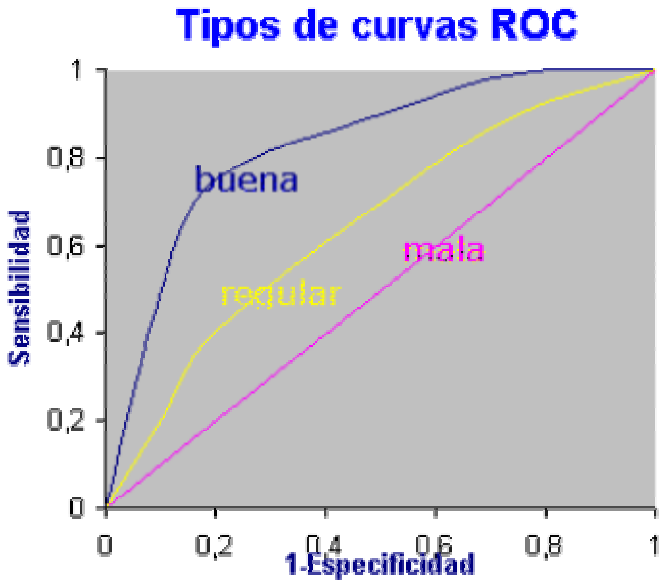


Figura 3.14. Distintos tipos de curvas ROC.

Muchas pruebas producen resultados continuos, por ejemplo, el nivel de glucosa en sangre para diagnosticar la diabetes. El comportamiento de dichas pruebas depende de donde se ponga el punto de corte y lo habitual es que exista un grado variable de solapamiento en la función de probabilidad de la variable resultado. En el caso de la glucosa, la situación se representa en la gráfica figura 3.15.

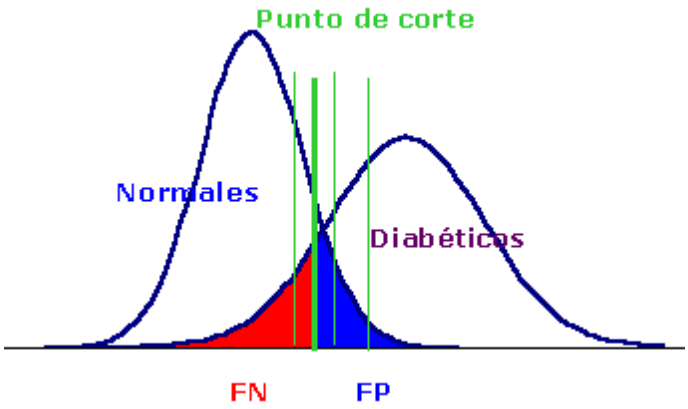


Figura 3.15. Distribución de resultados de la prueba del nivel de glucosa en sangre para diagnosticar diabetes en las poblaciones de pacientes sanos y enfermos.

Si se desplaza el punto de corte a la derecha (valores mayores de glucosa) disminuyen los falsos positivos (región azul) pero aumentan los falsos negativos (región roja) o, en otros términos, disminuye la sensibilidad y aumenta la especificidad e

inversamente si se desplaza a la izquierda, de modo que un problema en estas pruebas es la selección del punto de corte óptimo.

El último punto de los expuestos (elección del punto de corte más apropiado) se desarrollará con más detalle en la sección 3.7.3.

El modelo anterior, aplicable en principio a datos continuos, puede generalizarse al caso en que los datos se obtiene por algún sistema de clasificación en una escala discreta ordinal. Para ello basta suponer la existencia de unas variables latentes con distribución normal y de unos límites fijos que marcan los extremos de cada categoría. La figura 3.16 muestra esquemáticamente este modelo para un ejemplo con cinco categorías.

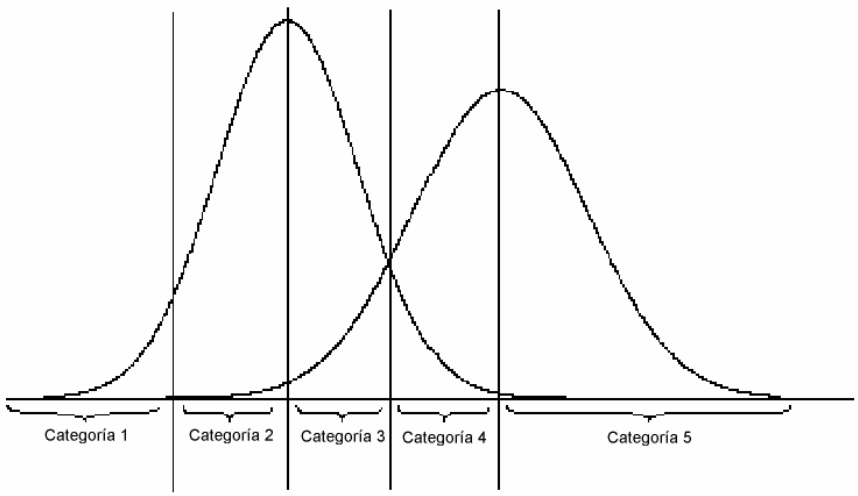


Figura 3.16. Representación esquemática de un modelo para datos de clasificación con cinco categorías.

Las curvas ROC son muy utilizadas para evaluar pruebas diagnósticas en el campo de la medicina. Como ejemplos, se pueden citar los trabajos relacionados con el diagnóstico de la disfunción sistólica del ventrículo izquierdo [Muders et al., 1997] , la hipoxemia infantil [Usen et al., 1999], el cáncer de próstata [Parkes et al., 1995] y la pancreatitis aguda [Hedstrom et al., 1996].

Las curvas ROC son útiles para:

- Conocer el rendimiento global de una prueba.
- Comparar dos pruebas o dos puntos de corte.
- Comparar dos curvas o dos puntos sobre una curva.

- Elegir el punto de corte más apropiado para un determinado paciente.

Las principales ventajas de las curvas ROC son las siguientes:

- Son una representación gráfica simple, fácil de interpretar, de la capacidad de discriminación según todos los puntos de corte.
- No requieren un nivel de decisión particular porque están incluidos todos los posibles puntos de corte.
- Al obtenerse a partir de los valores de sensibilidad y especificidad, las curvas ROC son independientes de la prevalencia de la enfermedad en el grupo de sujetos estudiado.
- Proporcionan una comparación visual directa entre pruebas en una escala común.

Las principales desventajas se pueden resumir en los siguientes puntos:

- Los puntos de corte utilizados para elaborar la curva no aparecen en el gráfico. Al observar un punto de la curva se deduce la sensibilidad y la especificidad que tiene asociadas, pero no se conoce el valor concreto de dicho punto.
- El número de sujetos de la muestra estudiada tampoco aparece en el gráfico.
- Al disminuir el tamaño de la muestra, la curva tiende a hacerse más escalonada y desigual.

Obviamente, el escenario en el que se ha presentado la curva ROC es completamente teórico, por dos razones relacionadas entre sí:

- En la práctica no se dispone de las poblaciones (abstractas) de enfermos y sanos, sino simplemente de una muestra de ellas.
- En general, no se conocen las distribuciones de los valores de la prueba diagnóstica en dichas poblaciones.

Estas limitaciones hacen obligatorio considerar el problema práctico de la construcción de curvas ROC, que se trata a continuación, desde un punto de vista típicamente estadístico.

3.7.2 Métodos de cálculo de la curva ROC.

Un primer grupo de métodos para construir la curva ROC lo constituyen los llamados métodos *no paramétricos*. Se caracterizan por no hacer ninguna suposición sobre la distribución de los resultados de la prueba diagnóstica. El más simple de estos métodos se conoce como *empírico*, consiste simplemente en representar todos los pares [FFP, FVP] (todos los pares [1-especificidad, sensibilidad]), para todos los posibles valores de corte que se puedan considerar con la muestra particular de que se dispone. Éste es el método que se aplica en este trabajo.

Desde un punto de vista técnico, este método sustituye las funciones de distribución teóricas por una estimación no paramétrica de ellas, es decir, la función de distribución empírica construida a partir de los datos. Informalmente, es como si en la figura 3.12 se sustituyesen las funciones de densidad por histogramas obtenidos a partir de la muestra de pacientes sanos y enfermos y se construyera la curva ROC a partir de ellos.

La representación obtenida por este método tiene forma de escalera, tal y como se puede observar en la figura 3.17. Para cada variación mínima del valor de corte que produzca cambios en sensibilidad o especificidad, al menos un caso pasa a ser considerado bien como verdadero positivo, lo que se corresponde con un trazo vertical, bien como falso positivo, lo que da lugar a un trazo horizontal. Existe aún otra posibilidad, derivada de la posibilidad de que se produzcan empates, es decir, dos o más casos con el mismo valor de la prueba: si el empate ocurre entre un caso del grupo enfermo y otro del grupo sano aparecerá un trazo diagonal en la representación.

Es evidente que este método es especialmente idóneo para datos de tipo continuo, sobre todo si la discretización (el redondeo) inducida por la precisión del método analítico utilizado no es muy importante, de modo que el número de empates sea proporcionalmente escaso. En este caso, la apariencia dentada de la curva es menos notoria a medida que crece el tamaño de la muestra e, idealmente, en el límite tendríamos una curva suave, la propia curva ROC teórica.

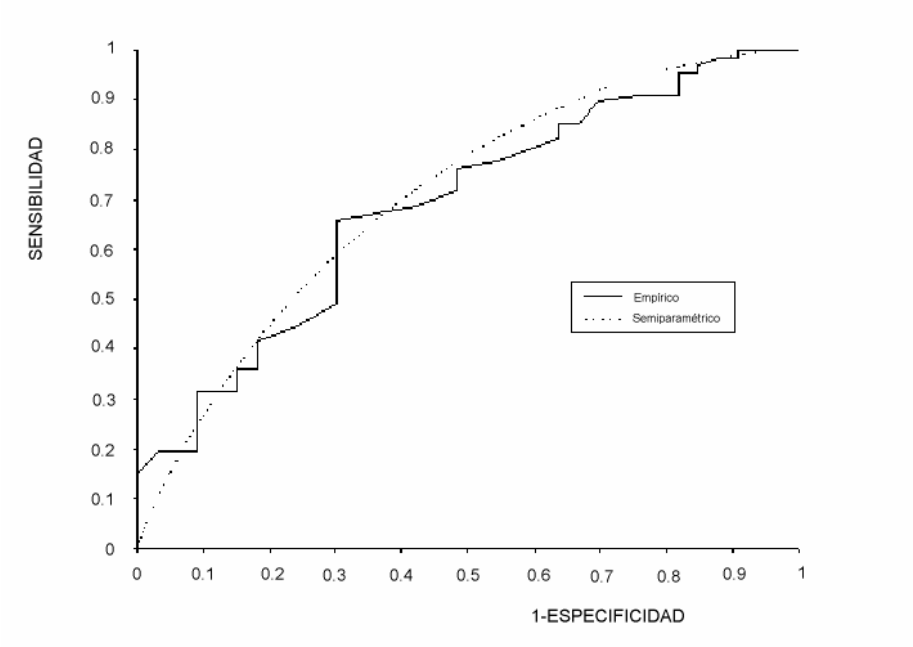


Figura 3.17. Curvas ROC calculadas por los métodos empírico y semiparamétrico.

No obstante, también puede aplicarse a datos de tipo categórico. En este caso resulta inevitable la aparición de empates (al menos si el tamaño de la muestra es mayor que el número de categorías), con la consecuencia de que el gráfico consistirá, independientemente del tamaño de la muestra, en un número fijo de líneas en general diagonales que unen los puntos correspondientes a los pares (1-especificidad, sensibilidad) calculados para cada categoría. En la figura 3.18 se presenta un ejemplo de la aplicación de este método a un conjunto de datos procedente de la clasificación en cinco categorías de imágenes obtenidas por tomografía computerizada. Estos datos están recogidos en la tabla 3.2 [Hanley y McNeil, 1982].

Existen otros métodos no paramétricos [Zou et al., 1997] aplicables a datos continuos que permiten obtener curvas ROC suavizadas, en contraposición con la forma dentada de la curva obtenida por el método empírico. Se basan en obtener estimaciones no paramétricas suavizadas de las funciones de densidad de las dos distribuciones de resultados de la prueba empleando generalmente estimadores de tipo núcleo. A partir de dichas densidades (en lugar de a partir de los histogramas, como en el método anterior) se obtiene directamente una curva ROC más suave.

Condición Verdadera	Seguramente Normal	Probablemente Normal	Dudosa	Probablemente Anormal	Seguramente Anormal
Normal	33	6	6	11	2
Anormal	3	2	2	11	33

Tabla 3.2. Clasificación de 109 imágenes de tomografía computerizada.

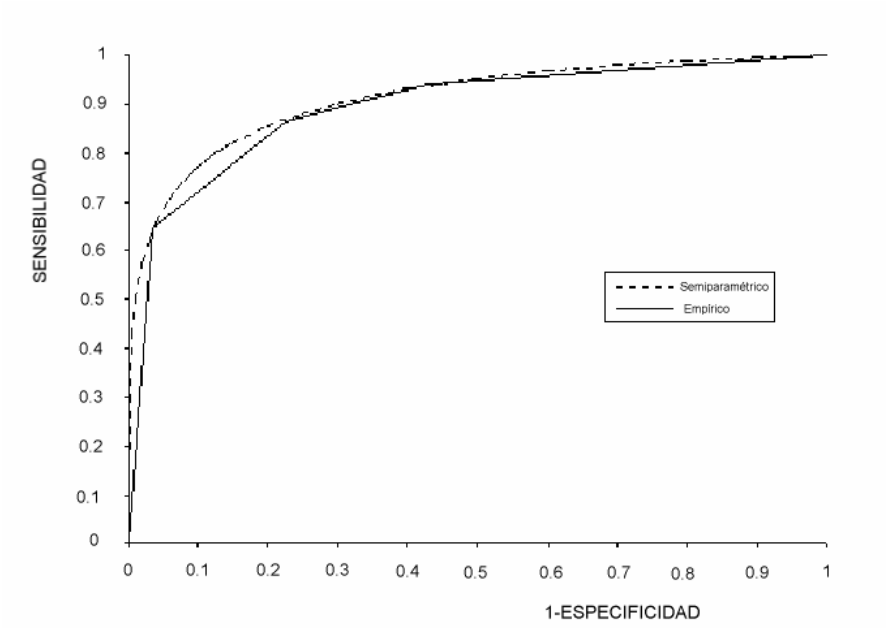


Figura 3.18. Curvas ROC calculadas por los métodos empírico y semiparamétrico para los datos de tomografías computerizadas.

Los métodos *paramétricos* se basan en postular un determinado tipo de distribución para la variable de decisión en las dos poblaciones que se trata de distinguir [Swets y Piccket, 1982]. El modelo más frecuentemente utilizado es el binormal, que supone la normalidad de las variables tanto en la población sana como en la enferma, pero existen muchos otros modelos posibles que surgen al considerar distintas distribuciones, similares a la normal como la logística (modelo bilogístico) o no, como la exponencial negativa.

El problema ahora se reduce a estimar los parámetros de cada distribución por un método estadísticamente adecuado, en general, el método de máxima verosimilitud. Se obtiene así una curva ROC suave, pero puede ocurrir una sustancial falta de ajuste si los supuestos distribucionales resultan ser erróneos [Zweig y Campbell, 1993], [Burgueño et al., 1995]. Por ello, si se va a emplear este método, debe previamente someterse la hipótesis sobre la naturaleza de las distribuciones a un contraste de significación. También es recomendable emplear una transformación de la variable inicial que logre que los datos sean más compatibles con las distribuciones asumidas,

aunque este juicio sólo puede basarse en un examen en gran medida visual y por lo tanto está expuesto a interpretaciones subjetivas.

Estas limitaciones hacen que el método no sea en general recomendable para datos continuos. Su utilidad es mayor con datos discretos: varios investigadores han examinado el modelo binormal para datos de clasificación, sin encontrar situaciones en las que el modelo fallara seriamente [Hanley, 1988].

Según otro método [Metz et al., 1998], primero se agrupan los datos en categorías ordenadas y después se aplica un algoritmo paramétrico para crear una curva ROC suave. Del método se dice que es *semiparamétrico* [Zou et al., 1997], [Hsieh y Turnbull, 1996] porque aunque supone la existencia de una transformación que haga que las dos distribuciones sean aproximadamente normales, ésta se deja sin especificar.

La dependencia mucho menor de la validez de las asunciones se debe principalmente a la invarianza de la curva ROC frente a las transformaciones monótonas de la escala de la variable de decisión [Metz et al., 1998]. Otras investigaciones parecen indicar que el método se comporta empíricamente bien en una amplia variedad de situaciones [Hanley, 1996]. Permanece, no obstante, el problema de que el ajuste no es reproducible a menos que el esquema de categorización empleado sea objetivo y esté estandarizado. Las figuras 3.17 y 3.18 muestran las curvas ROC ajustadas por este método y permiten compararlas a las obtenidas por el método empírico.

3.7.3 Análisis estadístico de las curvas ROC.

En el análisis de las curvas ROC es importante considerar el factor de la v*ariabilidad interobservador*, ya que todas las pruebas (unas más que otras) requieren cierto grado de pericia en su realización e interpretación. Dos observadores pueden ser igualmente exactos pero ser uno más sensible o específico que otro, en otras palabras operar con la misma curva ROC pero en puntos distintos o pueden tener distinta exactitud (operar en la misma prueba con distinta curva ROC). Por este motivo, se intentan definir métodos que permitan realizar un análisis más independiente del observador.

3.7.3.1 Área bajo la curva.

Como ya se ha mencionado anteriormente, la mayor exactitud diagnóstica de una prueba se traduce en un desplazamiento "hacia arriba y a la izquierda" de la curva ROC. Esto sugiere que el área bajo la curva ROC se puede emplear como un índice conveniente de la exactitud global de la prueba: la exactitud máxima correspondería a un valor de área de 1 y la mínima a uno de 0.5 (si fuera menor de 0.5 debería invertirse el criterio de positividad de la prueba).

En términos probabilísticos, si X_E y X_S son las dos variables aleatorias que representan los valores de la prueba en las poblaciones enferma y sana, respectivamente, puede probarse que el área de la "verdadera" curva ROC (intuitivamente, aquella que se obtiene si el tamaño de la muestra fuera infinito y la escala de medida continua) es precisamente $\theta = Pr\{X_E > X_S\}$, es decir, la probabilidad de que, si se eligen al azar un paciente enfermo y otro sano, sea mayor el valor de la prueba correspondiente al enfermo [Hanley y McNeil, 1982].

Cuando la curva ROC se genera por el método empírico, independientemente de que haya empates o no, el área puede calcularse mediante la regla trapezoidal, es decir, como la suma de las áreas de todos los rectángulos y trapecios (correspondientes a los empates) que se pueden formar bajo la curva. Estadísticamente, la observación importante, puesto que permite hacer contrastes de significación y dar intervalos de confianza para la verdadera área bajo la curva, es que el área calculada por el método geométrico anterior coincida con el valor del estadístico de suma de rangos de Wilcoxon, W [Bamber, 1975].

Supongamos que tenemos n observaciones aparejadas de la forma (X_i, Y_i) y que las diferencias son $D_i = X_i - Y_i$. Cuando se intenta comprobar si X y Y tienen la misma distribución, frente a la alternativa de que las distribuciones difieren en ubicación, se puede utilizar la prueba de Wilcoxon de rangos con signo para un experimento aparejado. Formalmente, este estadístico no paramétrico es el siguiente:

Hipótesis:

H_0: Las distribuciones poblacionales para las X y las Y son idénticas.

H_a: Las dos distribuciones poblacionales difieren en ubicación.

Estadístico de la prueba:

Se hace $T = mín(T^+, T^-)$, donde T^+ es la suma de los rangos de las diferencias positivas y T^- es la suma de los rangos de las diferencias negativas.

Región de rechazo:

Se rechaza H_0 si $T \leq T_0$, donde T_0 es el valor crítico (valor tabulado).

Conforme a la hipótesis nula de que no hay diferencia entre las X y las Y, se esperaría (en promedio) que la mitad de las diferencias de los pares sean negativas y la otra mitad positivas. Es decir, el número esperado de los pares con diferencias negativas sería $n/2$ (donde n es el número de pares). Además, las diferencias positivas y las diferencias negativas con el mismo valor absoluto deberían ocurrir con la misma probabilidad. Si se ordenan las diferencias según su valor absoluto y se les asigna un rango de menor a mayor, el valor esperado de las sumas de los rangos para las diferencias negativas y positivas sería igual. Diferencias considerables entre las sumas de los rangos asignados a las diferencias positivas y negativas proporcionan evidencia para indicar un desfasamiento en la ubicación de las distribuciones.

Para realizar la prueba de Wilcoxon, se calculan las diferencias D_i para cada uno de los n pares. Se eliminan las diferencias que son iguales a cero y se reduce con ello el número de pares n. Luego se ordenan los valores absolutos de las diferencias asignándole el rango 1 al más pequeño, el rango 2 al siguiente, etc. Si dos o más valores absolutos de las diferencias empatan para un mismo rango, entonces se asigna a cada miembro del conjunto empatado el promedio de los rangos que se habrían asignado a estas diferencias. Luego se calcula la suma de los rangos para las diferencias positivas, T^+ , y también para las diferencias negativas, T^-. Se elige la menor de estas dos cantidades, T, como un estadístico de prueba, para probar la hipótesis nula si T es menor o igual que cierto valor T_0.

Cuando X e Y son dos variables aleatorias independientes cualesquiera, dicho estadístico es conocido precisamente por su uso para contrastar la hipótesis $Pr\{X > Y\} =$

1/2, que en nuestro contexto es la hipótesis nula de que el área sea 1/2, es decir, de que la prueba no sea capaz de discriminar entre los dos grupos. Hanley y McNeil [Hanley y McNeil, 1982] dan fórmulas tanto para el estadístico W como para su error estándar y discuten el problema de la estimación de este último. En general, se suelen dar intervalos de confianza construidos de la manera estándar, *v. g.* al nivel de confianza del 95% intervalos de extremos W ± 1.96*EE(W), siendo EE(W) una estimación del error estándar de W.

Cuando se ajusta un modelo como el binormal empleando técnicas estadísticas, se obtienen, además de las estimaciones de los parámetros que definen la curva ROC, estimaciones del área y de su error estándar, que pueden emplearse para construir intervalos de confianza y efectuar contrastes de significación como en el caso no paramétrico.

Cuando el número de empates es elevado, como ocurre cuando se emplean datos de clasificación, el estadístico W (el área calculada por el método empírico) proporciona un estimador sesgado de la verdadera área, lo que hace recomendable emplear un método distinto, por ejemplo uno basado en un método paramétrico.

3.7.3.2 Área parcial.

Existen situaciones en las que las propias características ventajosas del área se conviertan en un inconveniente para su uso clínico. El área puede interpretarse como un promedio de la sensibilidad sobre todos los valores posibles de especificidad. Puede que clínicamente sólo interese los puntos de la curva ROC que aseguren altos valores de sensibilidad o especificidad. Un caso típico es el de las mamografías en programas de detección precoz del cáncer, donde debe asegurarse una alta sensibilidad de la prueba. Se han propuesto índices de área parcial que pueden ser empleados para evaluar la exactitud restringida a los puntos de operación de interés de la curva ROC [Jiang et al., 1996], [McClish, 1989].

3.7.3.3 Comparación de dos pruebas.

Cuando se dispone de dos (o más) pruebas para abordar el diagnóstico de un mismo problema clínico, el cálculo del área brinda un método conveniente para comparar

globalmente su exactitud diagnóstica relativa. En principio, al comparar dos pruebas se prefiere la que tenga mayor área, por ser la de mayor exactitud diagnóstica de las dos.

Desde un punto de vista estadístico, el problema es valorar si la diferencia observada entre las áreas calculadas para dos pruebas distintas es debida a la variabilidad inherente al muestreo o es más bien atribuible a una diferencia real en la exactitud de ambas pruebas. Podemos enunciar este problema como un contraste de la hipótesis nula de igualdad de las dos áreas, que denotaremos por ABC_A y ABC_B, frente a una alternativa bilateral. En general, se dispone de los valores para las dos pruebas en una única muestra de pacientes. El contraste que se usa frecuentemente es el debido a Hanley y McNeil [Hanley y McNeil, 1983], que podemos considerar representativo de los desarrollados en esta situación. Se utiliza como estadístico del contraste:

$$z = \frac{(ABC_A - ABC_B)}{\sqrt{EE_A^2 + EE_B^2 - 2REE_AEE_B}} \tag{3.53}$$

siendo ABC el área observada, EE el error estándar del ABC y R la correlación entre ABC_A y ABC_B. Al nivel de significación α se rechaza la hipótesis nula cuando $|z| > z\alpha_{/2}$, siendo $z\alpha_{/2}$ el cuantil de orden $1-\alpha/2$ de una distribución normal estándar, *v. g.* si $\alpha = 0.05$ es $z\alpha_{/2} = 1.96$.

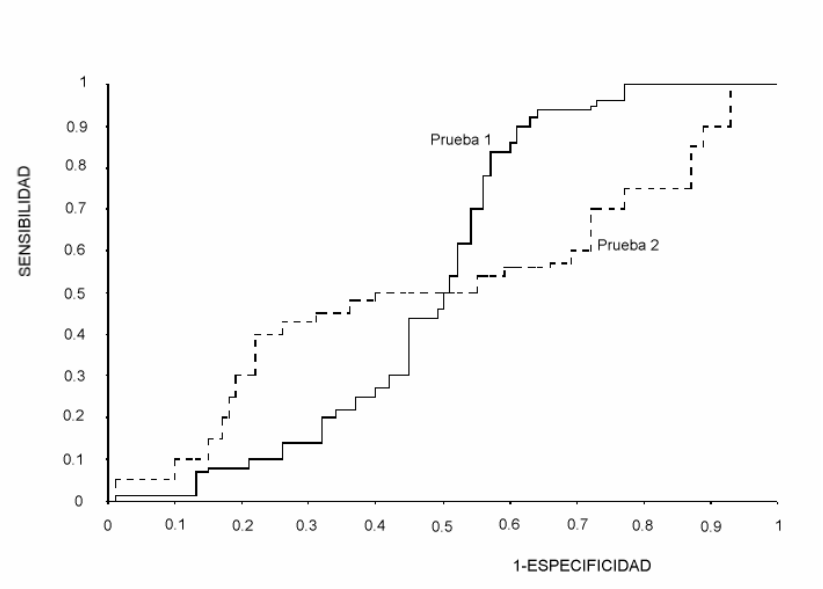


Figura 3.19. Curvas ROC empíricas de dos pruebas diagnósticas distintas.

La comparación entre dos pruebas no debe depender exclusivamente en contrastes como el anterior. Pueden existir dos pruebas con sendas curvas ROC muy distintas de forma, hecho que puede tener importantes implicaciones prácticas, y que, sin embargo, sean prácticamente iguales respecto a su área, como se observa en la figura 3.19. El empleo del área parcial puede permitir manejar correctamente estas situaciones. En cualquier caso, es evidente que nunca se debe prescindir de un examen visual detenido de un gráfico que muestre simultáneamente ambas curvas ROC.

3.7.3.4 Elección del valor de corte.

El empleo en la práctica médica de una prueba diagnóstica exige la elección de un valor de corte. Para ello es imprescindible un conocimiento detallado de los riesgos y beneficios de las decisiones médicas derivadas del resultado de la prueba. Un enfoque sencillo aparece en [Zweig y Campbell, 1993], [McNeil et al., 1975], donde la elección del valor de corte se basa en la importancia relativa que para el paciente tenga hacer un diagnóstico falso positivo o falso negativo. El diagrama de la decisión es el representado en la figura 3.20.

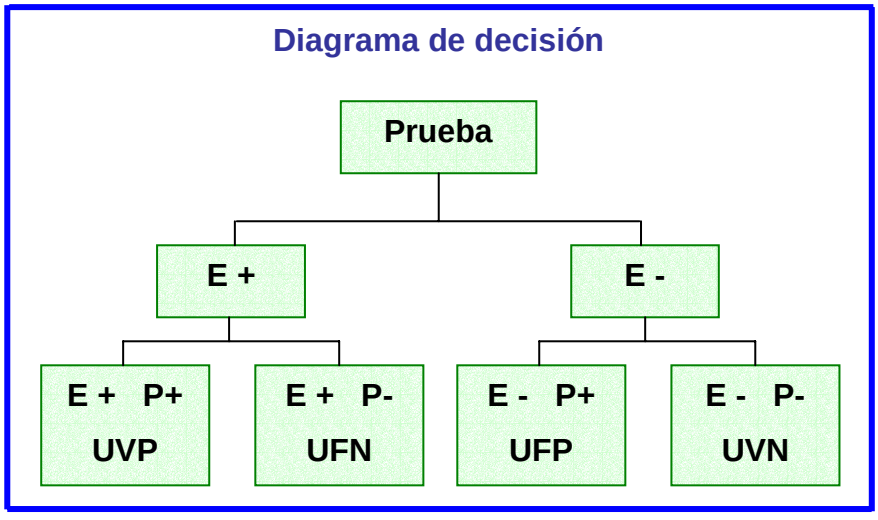


Figura 3.20. Diagrama de decisión de la elección del punto de corte más apropiado para un paciente.

Las curvas ROC facilitan la posibilidad de seleccionar los niveles de decisión (puntos de corte) considerando el coste de los resultados falsos positivos y falsos negativos (en términos clínicos, económicos o de trauma psicológico para los pacientes, por ejemplo).

Si el coste de un falso positivo y negativo no difieren, el criterio elegido será maximizar la suma de la sensibilidad y especificidad, es decir, maximizar la proporción de clasificaciones correctas. En este caso el punto de corte óptimo es el más próximo al ángulo superior izquierdo del gráfico. Si el coste de un falso positivo y negativo difieren, se deberá tener en cuenta este distinto coste para el cálculo del punto óptimo de corte.

Representamos por *U* la *utilidad* y es un valor normalizado que dependerá de las consecuencias de la decisión y de las preferencias del paciente. La utilidad esperada de la prueba es el promedio ponderado, por las respectivas probabilidades, de las diferentes utilidades, como se muestra en la expresión 3.54.

$$\begin{aligned} U = {} & p(E)\cdot p(P+\mid E)\cdot UVP + p(E)\cdot p(P-\mid E)\cdot UFN + \\ & + p(\overline{E})\cdot p(P+\mid \overline{E})\cdot UFP + p(\overline{E})\cdot p(P-\mid \overline{E})\cdot UVN \end{aligned} \tag{3.54}$$

La expresión 3.54 que se puede escribir como:

$$\begin{aligned} U = {} & (UVP-UFN)\cdot p(E)\cdot p(P+\mid E) + (UFN-UVN)\cdot p(\overline{E})\cdot p(P+\mid \overline{E}) + \\ & + UFN\cdot p(E) + UVN\cdot p(\overline{E})\cdot \end{aligned} \tag{3.55}$$

La curva ROC describe la relación entre: $p(P+|E)$ y $p(P+|\overline{E})$. Se trata de elegir un punto de esa curva que maximice la función de utilidad. Para ello hay que resolver la ecuación que resulta de igualar a 0 la derivada de la utilidad respecto a $p(P+|\overline{E})$. El resultado es:

$$\frac{d\quad p(P+\mid E)}{d\quad p(P+\mid \overline{E})} = \frac{UVN-UFP}{UVP-UFN}\times\frac{p(\overline{E})}{p(E)} \tag{3.56}$$

El primer miembro de la igualdad es la pendiente de la curva ROC. La ecuación nos da un criterio para elegir el punto de corte: la pendiente en este punto debe ser la de la expresión. *UVN-UFP* es la diferencia en beneficio entre no tratar a *VN* y tratar *FP*. Habitualmente se denomina *coste neto* (C) de tratar pacientes no enfermos. *UVP-UFN*

es la diferencia en beneficio entre tratar a *VP* y no tratar *FN*. Habitualmente se denomina *beneficio neto* (B) de tratar sujetos enfermos.

La pendiente de la curva en cada punto se puede estimar ajustando los puntos a una curva y calculando la pendiente. En el caso en el que se analizan polígonos en lugar de curvas ROC (lo más frecuente), se hace calculando el cociente entre el cambio de la sensibilidad y el cambio de la especificidad para cada tramo (marcados en verde y en rojo el primer y segundo tramo respectivamente en la figura 3.21) y asignando a cada punto como pendiente el promedio de los tramos respectivos.

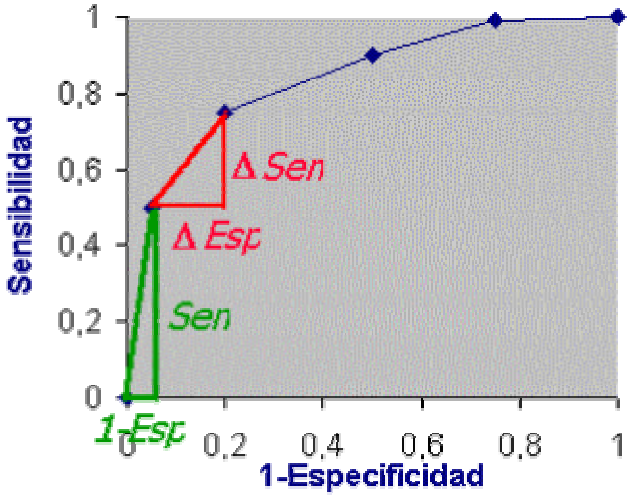


Figura 3.21. Cálculo de las pendientes de cada tramo en una curva ROC expresada por medio de polígonos.

El mejor punto de corte será aquel cuyo valor de pendiente sea lo más parecido al valor de $\frac{d\ \ p(P+|E)}{d\ \ p(P+|\overline{E})}$.

En la mayoría de los casos, calcular un valor de corte óptimo es una tarea muy complicada. De hecho, es un problema que se aborda más adecuadamente con otras herramientas más complejas disponibles en el seno del Análisis de Decisiones Clínicas [Weinstein y Fineberg, 1980], [Beck y Schultz, 1986].

Capítulo 4
Algoritmos Genéticos.

4.1 Introducción.

Las técnicas de optimización han llegado a ser un punto muy importante en el diseño de sistemas. Para la utilización de cualquier sistema se necesita previamente realizar una selección de su estructura y una intensiva optimización de sus parámetros. Normalmente hay que explorar un número significativo de opciones o alternativas antes, para decidir cuál es la mejor de ellas. En el proceso de desarrollo es muy frecuente encontrar criterios de diseño que entran en conflicto.

Las distintas técnicas de optimización existentes se clasifican en función de su habilidad general de optimización. Se puede distinguir entre técnicas de optimización de un punto y de múltiples puntos. Esta taxonomía tiene en cuenta la manera en que se realiza la optimización. Aceptando que los términos de “optimización” y “búsqueda a través de un espacio de optimización” pueden ser intercambiables, podemos describir la optimización de un punto como el proceso de búsqueda basado en la posición actual en el espacio de búsqueda y en el marco dado por un criterio de optimización. La guía necesaria para avanzar en el proceso de búsqueda se puede basar, por ejemplo, en el gradiente del criterio de optimización. La otra aproximación explota algunas

características probabilísticas del espacio de búsqueda dando lugar a un gran número de métodos también llamados métodos de búsqueda probabilísticos.

Los métodos de optimización basados en el gradiente presentan una serie de ventajas e inconvenientes. Cuando la búsqueda no da buenos resultados, implica que el camino seguido termina en un mínimo local (en el mejor de los casos) o que es un camino erróneo completamente. Una mejora de los métodos de optimización de un punto consiste en considerar un conjunto de puntos de búsqueda (población) y permitir una búsqueda colectiva del valor óptimo. La optimización basada en la población se apoya en una familia de puntos de búsqueda distribuidos en el espacio de búsqueda y tiene en cuenta la interacción y el paso de mensajes entre los elementos individuales de la población. Esta filosofía de búsqueda mediante poblaciones dio lugar a la computación evolutiva y a los algoritmos genéticos (en inglés, *genetic algorithm – GA*) [Michalewicz, 1992], [Davis, 1991], [Goldberg, 1989].

Los algoritmos genéticos constituyen una aproximación al aprendizaje inspirada en la evolución natural. Estos procedimientos se basan en ideas parecidas, en ciertos aspectos, a las de *individuos*, *cruce, recombinación de cromosomas, mutación genética, adaptación* y *selección natural*.

4.2 Los algoritmos genéticos dentro de la teoría del aprendizaje automático.

Vamos a situar los algoritmos genéticos dentro de la teoría del aprendizaje automático. Se entiende como "aprendizaje automático" la capacidad de un programa de computadora de aumentar su eficiencia a través de la experiencia. De un modo más formal, se dice que un programa "aprende" una serie de tareas T si la medida de su eficiencia P aumenta con la experiencia E. Por lo tanto, para definir correctamente un problema de aprendizaje, es necesario identificar la clase de tareas, el tipo de medida de eficiencia y la fuente de la experiencia.

Los algoritmos de aprendizaje automático han sido utilizados en una gran variedad de dominios de aplicaciones. Son especialmente útiles en:

a) Problemas de *data mining*, donde se trata de descubrir automáticamente irregularidades implícitas existentes en bases de datos muy grandes.

b) Dominios difícilmente comprensibles, donde los humanos no tienen el conocimiento necesario para desarrollar algoritmos efectivos.

c) Dominios donde el programa tenga que adaptarse dinámicamente a las condiciones cambiantes del entorno.

El aprendizaje automático incluye aspectos de distintas disciplinas: inteligencia artificial, probabilidad y estadística, complejidad computacional, teoría de la información, sicología y neurobiología, teoría de control y filosofía.

El problema central del aprendizaje es inducir funciones generales de ejemplos específicos de entrenamiento. El aprendizaje de conceptos se basa en adquirir la definición de una categoría general a partir de una muestra dada de ejemplos de entrenamiento positivos y negativos de esa categoría. Se puede formular como un problema de búsqueda de la hipótesis que mejor encaja con los ejemplos de entrenamiento a través de un espacio predefinido de hipótesis potenciales.

Gran parte del aprendizaje implica adquirir conceptos generales de muestras de entrenamiento específicas. Cada concepto describe un conjunto de subconceptos o eventos definidos sobre un conjunto mayor (por ejemplo, el subconjunto "pájaros" del conjunto total "animales"). Alternativamente, cada concepto se puede considerar como una función booleana valuada definida sobre ese conjunto (por ejemplo, una función definida sobre todos los "animales", cuyo valor es "cierto" para elementos del subconjunto "pájaros" y falso para otro tipo de animales).

Cuando se trata de aprender un concepto objetivo, se presenta al sistema que aprende un conjunto de ejemplos de entrenamiento, cada uno consistente en una muestra x de X, junto con su valor de concepto $c(x)$. Las muestras para las que $c(x) = 1$, son *ejemplos positivos* y las muestras para las que $c(x) = 0$, son *ejemplos negativos*, o muestras no representativas del concepto objetivo. Se utiliza el par ordenado $\langle x, c(x) \rangle$ para describir el ejemplo de entrenamiento compuesto por la muestra x y su valor de concepto objetivo $c(x)$. El símbolo D denota el conjunto de ejemplos de entrenamiento disponibles.

Dado un conjunto de ejemplos de entrenamiento D del concepto objetivo c, el problema que debe resolver el sistema aprendiz es la estimación de c. El símbolo H denota el conjunto de todas las posibles hipótesis que el aprendiz debe considerar. Normalmente, H viene determinado por el diseñador, que elige la representación de las hipótesis. En general, cada hipótesis h de H representa una función booleana definida sobre X, esto es, $h : X \rightarrow \{0,1\}$. El objetivo del sistema que aprende es encontrar una hipótesis h tal que $h(x) = c(x)$ para todo x en X.

La única información de la que dispone el sistema aprendiz es la suministrada por los ejemplos de entrenamiento. Por lo tanto, los algoritmos de aprendizaje inductivo pueden ser la mejor garantía de que la hipótesis final resultante estime adecuadamente el concepto objetivo sobre todos los datos de entrenamiento. Se asume que la mejor hipótesis es aquella que mejor haga esto. Bajo este supuesto, se formula la *hipótesis del aprendizaje inductivo*: cualquier hipótesis encontrada que aproxima bien la función objetivo sobre un conjunto de ejemplos de entrenamiento suficientemente grande, aproximará también correctamente la función objetivo sobre ejemplos no vistos. Es muy importante que el conjunto de ejemplos elegidos para realizar el entrenamiento represente bien la distribución de los ejemplos sobre los que la eficiencia final del sistema P será medida. En general, el aprendizaje es mejor si los ejemplos de entrenamiento siguen una distribución similar a la de los ejemplos de test futuros. En la práctica, en muchas situaciones el sistema debe aprender de una distribución de ejemplos distinta en algún modo de los ejemplos que se utilizarán para evaluarlo. Esto puede resultar problemático: que el sistema tenga una buena eficiencia sobre una distribución determinada de ejemplos no implica que presente una eficiencia tan buena sobre otra distribución distinta.

El aprendizaje de conceptos puede ser visto como una tarea de búsqueda en el amplio espacio de hipótesis. El objetivo de esta búsqueda es encontrar la hipótesis que mejor se ajusta a los ejemplos de entrenamiento. Al seleccionar una representación para las hipótesis, el diseñador del algoritmo de aprendizaje define implícitamente el espacio de todas las hipótesis que el programa puede aprender. La mayor parte de las tareas de aprendizaje implican espacios de hipótesis infinitos.

En los algoritmos genéticos, las hipótesis son los *individuos*, y se describen a menudo por cadenas de bits cuya interpretación depende de la aplicación, aunque también pueden ser descritas por expresiones simbólicas o incluso, por programas. La

búsqueda de la hipótesis apropiada comienza con una *población* o colección de hipótesis iniciales. Aplicando sobre los miembros de la población actual operadores basados en fenómenos de evolución biológica, como la mutación aleatoria y el cruce, se crean nuevos miembros que se introducen en la población, dando lugar a una nueva generación. En cada iteración del algoritmo, las hipótesis de la población actual se evalúan en relación a una medida dada de aptitud o *fitness*, y se eligen probabilísticamente las hipótesis de la población con mejor valor de aptitud como semillas para producir la próxima generación, emulando la *selección natural*. Más que buscar hipótesis desde más generales a más específicas o desde más simples a más complejas, los GAs generan hipótesis sucesoras por medio de la aplicación repetida de la mutación y la recombinación sobre partes de las mejores hipótesis conocidas hasta ese momento.

Los algoritmos genéticos han sido aplicados exitosamente en una gran variedad de tareas de aprendizaje y en otros problemas de optimización. La popularidad de los GAs se debe a diversos factores:

- La evolución es un método de adaptación robusto dentro de los sistemas biológicos.

- Los GAs pueden realizar búsquedas de espacios de hipótesis que contengan partes complejas que interactúan, donde el impacto de cada parte sobre la aptitud global puede resultar difícil de modelar.

- Los algoritmos genéticos son fácilmente paralelizables, permiten aprovechar de este modo la potencia del hardware para disminuir costes computacionales.

John Holland y David Goldberg son unos de los pioneros de los trabajos modernos sobre los algoritmos de adaptación natural, y han contribuido en numerosas ocasiones en esta bibliografía específica con sus trabajos [Goldberg, 1989]. Una de las primeras aplicaciones de los algoritmos genéticos en el campo del reconocimiento de patrones, que es el campo en que se aplican los algoritmos genéticos en esta tesis, fue [Cavicchio, 1970]. En este trabajo, Cavicchio emplea algoritmos genéticos para diseñar un conjunto de detectores que serán utilizados por una máquina de reconocimiento de

patrones. En [Fitzpatrick et al., 1984], [Grefenstette y Fitzpatrick, 1985] podemos encontrar una aplicación de los algoritmos genéticos en un sistema de procesamiento de imágenes médicas. El objetivo era buscar con algoritmos genéticos los coeficientes de la transformación bilineal que debe ser aplicada sobre una imagen médica para alinearla con otra. Otros trabajos de interés en el campo del tratamiento de imágenes y reconocimiento de patrones se pueden encontrar en [Gillies, 1985], [Englander, 1985], [Stadnyk, 1987]. También se pueden citar, como primeras aplicaciones de algoritmos genéticos en muchos otros campos, las siguientes contribuciones: en biología [Rosenberg, 1970a], [Rosenberg, 1970b], en informática [Bagley, 1967], [Raghavan y Bichard, 1979], en ingeniería [Davis y Smith, 1985], en física [Shaefer, 1985], en ciencias sociales [Smith y De Jong, 1981], etc.

4.3 Robustez de los métodos de optimización y búsqueda tradicionales. Comparación con los algoritmos genéticos.

Existen tres tipos de métodos tradicionales de búsqueda: basados en cálculo, enumerativos y aleatorios. Cada uno presenta una serie de ventajas e inconvenientes, como se comenta a continuación.

Los métodos basados en cálculo han sido estudiados con mucha profundidad. Se subdividen en dos clases principales: indirectos y directos. Los métodos indirectos buscan extremos locales resolviendo el conjunto no lineal de ecuaciones que resultan de igualar el gradiente de la función objetivo a cero. Dada una función suave y sin restricciones, para encontrar un posible pico se comienza por restringir la búsqueda a aquellos puntos con pendiente igual a cero en todas las direcciones. Por otro lado, los métodos directos buscan óptimos locales moviéndose en las direcciones relacionadas con el gradiente local. Es lo que se conoce por “subir colinas” (en inglés, *high climbing*): subir por la función en la dirección que permita más recorrido.

Este conjunto de métodos tienen un alcance local, los óptimos buscados son los mejores en un vecindario del punto actual. Por lo tanto, si existen varios picos, el método se puede quedar en picos más bajos, que no sean los más óptimos. Además, estos métodos dependen de la existencia de derivadas (valores de pendientes bien

definidos). Incluso permitiendo aproximaciones numéricas a estas derivadas, esto representa un gran inconveniente, y limita el uso de estos métodos a un dominio pequeño de problemas. En la práctica, muchos espacios de parámetros no implican suavidad y derivadas bien definidas. Están llenos de discontinuidades y ruido.

Los esquemas enumerativos son algoritmos que trabajan en un espacio de búsqueda finito, o bien, en un espacio de búsqueda infinito discretizado, y analizan los valores de la función objetivo en cada punto del espacio de uno en uno. El método es sencillo, y representa un tipo muy humano de búsqueda cuando el número de posibilidades es pequeño, pero en general presenta una baja eficiencia, ya que muchos espacios de búsqueda son demasiado grandes para ser recorridos punto por punto.

Los algoritmos de búsqueda aleatoria buscan de modo aleatorio los mejores esquemas según un criterio de eficiencia. Pero, en ejecuciones largas, no son mejores que los métodos enumerativos.

Por todo esto, se deduce que los métodos tradicionales de búsqueda no son robustos. Esto no implica que no sean útiles, ya que han sido aplicados con éxito en diversas aplicaciones. Pero, cuando el problema de búsqueda es muy complejo, como ocurre en el problema abarcado en esta tesis, se hace necesario recurrir a métodos con filosofías de búsqueda diferentes.

Los algoritmos genéticos, GAs, son distintos de los métodos tradicionales de búsqueda principalmente por cuatro razones:

- Los GAs trabajan con un conjunto de parámetros codificado, no con los parámetros en sí mismos. El conjunto natural de parámetros del problema de optimización se codifica con algún alfabeto finito como cadenas de longitud finita.

- Los GAs realizan búsquedas partiendo de una población de puntos, no de un único punto. Los métodos tradicionales parten de un único punto, y siguiendo alguna regla de transición, determinan cual debe ser el siguiente punto al que deben ir. Estos métodos punto a punto son peligrosos porque suelen localizar falsos picos en espacios de búsqueda multi-modales. En cambio, los GAs trabajan con una amplia base de puntos simultáneamente (una población de cadenas), escalando varios picos en paralelo, por lo que presentan una probabilidad menor de encontrar un falso pico.

- Los GAs utilizan información de la función objetivo, no hacen uso de las derivadas de la misma u otro tipo de información adicional. La mayoría de las técnicas de búsqueda requieren mucha información auxiliar para trabajar adecuadamente. Por ejemplo, en el caso de las técnicas basadas en el gradiente, se necesitan las derivadas (calculadas analítica o numéricamente) para poder escalar picos. Los GAs son ciegos en este sentido: no necesitan información auxiliar. Para realizar una búsqueda efectiva de estructuras mejores solamente requieren los valores de la función objetivo asociados a los individuos (cadenas) con las que trabaja.

- Los GAs emplean reglas de transición probabilísticas, no deterministas, para guiar su búsqueda. No son métodos de búsqueda aleatorios, utilizan la probabilidad para moverse a regiones del espacio de búsqueda que parecen mejores que otras.

Estas cuatro propiedades hacen que los GAs sean métodos de búsqueda más robustos y con mejor respuesta que otros métodos de búsqueda más comunes.

4.4 Funcionamiento básico de los algoritmos genéticos.

Los algoritmos genéticos buscan un espacio de hipótesis candidatas para identificar la mejor hipótesis. En GAs, la "mejor hipótesis" se define como aquella que optimiza una medida numérica predefinida para un determinado problema, llamada *aptitud* de la hipótesis. Por ejemplo, si la tarea de aprendizaje es el problema de aproximar una función desconocida dados unos ejemplos de entrenamiento de sus entradas y salidas, entonces la aptitud se podría definir como la precisión de las hipótesis sobre estos datos de entrenamiento. Si la tarea es aprender estrategias para jugar al ajedrez, la aptitud podría definirse como el número de juegos ganados por un individuo cuando juega en contra de otros individuos de la población actual.

Aunque las implementaciones de los algoritmos genéticos varían en sus detalles, suelen compartir la siguiente estructura: el algoritmo opera en iteraciones, actualizando en cada una un conjunto de hipótesis, llamada *población*. Estas hipótesis son los *individuos*, y se codifican en cadenas finitas con un alfabeto finito. En cada iteración, todos los miembros de la población se evalúan según una función de aptitud. Luego se genera una nueva población seleccionando probabilísticamente los individuos con mejor

valor de aptitud de la población actual. Algunos de estos individuos pasan intactos a la siguiente población generada. Otros se utilizan para crear nuevos individuos sucesores por medio de la aplicación de operaciones genéticas, como por ejemplo, la mutación y el cruce. Las entradas a este algoritmo son la función de aptitud para hacer un ranking de las hipótesis candidatas, un umbral que define un nivel de aptitud aceptable para finalizar el algoritmo, el tamaño de la población a mantenerse y los parámetros que determinan cómo generar las poblaciones sucesoras: la fracción de la población que se reemplazará y el ritmo de mutación.

Sea $P(t)$ la población de cadenas (individuos) en un instante de la evolución t. El esquema básico del algoritmo es el siguiente:

Comienzo:
    iteración = 0
        Inicialización de la población P(0)
        Evaluación de la población P(0)
    mientras (no se cumple el criterio de parada) hacer:
    comienzo:
        iteración = iteración +1
        Selección P(iteración) de P(iteración – 1)
        Alteración P(iteración)
        Evaluación P(iteración)
    fin
fin

El algoritmo produce en cada iteración una nueva generación de hipótesis (individuos) basadas en la población actual. Para ello, primero se selecciona un cierto número de hipótesis de la población actual, que se incluirán en la siguiente generación. El criterio más común para llevar a cabo esta selección de individuos se basa en la selección natural, donde los individuos más fuertes son los que sobreviven normalmente. Por lo tanto, en el proceso artificial, esto se imita intentando que los individuos que tengan mayor valor de aptitud tengan más posibilidades de sobrevivir. El método más común de selección es el de la ruleta. Con este método las hipótesis se

seleccionan probabilísticamente, donde la probabilidad de seleccionar la hipótesis h_i viene dada por:

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)} \tag{4.1}$$

Por lo tanto, la probabilidad de que una hipótesis sea seleccionada es proporcional a su propia aptitud e inversamente proporcional a la suma de la aptitud de las otras hipótesis competidoras en la población actual.

Este experimento se corresponde a construir una ruleta cuyos sectores reflejan la probabilidad de selección de cada cadena, tal y como se indica en la figura 4.1. Moviendo la ruleta *N* veces, se observa que las cadenas con un valor de aptitud pequeño se eligen en muy pocas ocasiones, mientras que las cadenas con altos valores de aptitud (que están representadas por sectores mayores en la ruleta) se escogen con mayor frecuencia. Los individuos seleccionados pasarán a la población en la siguiente iteración, es decir, sobrevivirán.

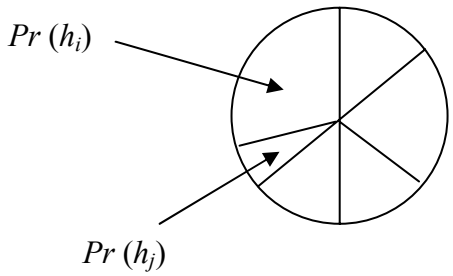


Figura 4.1. Método de selección de la ruleta.

Una vez que se han seleccionado estos miembros de la generación actual para ser incluidos en la siguiente generación, se generan miembros adicionales usando una operación de cruce. El cruce toma dos hipótesis padres elegidas probabilísticamente de la generación actual y crea dos hipótesis descendientes recombinando porciones de ambos padres. Una vez creados estos nuevos miembros por medio de cruce, la población de la nueva generación contiene el número deseado de miembros. Llegados a este punto, una cierta fracción *m* de estos miembros se eligen al azar y se alteran por medio de operaciones aleatorias de mutación. Estos operadores genéticos serán descritos detalladamente en secciones posteriores.

Una cuestión importante es saber cuántos individuos deben existir en la población. Si el número es muy bajo, pronto todos los individuos tendrán rasgos idénticos y la recombinación no hará nada; si el número es muy alto, el tiempo de cálculo será innecesariamente excesivo.

Las soluciones codificadas en un individuo compiten para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El ambiente, constituido por los otros individuos, ejercerá una presión selectiva sobre la población, de forma que sólo los mejor adaptados (aquellos que resuelvan mejor el problema) sobrevivan o leguen su material genético a las siguientes generaciones, igual que en la evolución de las especies.

La nomenclatura utilizada en los algoritmos genéticos es muy análoga a la empleada en la evolución biológica. Las cadenas (individuos) de los sistemas genéticos artificiales son análogas a los *cromosomas* en los sistemas biológicos. En los sistemas naturales, uno o más cromosomas se combinan para formar la constitución genética de un organismo y el paquete genético total recibe el nombre de *genotipo*. En los sistemas genéticos artificiales el paquete total de cadenas se llama *estructura*. En los sistemas naturales, el organismo formado por la interacción de todo el paquete genético con su entorno es el *fenotipo*, mientras que en los sistemas artificiales, las estructuras se decodifican para formar un determinado *conjunto de parámetros*, *solución alternativa* o *punto* (en el espacio de soluciones).

En la terminología natural, los cromosomas están compuestos por *genes*, los cuales toman unos determinados valores llamados *alelos*. En genética, la posición de un gen se identifica separadamente de la función del gen. Así, por ejemplo, se puede hablar de un gen que especifica el color de los ojos, su localización está en la posición 10 y su valor de alelo puede ser “ojos azules”. En los sistemas genéticos artificiales, se dice que las cadenas están compuestas por *características* o *detectores*, las cuales toman distintos *valores*. Las características pueden estar localizadas en distintas posiciones dentro de la cadena.

4.5 Relación entre la función objetivo y la función de aptitud.

En un sistema de aprendizaje automático es muy importante determinar exactamente qué tipo de conocimiento será aprendido y cómo este conocimiento será usado por el programa que calcula la eficiencia. El tipo de información que debe aprender el sistema es una función que elige la mejor hipótesis entre todas las posibles, para un problema específico. Esta función recibe el nombre de *función objetivo*. Suele ser muy difícil conseguir que el sistema aprenda la función objetivo ideal V. En la práctica, se espera que los algoritmos de aprendizaje obtengan una aproximación $\hat{V}$ de la función objetivo, y por este motivo el proceso de aprendizaje de la función objetivo es frecuentemente denominado *aproximación funcional*.

Una vez especificada la función objetivo ideal V, se debe elegir la representación que el programa de aprendizaje utilizará para describir la función $\hat{V}$ que aprenderá. Para esto existen distintas opciones: se puede representar $\hat{V}$ utilizando tablas, conjuntos de reglas, funciones polinómicas cuadráticas, redes neuronales, etc. Es deseable elegir representaciones muy expresivas que permitan la aproximación más cercana posible a la función objetivo ideal V. Pero se debe tener en cuenta el hecho de que cuanto más expresiva es la representación, más datos de entrenamiento necesita el sistema para elegir entre las hipótesis que se pueden presentar.

El algoritmo de aprendizaje debe realizar la aproximación de la función objetivo. Lo más común es que este algoritmo utilice los ejemplos del entrenamiento b y los valores de entrenamiento de estos ejemplos (valores de la función objetivo) $V_{entrena}(b)$ para definir la mejor hipótesis o conjunto de pesos que minimice el error entre los valores de entrenamiento y los valores que predice la hipótesis $\hat{V}$:

$$E \equiv \sum_{(b, V_{entrena}(b)) \in \text{ejemplos entrenamiento}} (V_{entrena}(b) - \hat{V}(b))^2 \qquad (4.2)$$

La función de aptitud mide este error y define el criterio según el cual se construye un ranking de las hipótesis potenciales. Este ranking influye en el proceso de selección probabilística de estas hipótesis para ser incluidas en la siguiente generación.

Si la tarea consiste en aprender reglas de clasificación, la función de aptitud tiene normalmente una componente que puntúa la precisión de la clasificación de la regla sobre unos ejemplos de entrenamiento dados. A menudo, dependiendo de la aplicación, se tienen en cuenta otros criterios, como por ejemplo la complejidad o generalidad de la regla. Desde un punto de vista más general, cuando las hipótesis expresadas como cadenas se interpretan como procedimientos complejos (por ejemplo, cuando la cadena representa una colección de reglas del tipo *si-entonces*), la función de aptitud puede medir la eficiencia global del procedimiento resultante en vez de la eficiencia de reglas individuales.

En muchos problemas, el objetivo consiste en minimizar una función de coste $g(x)$ o en maximizar una función de utilidad $u(x)$. Incluso en los problemas en los que el objetivo es maximizar una función, esto no garantiza que la función de utilidad sea positiva para todos los posibles valores de x, tal y como se requiere en una función de aptitud. Por lo tanto, a menudo es necesario hacer una correspondencia en la función objetivo natural para convertirla en una función de aptitud adecuada.

La dualidad de minimización de costes y maximización de utilidad es muy común. Para transformar un problema de minimización en uno de maximización, simplemente se multiplica la función de coste por menos uno. En algoritmos genéticos, esta operación es insuficiente porque la medida así obtenida no garantiza ser positiva para todas las muestras. Con GAs, la transformación coste a aptitud normalmente utilizada es la siguiente:

$$f(x) = \begin{cases} C_{\max} - g(x) & si \quad g(x) < C_{\max} \\ 0 & en \quad otro \quad caso \end{cases} \tag{4.3}$$

El coeficiente C_{max} se puede elegir de diversas formas. Puede ser el mayor valor de g observado, el mayor valor de g en la población actual, o el mayor en las últimas k iteraciones del algoritmo. También podría depender de la varianza de la población.

Cuando la función objetivo natural es una función de utilidad, no hay dificultad con la dirección de la función: las mayores utilidades se acercan más al comportamiento deseado. Podría haber problemas si la función de utilidad $u(x)$ tuviera valores negativos. En este caso, se podría aplicar la siguiente transformación:

$$f(x) = \begin{cases} u(x) + C_{\min} & si \quad u(x) + C_{\min} > 0 \\ 0 & en \quad otro \quad caso \end{cases} \tag{4.4}$$

El coeficiente C_{min} puede ser el valor absoluto del peor valor de u en las últimas k generaciones o en la generación actual, o puede ser una función de la varianza de la población.

Existe una relación muy importante entre la función objetivo y la función de aptitud. En los algoritmos genéticos se puede regular el nivel de competición entre los miembros de la población por medio del escalado de la aptitud. Regular el número de copias es especialmente importante cuando se trabaja con poblaciones pequeñas. En el comienzo del algoritmo encontraremos algunos individuos extraordinarios rodeados de colegas mediocres. Aplicando el método de selección más común ($Pselección_i = f_i/\sum f$), los individuos más extraordinarios constituirán una proporción muy significativa de la población en la siguiente generación, y esto no es deseable, ya que origina una convergencia prematura. Con el desarrollo del algoritmo, la media de aptitud de la población será muy parecida al mejor valor de aptitud de la misma, lo que hace que los miembros medios y los mejores obtengan el mismo número de copias en futuras generaciones. Estas dos situaciones hacen que la búsqueda sea mediocre. Para intentar solventar estos problemas se utiliza el escalado de la aptitud.

El escalado sirve para prevenir la dominación temprana sobre el proceso de selección de individuos extraordinarios, y posteriormente, provoca una competición alta entre individuos igualados, acentuando sus diferencias.

Los mecanismos de escalado de aptitud se iniciaron con los trabajos de [Bagley, 1967], [Rosenberg, 1970a], [Rosenberg, 1970b]. Entre estos métodos, podemos destacar tres: el escalado lineal, el truncamiento sigma (σ) y el escalado por ley de potencias.

Sea f la aptitud original y f' la aptitud escalada. El escalado lineal requiere una relación lineal entre f' y f como la siguiente:

$$f' = af + b \tag{4.5}$$

Los coeficientes a y b se pueden elegir de muchas formas, sin embargo, en todos los casos, el objetivo buscado al fijarlos es conseguir igualar la media de la aptitud escalada, f'_{media}, a la media de la aptitud original, f_{media}, ya que así sucesivos usos del

procedimiento de selección asegurarán que cada miembro medio de la población contribuya con un descendiente en la siguiente generación, y obligar al máximo valor de aptitud escalada a ser un múltiplo especificado (normalmente 2) de la media de aptitud. En la figura 4.2 se muestra un ejemplo de escalado lineal.

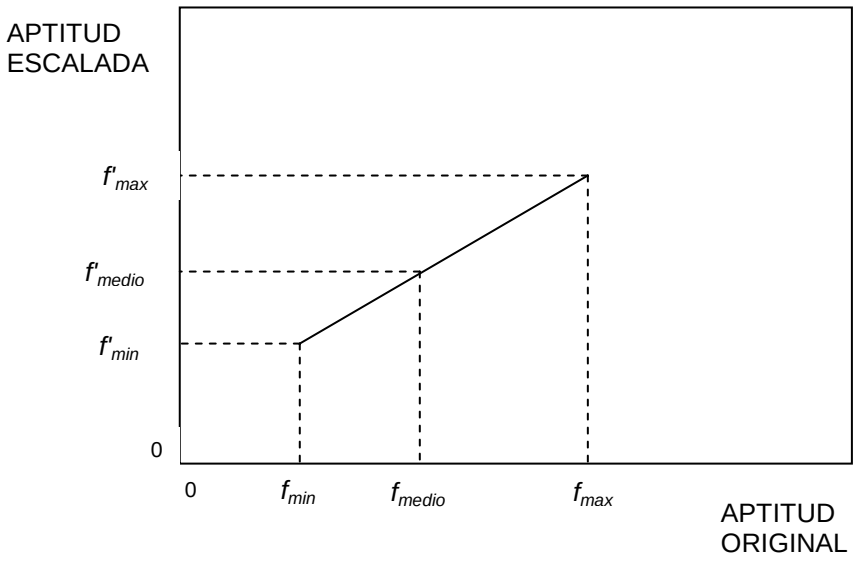


Figura 4.2. Escalado lineal de la aptitud.

Para controlar el número de descendientes correspondientes a los elementos con mayor aptitud original, se puede elegir otro método de escalado que obtiene una aptitud máxima escalada:

$$f'_{max} = C_{mult} \cdot f_{media} \tag{4.6}$$

donde C_{mult} es el número de copias esperadas que se desea hacer corresponder a los mejores individuos de la población. En principio, parece no haber problema para aplicar la regla de escalado lineal, ya que los individuos más extraordinarios se escalan hacia abajo y los individuos con valor de aptitud más bajo se escalan hacia arriba. Pero sí existe un problema, cuando varios individuos malos están muy por debajo de la media de la población y del máximo, y media y máximo a su vez están muy cercanos. En esta situación, el escalado lineal podrá dar lugar a valores negativos de aptitud, tal y como se muestra en la figura 4.3. Hay varias soluciones a este problema, por ejemplo, mantener la igualdad entre las medias de la aptitud original y escalada, y hacer corresponder el mínimo valor de aptitud original f_{min} al valor de aptitud escalada $f'_{min} = 0$.

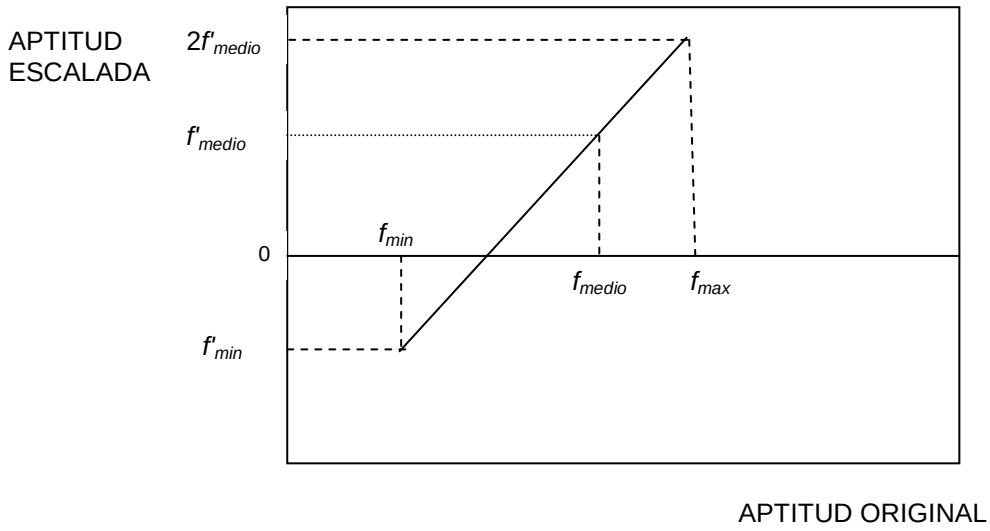


Figura 4.3. Dificultad en el escalado lineal: puntos con valor de aptitud muy bajo pueden ser escalados a valores negativos.

Para intentar evitar este problema, se puede usar información de la varianza de la población para preprocesar la aptitud original antes de escalarla. Este procedimiento es el método de truncamiento sigma (σ). Usando la información de la desviación estándar de la población, resta una constante de la aptitud original según:

$$f' = f - (\bar{f} - c\sigma) \tag{4.7}$$

En esta ecuación, la constante c es un múltiplo de la desviación estándar de la población (entre 1 y 3), y a los resultados negativos ($f' < 0$) se les asigna arbitrariamente un valor de 0.

Gillies [Gillies, 1985] sugirió una forma de escalado por ley de potencias, donde la aptitud escalada es una potencia especificada de la aptitud original:

$$f' = f^k \tag{4.8}$$

En el trabajo [De Jong, 1975] se presenta un estudio en el que se aplican algoritmos genéticos para optimizar funciones de distinta naturaleza: continuas /discontinuas, convexas /no convexas, unimodales /multimodales, cuadráticas /no cuadráticas, de baja dimensionalidad /de alta dimensionalidad, deterministas /estocásticas. Para cuantificar la efectividad de los distintos algoritmos genéticos, De

Jong diseña dos medidas: una mide la capacidad de convergencia y la otra mide la eficiencia, basándose en la función objetivo de cada problema planteado. Construye seis modelos distintos e investiga qué modelos optimizan mejor determinados tipos de funciones. En cada modelo cambia los mecanismos de selección y prueba distintos tipos de cruce y mutación.

4.6 Métodos de selección.

En el esquema típico de un GA desarrollado anteriormente, la probabilidad de seleccionar una hipótesis viene dada por el cociente entre su aptitud y la suma de la aptitud de los otros miembros de la población actual. Este método recibe el nombre de *selección proporcional de aptitud*, o *selección de ruleta*.

Existen otros métodos para seleccionar las hipótesis. Por ejemplo, la *selección por torneo* está basada en la competición entre subconjuntos de la población. Un número de individuos, llamado tamaño del torneo, se selecciona aleatoriamente y se lleva a cabo una competición selectiva. En el torneo de menor tamaño posible, se eligen dos individuos de la población actual al azar para competir. Al individuo con mayor valor de aptitud se le asigna una probabilidad de selección p, y al de menor aptitud se le asigna una probabilidad de selección $(1 - p)$. El ganador del torneo se copia en la población, reemplazando al perdedor. Este método produce poblaciones con más diversidad que el método de selección proporcional de aptitud. Además, el tamaño del torneo permite ajustar la presión de selección: un torneo de tamaño pequeño causa una presión de selección baja, y uno de tamaño grande, una presión de selección alta. Otra ventaja es que no requiere una comparativa centralizada entre las aptitudes de todos los individuos de la población, lo que permite paralelizar el algoritmo.

En otro método, llamado *selección por rangos* [Grefenstette y Baker, 1989], [Withley, 1989], los individuos de la población se ordenan según sus valores de aptitud, y la probabilidad de seleccionar una hipótesis es proporcional al rango que le corresponde en esta ordenación en vez de a su valor de aptitud.

En el método de *truncamiento* o *selección* (μ,λ) [Schwefel, 1995], a un número μ de padres se les permite generar λ descendientes, de los cuales se escogerán los μ mejores para ser padres en la siguiente generación. Una variante de este método es la

selección ($\mu + \lambda$) [Rechenberg, 1994], en el que tanto los descendientes como los padres participan en el proceso de selección.

El término "elitismo" fue introducido por De Jong [De Jong, 1975] y se aplica al conjunto de métodos de selección que obligan al GA a retener un número determinado de los mejores individuos en cada generación. Estos individuos se podrían perder si no son seleccionados para la reproducción o si son destruidos por el cruce o la mutación. Las estrategias de selección elitistas aumentan significativamente la eficiencia de los algoritmos genéticos.

4.7 Los operadores genéticos.

La generación de sucesores en los GAs viene determinada por un conjunto de operadores que recombinan y mutan los miembros seleccionados de la población actual. Estos operadores se corresponden a versiones idealizadas de operadores genéticos encontrados en la evolución genética. Los operadores más comunes son la reproducción, el cruce y la mutación.

En la reproducción, se elige un individuo de los seleccionados de la población actual y se copia íntegramente. Esta copia se inserta en la población de la siguiente generación. Con la reproducción no se introduce diversidad en la población, pero se garantiza que existan copias de los mejores individuos de una generación en la siguiente. Este operador es muy utilizado en las estrategias elitistas.

El cruce permite a los algoritmos genéticos buscar con eficiencia en espacios de muchas dimensiones. En esencia, el cruce reduce las dimensiones del espacio de búsqueda. Este operador genético produce dos nuevos descendientes de dos cadenas padres, copiando posiciones seleccionadas de cada padre. El elemento de la posición *i* en cada descendiente es copiado del elemento en la posición *i* de uno de los dos padres. La elección de cual de los dos padres contribuye con el elemento de la posición *i* se determina por una cadena adicional llamada *máscara de cruce*. En la Tabla 4.1 se muestran ejemplos de distintos tipos de cruce y mutación sobre cadenas codificadas con un alfabeto binario.

	Cadenas iniciales (padres)	Máscara de cruce	Descendientes (hijos)
Cruce en un único punto (*single-point crossover*)	11101001000 00001010101	11111000000	11101010101 00001001000
Cruce en dos puntos (*two-point crossover*)	11101001000 00001010101	00111110000	11001011000 00101000101
Cruce uniforme (*uniform crossover*)	11101001000 00001010101	10011010011	10001000100 01101011001
Mutación en un punto (*point mutation*)	11101001000		11101011000

Tabla 4.1. Ejemplos de distintos operadores de cruce y mutación sobre cadenas binarias.

Estudiando los ejemplos de la Tabla 4.1, podemos observar que, en la operación de cruce en un único punto, el primer descendiente toma sus primeros cinco bits del primer padre y los siguientes seis bits del segundo padre, porque la máscara de cruce 11111000000 especifica estas elecciones para cada una de las posiciones de los bits. El segundo descendiente usa la misma máscara de cruce, pero intercambiando los papeles de los padres, de modo que contiene los bits que no fueron usados para construir el primer descendiente. En este tipo de cruce, la máscara de cruce se configura siempre con una cadena que contiene n 1's contiguos, seguidos del número necesario de 0's para completar la cadena. La aplicación de esta máscara dará como resultado un descendiente en el que los primeros n bits se tomarán de uno de los padres y los restantes del otro. Cada vez que se desea utilizar este operador, el punto de cruce n se elige aleatoriamente, y la máscara de cruce es entonces creada y aplicada.

En el cruce en dos puntos, los descendientes se crean sustituyendo segmentos intermedios de uno de los padres en medio de la segunda cadena padre. La máscara de cruce es una cadena con n_0 0's, seguidos de una cadena contigua de n_1 1's, seguidos del número necesario de 0's para completar la cadena. Cada vez que se desea utilizar este operador, la máscara de cruce se genera eligiendo antes aleatoriamente los enteros n_0 y n_1. En el ejemplo de la Tabla 4.1 los descendientes se crean usando una máscara en la cual $n_0 = 2$ y $n_1 = 5$. Ambos descendientes se crean del mismo modo, pero intercambiando los papeles de los padres.

El caso más general, que engloba a estos dos métodos, es el cruce multi-punto, donde se toman varios puntos de cruce para intercambiar el contenido de las cadenas padres y generar los descendientes.

El cruce uniforme combina bits de los padres muestreados uniformemente. En este caso, la máscara de cruce es una cadena de bits generada aleatoriamente, en la que cada bit es elegido aleatoria e independientemente de los demás.

Otros métodos de cruce existentes intercambian subcadenas correspondientes a las mismas posiciones en las cadenas padres, eliminan repeticiones de elementos, realizan cruces cíclicos, etc.

Existen otros métodos más avanzados de cruce. El método de *recombinación de nidos* (en inglés, *brood recombination*) [Tackett, 1994], se basa en un hecho observado de la evolución natural: muchas especies producen más descendientes de los que se espera que sobrevivan. El exceso de descendientes no sobrevive, y así se corrigen los efectos nocivos de un mal cruce. Este método crea un nido de tamaño N cada vez que realiza un cruce, escoge dos padres de la población, realiza cruce aleatorio sobre los padres N veces y evalúa la aptitud de los descendientes, seleccionando los 2 mejores y descartando el resto. El gran inconveniente de este método es que normalmente la evaluación es un proceso lento y computacionalmente costoso, y es necesario realizar $2N$ evaluaciones. Una solución consiste en evaluar estos descendientes con una porción del conjunto de entrenamiento, en vez de hacerlo con el conjunto entero, para obtener una referencia de qué descendientes pueden ser mejores que otros de un modo más rápido.

Varios autores [Teller, 1996], [Zanonni y Reynolds, 1996] han propuesto operadores de cruce inteligentes. Hacer inteligente al operador consiste en proporcionarle información adicional que lo guíe al elegir los individuos que va a cruzar.

El operador de mutación produce un descendiente de un único padre introduciendo pequeños cambios aleatorios en esa cadena padre. Elige al azar un único alelo de la cadena y cambia su valor, con una probabilidad de mutación p_m. Un ejemplo de esta operación se muestra en la Tabla 4.1. Lo más frecuente es aplicar la mutación después del cruce. Otro tipo de mutación consiste en intercambiar dos alelos entre sí, o también, existen métodos en los que se elige una subcadena y se inserta aleatoriamente en otro punto de la cadena, cambiando así el orden del cromosoma. Si la probabilidad

de mutación p_m es pequeña, aparecerán los nuevos rasgos muy lentamente en la población; si esta probabilidad es muy alta, cada generación estará desligada de la anterior.

Algunos sistemas GAs utilizan otros operadores adicionales, normalmente operadores que son específicos para la representación de las hipótesis concreta empleada en el sistema. Por ejemplo, [Grefenstette, 1991] describe un sistema que aprende conjuntos de reglas para el control de un robot. Utiliza mutación y cruce, junto con un operador para reglas especializadas. [Janikow, 1993] describe un sistema que aprende conjuntos de reglas utilizando operadores que generalizan y especializan reglas por medio de una gran variedad de métodos (por ejemplo, reemplazando la condición de un atributo por un símbolo *"don't care"*).

4.8 La influencia de la diversidad de la población en la calidad del aprendizaje de un algoritmo genético.

Como ya se ha mencionado antes, los GAs son métodos de búsqueda aleatorios que tratan de encontrar las hipótesis que más se ajustan. Esta búsqueda es muy diferente a otros métodos de aprendizaje. Por ejemplo, en la búsqueda de espacios de hipótesis realizada por las redes neuronales *back-propagation*, el gradiente descendiente se mueve suavemente de una hipótesis dada a otra nueva que es muy similar a la primera. En cambio, los GAs se pueden mover de un modo mucho más abrupto, reemplazando las hipótesis "padres" por hipótesis descendientes que pueden ser radicalmente distintas a estas hipótesis padres. Una ventaja de esto es que la búsqueda por medio de GAs no cae tan fácilmente en mínimos locales como los métodos de gradiente descendiente.

Una dificultad práctica que parece en algunas aplicaciones con GAs es el problema del *crowding*. Este problema surge cuando existe un individuo en la población con un valor de aptitud muy alto (en comparación con los valores de aptitud de los restantes individuos) que se reproduce muy rápidamente, de modo que una fracción muy grande de la población está compuesta por copias de este individuo o elementos muy similares. El impacto negativo del *crowding* es que reduce la diversidad de la población, lo que ocasiona un progreso más lento en el algoritmo genético.

Se han estudiado diversas estrategias para reducir el fenómeno de *crowding*, además de los métodos de escalado de la función de aptitud ya expuestos anteriormente. Una aproximación consiste en cambiar la función de selección, por medio de criterios como la selección por torneo o por rangos, en lugar del método de la selección proporcional de aptitud por ruleta. Otra estrategia reduce la aptitud medida de un individuo si existe otro individuo similar en la población. Una tercera aproximación consiste en restringir los tipos de individuos a los que se les permite recombinarse para formar descendientes. Por ejemplo, si se permite la recombinación solamente entre individuos muy similares, se formarán clusters de individuos similares, o múltiples subespecies dentro de la población. Otro método se basa en distribuir espacialmente los individuos y permitir la recombinación sólo entre los individuos más cercanos. Muchas de estas técnicas están inspiradas en la evolución biológica.

La adaptación de un cromosoma es la probabilidad de que éste sobreviva a la siguiente generación. La adaptación ignora la diversidad, que se puede considerar como el grado con el que los cromosomas muestran genes diferentes. Según esto, los cromosomas tienden a ser barridos si su resultado es un poco menor que el de uno que está próximo al mejor cromosoma actual. Aun en poblaciones grandes, el resultado es la uniformidad. Sin embargo, en una escala mayor, los individuos y las especies no adaptados sobreviven bastante bien en nichos ecológicos que se encuentran fuera de la vista de otros individuos y especies relativamente adaptados, por lo que de aquí surge el principio de la diversidad: puede ser tan bueno ser diferente como lo es estar adaptado.

Cuando se seleccionan cromosomas para una nueva generación, una forma de medir la diversidad a la cual contribuiría un cromosoma candidato es calcular la suma de la inversa del cuadrado de las distancias entre el cromosoma y otros cromosomas ya seleccionados. Después, se determina el rango de diversidad de un cromosoma mediante dicha suma:

$$\sum_i \frac{1}{d_i^2} \tag{4.9}$$

Los máximos locales son más fáciles de manejar cuando se mantiene la diversidad. La mayoría de los planteamientos de búsqueda suponen que los máximos locales son trampas. En consecuencia, algunos de ellos implican mecanismos de escape de trampas como el retroceso y un tamaño de paso inicialmente grande que se va

haciendo cada vez más pequeño. Otros planteamientos implican una búsqueda paralela con un gran número de posiciones iniciales al azar, con la esperanza de que una de las búsquedas paralelas quede atrapada en el máximo local, que resulta ser también el máximo global.

Al contrario, si un algoritmo genético trata la diversidad como un componente de la adaptación, entonces algunos de los individuos de la población tienden a quedarse siempre alrededor de los máximos locales, en calidad o diversidad, ya descubiertos, alejando a los otros individuos. Siempre que haya suficientes individuos para poblar suficientemente todos los máximos locales, existirá una probabilidad razonable de que un individuo encuentre su camino hacia el máximo global. Por lo tanto, los máximos locales deben poblarse, no evitarse, cuando se está buscando un máximo global.

Michael de la Maza y Bruce Tidor muestran cómo la presión selectiva variable con el tiempo proporciona una forma de mantener la diversidad en una gran cantidad de problemas de optimización de muestra, entre los que figuran problemas sobre reconocimiento de proteínas [de la Maza y Tidor, 1991].

4.9 Teorema del esquema.

Una cuestión importante que se debe considerar es qué tipo de información maneja un GA. A primera vista, parece que solamente trabaja con cadenas (individuos) independientes y sus correspondientes valores de aptitud. En la realidad, existen patrones de cadenas (cadenas con similaridades en determinadas posiciones) asociados a una buena eficiencia. Por lo tanto, el GA también juega con estos patrones beneficiosos, que constituyen una información que lo ayuda en la búsqueda. Estos patrones reciben el nombre de *esquemas* o *patrones de similaridad*.

Supongamos que el GA trabaja con un alfabeto binario {0,1}. La idea de *esquema* es más fácil de entender añadiendo un símbolo especial a este alfabeto, el "*", que representa un *don't care*. Con el alfabeto extendido, {0,1,*}, un esquema encaja con una determinada cadena si en cada posición del esquema en la que hay un 1 o un 0, existe un 1 o un 0 respectivamente en la cadena en la misma posición, y si hay un * en el esquema, no importa qué símbolo correspondiente hay en la cadena. Por ejemplo, el esquema *0000 encaja con dos cadenas distintas {10000, 00000}. El símbolo * es sólo un meta-símbolo (un símbolo sobre otros símbolos), el GA nunca lo procesa

explícitamente. Se utiliza como un elemento de notación que permite la descripción de posibles similaridades entre cadenas de una determinada longitud y alfabeto.

En general, para alfabetos de cardinalidad k y cadenas de longitud l, existen $(k+1)^l$ esquemas posibles. Una cadena contiene 2^l esquemas. Por lo tanto, una población de tamaño n contiene entre 2^l y n esquemas, dependiendo de la diversidad de la población. No todos los esquemas se crean del mismo modo, algunos son más específicos que otros. Por ejemplo, el esquema 011*1** define similaridades más importantes que el esquema 0******. Algunos esquemas se expanden más a lo largo de la cadena que otros. Por ejemplo, el esquema 1****1* se expande más que el esquema 1*1****. Para cuantificar estas ideas, se introducen dos términos diferenciados: *orden* del esquema y *longitud de definición* del esquema.

El orden del esquema H, que se denota por $o(H)$, es el número de posiciones fijas (en un alfabeto binario, sería el número de 1's y 0's) que hay en el patrón de similaridad. Por lo tanto, el orden del esquema 011*1** es 4, mientras que el orden del esquema 0****** es 1.

La longitud de definición del esquema H, que se denota por $\delta(H)$, es la distancia entre la primera y la última posición específica de la cadena. Por ejemplo, el esquema 011*1** tiene una longitud de definición $\delta = 4$, porque la última posición específica es 5 y la primera es 1, siendo la distancia $\delta(H) = 5 - 1 = 4$. En el esquema 0******, sólo hay una posición fija, de modo que la primera y última posiciones específicas coinciden, siendo la longitud de definición $\delta = 0$.

Estos conceptos son importantes para analizar el efecto de los operadores genéticos sobre la población. Consideramos a continuación el efecto de la reproducción, el cruce y la mutación sobre las cadenas existentes dentro de una población.

El efecto de la reproducción sobre el número esperado de esquemas en la población es relativamente fácil de determinar. En general, existirán distintas cantidades de esquemas diferentes H en el instante t. Supongamos que en un determinado paso t del algoritmo existen m ejemplos de un determinado esquema H, contenidos en la población en ese momento $A(t)$, lo que se indica como $m = m(H,t)$. Durante la reproducción, una cadena se copia según su valor de aptitud, o más precisamente, una cadena A_i será seleccionada para pasar a la población en la siguiente generación con una

probabilidad $p_i = f_i / \sum f_i$. Después de seleccionar los individuos (cadenas) y formar una nueva población de tamaño *n* a partir de la población ***A***(*t*), se espera tener:

$$m(H,t+1) = m(H,t) \cdot n \cdot f(H) / \sum f_i \tag{4.10}$$

donde *f*(*H*) es la media de aptitud de las cadenas que contienen el esquema *H* en el instante *t*. Si escribimos la media de aptitud de toda la población como $\bar{f} = \sum f_i / n$, entonces la expresión 4.10 queda de la siguiente manera:

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \tag{4.11}$$

Por lo tanto, un esquema determinado aparece más en la población de la siguiente generación dependiendo de la media de aptitud de ese esquema en relación con la media de aptitud de la población. En otras palabras, los esquemas con valores de aptitud por encima de la media de aptitud de la población recibirán un número creciente de muestras en la siguiente generación, mientras que los esquemas con media de aptitud inferior a la media de aptitud de la población reciben un número decreciente de muestras en la siguiente generación. Es interesante destacar que esto se lleva a cabo en paralelo con cada esquema *H* que esté contenido en la población ***A***. Por lo tanto, todos los esquemas de una población crecen o decaen según sus valores medios de aptitud si se considera solamente el efecto de la reproducción.

Supongamos que la media de aptitud de un esquema particular *H* permanece por encima de la media de aptitud de la población en una cantidad $c\bar{f}$, siendo *c* una constante. Bajo esta suposición, se puede volver a escribir la ecuación 4.11 como:

$$m(H,t+1) = m(H,t) \cdot \frac{(\bar{f} + c\bar{f})}{\bar{f}} = (1+c) \cdot m(H,t) \tag{4.12}$$

Comenzando en el instante $t = 0$, y asumiendo un valor estacionario para *c*, se obtiene:

$$m(H,t) = m(H,0) \cdot (1+c)^t \tag{4.13}$$

La ecuación 4.13 es ecuación de la progresión geométrica, o la ecuación discreta de una exponencial. Por lo tanto, el efecto de la reproducción es el incremento (o decremento) exponencial de los esquemas con media de aptitud superior (o inferior) a la media de aptitud de la población.

La reproducción sola no promueve la exploración de nuevas regiones del espacio de búsqueda, ya que no busca nuevos puntos de búsqueda, solamente copia estructuras sin cambios. Es necesario aplicar el cruce para crear nuevas estructuras. No todos los esquemas se ven afectados por el cruce. Por ejemplo, supongamos una cadena binaria de longitud $l = 7$ y dos esquemas representativos dentro de esta cadena:

$$A = 0\ 1\ 1\ 1\ 0\ 0\ 0$$
$$H_1 = *\ 1\ *\ *\ *\ *\ 0$$
$$H_2 = *\ *\ *\ 1\ 0\ *\ *$$

En el cruce simple se elige aleatoriamente un punto de cruce para dos cadenas "padre" y se intercambian las subcadenas definidas por ese punto de corte entre los dos padres, dando lugar a dos descendientes. Supongamos que se elige la cadena A para realizar un cruce con otra cadena, y el punto de cruce se establece entre las posiciones 3 y 4, tal y como se muestra a continuación:

$$A = 0\ 1\ 1\ \ |\ 1\ 0\ 0\ 0$$
$$H_1 = *\ 1\ *\ \ |\ *\ *\ *\ 0$$
$$H_2 = *\ *\ *\ \ |\ 1\ 0\ *\ *$$

A menos que la cadena A' con la que se cruza A sea idéntica en las posiciones fijas del esquema, el esquema H_1 será destruido por que el 1 que aparece en la posición 2 y el 0 que aparece en la posición 7 serán colocados en distintos descendientes. Por el contrario, se observa que el esquema H_2 sobrevivirá puesto que las posiciones fijas de este esquema pasan intactas a los descendientes. Intuitivamente se puede apreciar que el esquema H_1 tiene menos posibilidades de sobrevivir que el esquema H_2, puesto que es más posible que el punto de corte caiga entre las posiciones fijas del primer esquema

que entre las posiciones fijas del segundo esquema. Esto está directamente relacionado con la longitud de definición del esquema. El primer esquema tiene una longitud de definición $\delta = 5$. Si el punto de corte para el cruce se elige uniformemente al azar entre los $l - 1 = 7 - 1 = 6$ sitios posibles, el esquema H_1 se destruirá con una probabilidad $p_d = \delta(H_1)/(l-1) = 5/6$, y sobrevivirá con una probabilidad de $p_s = 1 - p_d = 1/6$. Bajo el mismo planteamiento, el segundo esquema tiene una longitud de definición $\delta = 1$, y su destrucción ocurrirá si el punto de corte cae entre las posiciones 4 y 5, de modo que $p_d = 1/6$ y $p_s = 5/6$.

De un modo más general, se puede obtener una cota inferior en la probabilidad de supervivencia p_s de cualquier esquema tras el cruce. Un esquema sobrevive si el punto de cruce cae fuera de la longitud de definición, por lo tanto, la probabilidad de supervivencia del esquema será $p_s = 1 - \delta(H)/(l-1)$. La realización de una operación de cruce lleva asociada una elección aleatoria, con probabilidad p_c. Por lo tanto, la probabilidad p_s puede acotarse como:

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1} \tag{4.14}$$

que se reduce a la expresión anterior cuando $p_c = 1$.

Si consideramos la acción combinada de la reproducción y el cruce, la cantidad de un esquema determinado H que aparece en la población en la siguiente iteración será:

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1}\right] \tag{4.15}$$

El efecto combinado de las dos operaciones se obtiene multiplicando el número esperado de esquemas sólo por la reproducción por la probabilidad de supervivencia de estos esquemas tras el cruce. Ahora, que un esquema determinado aumente o disminuya en la población depende de dos factores: que su media de aptitud sea mayor o menor que la media de aptitud de la población y que el esquema tenga una longitud de definición mayor o menor. Aquellos esquemas con media por encima de la de la

población y longitud de definición pequeña irán aumentando exponencialmente su número en las siguientes generaciones.

El último operador a considerar es la mutación. La mutación es la alteración aleatoria con probabilidad p_m de una posición de la cadena. Desde el punto de vista de los esquemas, para que un esquema sobreviva, deben sobrevivir todas sus posiciones fijas, es decir, no debe ocurrir mutación sobre esas posiciones de la cadena. Un alelo sobrevive a la mutación con una probabilidad $(1 - p_m)$, y como cada una de las mutaciones son estadísticamente independientes, un esquema sobrevive cuando cada una de las $o(H)$ posiciones fijas del esquema sobreviven. Multiplicando la probabilidad de supervivencia $(1 - p_m)$ por sí misma $o(H)$ veces, obtenemos la probabilidad total del esquema de sobrevivir a la mutación, $(1 - p_m)^{o(H)}$. Para valores pequeños de p_m $(p_m << 1)$, la probabilidad de supervivencia a la mutación de un esquema será aproximadamente 1- $o(H)\, p_m$.

Por lo tanto, un esquema H recibe un número esperado de copias en la siguiente generación, bajo el efecto de la reproducción, el cruce y la mutación, según se indica en la ecuación 4.16.

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1} - o(H)\, p_m \right] \qquad (4.16)$$

La adición de cambios por mutaciones varía poco las conclusiones obtenidas hasta este momento. Los esquemas cortos (pequeña longitud de definición), de bajo orden y con media de aptitud superior a la media de aptitud de la población reciben un número creciente de copias en las siguientes generaciones, siendo este crecimiento un crecimiento exponencial. Esta conclusión es muy importante, y constituye el **Teorema del Esquema**, que es el teorema fundamental de los algoritmos genéticos.

En una población de n cadenas de longitud l se procesan entre 2^l y $n2^l$ esquemas. No todos se procesan con alta probabilidad puesto que el cruce destruye aquellos que tienen longitudes de definición relativamente grandes. Para contabilizar los esquemas que se procesan efectivamente se utiliza la estimación $O(n^3)$ de Goldberg [Goldberg, 1989]. Esta estimación indica que un algoritmo genético procesa con una población de n estructuras en cada generación, alrededor de n^3 esquemas. Este resultado es muy importante, y recibe el nombre de *paralelismo implícito*. En cada generación, se realiza

una computación proporcional al tamaño de la población y se procesan de modo efectivo n^3 esquemas en paralelo sin aumentar los costes de computación.

Consideremos, en una población de n cadenas binarias de longitud l, solamente los esquemas que sobreviven con una probabilidad mayor que una constante p_s. Asumiendo la operación de cruce simple y un ritmo de mutación pequeño, admitimos sólo aquellos esquemas cometiendo un error de $\varepsilon < 1 - p_s$. Esto nos permite considerar solo aquellos esquemas con longitud $l_s < \varepsilon(l - 1) + 1$.

Suponiendo una determinada longitud de esquema, podemos estimar una cota inferior del número de esquemas procesados en una población inicial aleatoria de cadenas. Para hacer esto, primero se contabiliza el número de esquemas de longitud l_s o menor. Luego se multiplica por un tamaño apropiado de población, elegido según el hecho esperado de que, en media, no hay más de un esquema de longitud $l_s/2$. El número total de esquemas de longitud l_s o menor será $2^{(l_s-1)} \cdot (l - l_s + 1)$. Este número indica los esquemas que hay para una cadena. Para sobreestimar el número de esquemas en toda la población, podríamos multiplicar este número por el tamaño de la población, n: $n \cdot 2^{(l_s-1)} \cdot (l - l_s + 1)$. Es una sobreestimación, ya que seguramente en poblaciones grandes existirán esquemas de bajo orden duplicados. Para refinar la estimación, se elige un tamaño de población $n = 2^{ls/2}$, de este modo se espera tener uno o menos esquemas de orden $l_s/2$ o mayor. El número de esquemas está distribuido binomialmente, por lo que se concluye que la mitad son de orden mayor que $l_s/2$ y la otra mitad de orden menor. Si contamos solamente los de mayor orden, estimamos una cota inferior del número de esquemas tal y como se muestra en la expresión 4.17.

$$n_s \geq \frac{n(l - l_s + 1)2^{(l_s-1)}}{2} \tag{4.17}$$

Esto difiere de la sobreestimación previa en un factor de ½. Además, al restringir el tamaño de la población al valor particular de $n = 2^{ls/2}$, la expresión 4.17 resulta ser:

$$n_s = \frac{(l - l_s + 1)n^3}{4} \tag{4.18}$$

que es equivalente a $n_s = Cn^3$, por lo que se concluye que el número de esquemas es proporcional al cubo del tamaño de la población, $O(n^3)$. A pesar de la destrucción de los esquemas de larga longitud de definición y de alto orden que realizan los operadores de cruce y mutación, los algoritmos genéticos procesan de modo inherente una gran cantidad de esquemas a la vez que procesan un número relativamente pequeño de cadenas. Los esquemas de bajo orden y longitud pequeña reciben un nombre especial: *bloques de construcción*.

4.10 El problema de la codificación de los individuos.

Las hipótesis en los GAs (individuos de la población) se suelen representar a menudo por cadenas de bits, para que puedan ser manipuladas fácilmente por operadores genéticos tales como la mutación y el cruce. Las hipótesis representadas por estas cadenas de bits pueden ser muy complejas. Por ejemplo, conjuntos de reglas *si-entonces* pueden ser representadas de esta manera eligiendo una codificación que establezca subcadenas específicas para cada pre-condición y post-condición de la regla. Ejemplos de esto se pueden encontrar en los trabajos de [Holland, 1986], [Grefenstette, 1998], [De Jong et al., 1993]. Pero la codificación binaria no es la única posible.

Como ya se ha expuesto antes, los algoritmos genéticos explotan similaridades entre las cadenas, los esquemas. Los esquemas que se procesan con mayor eficiencia son los de longitud de definición pequeña y bajo orden. Para elegir la codificación más adecuada para codificar las cadenas, se suelen seguir dos principios: el principio de los mayores esquemas de longitud corta y bajo orden, y el principio de los alfabetos mínimos:

- Según el primer principio, se debe seleccionar un código que permita hacer relevantes, para el problema considerado, los esquemas cortos y de bajo orden, e irrelevantes los esquemas distribuidos en otras posiciones fijas distintas.

- Según el segundo principio, se debe seleccionar el alfabeto más pequeño posible que permita realizar una expresión natural del problema.

El alfabeto binario ofrece el máximo número de esquemas por bit de información. Esto es bueno, ya que los esquemas son una guía en la búsqueda realizada por los algoritmos genéticos. Pero no siempre es el alfabeto más adecuado. En el trabajo de Jim Antonisse [Antonisse, 1989] se corrige la posición, largamente mantenida, de que las representaciones binarias son las mejores para los algoritmos genéticos.

Dependiendo de la naturaleza del problema, a veces es más adecuado extender los alfabetos admitiendo símbolos adicionales. En el caso extremo, el contenido de las cadenas se puede codificar con números reales. Esta codificación, normalmente conocida como codificación de punto flotante, da lugar a un aumento de cadenas de poca longitud. Al aumentar la riqueza del alfabeto, las operaciones de cruce y mutación se redefinen de diversas maneras. De hecho, el término de "cruce" se cambia por la noción de "recombinación". Por ejemplo, en la llamada *recombinación lineal*, dos cadenas de números reales $\boldsymbol{x}$ e $\boldsymbol{y}$ se recombinan dando lugar a dos descendientes $\boldsymbol{x'}$ e $\boldsymbol{y'}$ de la siguiente manera:

$$\boldsymbol{x'} = \alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y} \tag{4.19}$$

$$\boldsymbol{y'} = \alpha\boldsymbol{y} + (1 - \alpha)\boldsymbol{x} \tag{4.20}$$

donde α es un valor del intervalo (0,1). La continuidad del esquema de codificación también amplía el número de operadores de mutación. Por ejemplo, se pueden modificar los valores de las cadenas mediante incrementos aleatorios. En la Tabla 4.2 se muestran ejemplos de distintos operadores de cruce, siendo $\boldsymbol{x}$ e $\boldsymbol{y}$ las cadenas codificadas con números reales que participan como padres en el cruce, y $\boldsymbol{z}$ el resultado de la operación de cruce.

Cruce llano ***(flat crossover)*** [Radcliffe, 1991]	$z_i = U(\min(x_i,y_i), \max(x_i,y_i))$ con $i = 1, 2, \ldots, m$. $U(a,b)$ es una variable aleatoria con una función de distribución normal definida sobre [a,b]
Cruce simple ***(simple crossover)*** [Wright, 1991]	$z = [x_1 x_2 \ldots x_i \; y_{i+1} \ldots y_m]$ con i elegido aleatoriamente del rango del cromosoma (desde 1 hasta m).
Cruce BLX-α [Eshelman y Schaffer, 1993]	$z_i = U(\min(x_i,y_i) - I_i\alpha, \max(x_i,y_i) + I_i\alpha)$ donde $I_i = \max(x_i,y_i) - \min(x_i,y_i)$, con α perteneciente al intervalo unitario.

Tabla 4.2. Ejemplos de operadores de cruce definidos sobre cadenas codificadas con números reales.

Capítulo 5

Los sistemas clasificadores.

5.1 Introducción.

El aprendizaje automático mediante algoritmos genéticos aplicado al diseño de sistemas borrosos tiene sus raíces en los denominados sistemas clasificadores (en inglés *classifier system* (CS)).

En las conclusiones de un workshop dedicado a los CSs (IWLCS-92) [Smith, 1992] queda clara la dirección en la que hay procurar estudiar el denominado sistema clasificador:

"Un CS es generalmente descrito como un "método": es decir, un conjunto de elementos algorítmicos que definen el modo de resolver un problema. Sin embargo, en muchos sentidos el CS es más bien una estrategia: un conjunto de detalles conceptuales que definen cierta dirección para desarrollar métodos. Por lo tanto, las aspectos definitorios del CS no son necesariamente algorítmicos, sino conceptuales. El principal problema en el que se centraron las discusiones de este workshop fue clarificar estos aspectos definitorios conceptuales".

Así que cuando nos referimos a un CS estamos hablando de conceptos o estrategias con ciertos aspectos definitorios comunes.

Los CSs se basan en un tipo de aprendizaje, donde el conocimiento se adquiere del entorno, de forma que de este conocimiento se deriven estrategias para afrontar las situaciones que puedan acontecer. Con este enfoque podemos entender muchos problemas de aprendizaje.

En este sentido, se puede englobar al CS en la corriente general de aprendizaje automático denominada aprendizaje con refuerzo (*reinforcement learning* o RL). Los algoritmos RL interactúan con el entorno y tratan de construir un conjunto de reglas que clasifiquen los datos percibidos en clases de problemas con sus respectivas soluciones (figura 5.1). Por este motivo, en el caso de un CS, las reglas del conjunto son denominadas "clasificadores".

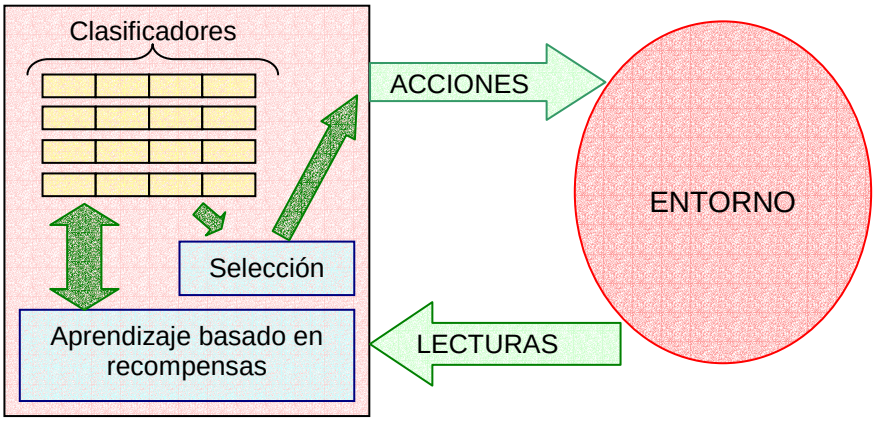


Figura 5.1. Esquema de un sistema basado en RL.

En cualquier caso nunca hay que perder de vista que existen diferencias muy importantes entre los CS y los algoritmos RL tradicionales, como vamos a ver. Sin embargo, dado que los algoritmos RL están en los fundamentos de los CS, pasamos a describir sus aspectos más importantes.

Los algoritmos RL tienen su origen en la cuestión de cómo un agente autónomo que lee el entorno mediante sus detectores y actúa sobre él, puede aprender a seleccionar las acciones óptimas a realizar para conseguir sus objetivos. Cada vez que el agente realiza una acción, el entorno cambia su estado. Un entrenador puede entonces asignar una recompensa al agente en función de la bondad del estado alcanzado en el entorno bajo la perspectiva de los objetivos a conseguir. A continuación el agente revisará su conjunto de reglas para determinar cuál se adapta a la nueva situación del

entorno y en el caso de que haya varias posibilidades, elegirá aquella con una mayor recompensa acumulada.

Con esta idea en mente es fácil entender que la mayoría de los algoritmos RL suficientemente entrenados acaban creando un homomorfismo, es decir, una relación uno a uno, entre todos los posibles pares problema-solución y un valor de aptitud. De esta forma, tras identificar el problema (estado del entorno) se buscará el conjunto de clasificadores que en su parte problema o condición refleje la situación actual. De todos los clasificadores se elegirá la solución (acción) propuesta por el de mayor aptitud. El proceso de aprendizaje tiene como objetivo crear un homomorfismo entre los pares (estado, acción) y valor de aptitud, de forma que, con el modo de actuación descrito, el sistema cumpla sus objetivos en el mayor grado posible. Como este homomorfismo supone una relación uno a uno entre todos los posibles pares (estado, acción) y valor de aptitud, se le denomina mapa completo.

5.2 El algoritmo Q-learning.

Para entender mejor el significado de este mapa completo, vamos a describir el algoritmo Q-learning. El objetivo del Q-learning es maximizar $Q(s,a)$, es decir, la recompensa esperada de tomar la acción a en el estado s. El algoritmo mantiene y actualiza una tabla de valores *Q*, uno para cada estado y cada acción.

Se define la utilidad E de un estado como el valor máximo de *Q* en un estado, tomando en consideración todas las acciones que se pueden aplicar en este estado. El valor *Q* que se deriva de la realización de una acción es la suma de la recompensa inmediata proporcionada por el entorno, *r*, y la utilidad *E*(*s'*) del siguiente estado (*s'*). La transición al siguiente estado queda definida por la denominada función de transición *T*, afectada por el parámetro γ. Formalmente:

$$\begin{aligned} & s' \leftarrow T(s,a) \\ & E(s) = \max_a Q(s,a) \\ & Q(s,a) = r + \gamma E(s'), 0 \le \gamma \le 1 \end{aligned} \tag{5.1}$$

donde s' es el estado alcanzado cuando se aplica la acción a, estando en el estado s, $E(s)$ es la utilidad del estado s', y r es la recompensa inmediata.

Los valores de Q se actualizan mediante la siguiente regla:

$$Q(s,a) \leftarrow Q(s,a) + \beta(r + \gamma E(s') - Q(s,a)) \; 0 \leq \beta \leq 1 \tag{5.2}$$

donde las constantes β y γ juegan un papel importante que luego se discutirá.

La aplicación de un método RL, usando Q-learning tiene la siguiente forma:

1. Inicializar $Q(s,a)$.
2. Realizar continuamente los siguientes pasos (bucle principal):
   a. Observar el estado actual.
   b. Elegir una acción que maximice $Q(s,a)$.
   c. Ejecutar la acción a.
   d. Sea r la recompensa inmediata por ejecutar la acción a en el estado s.
   e. Actualizar $Q(s,a)$ conforme a la regla anterior.

Observando la fórmula 5.2 podemos apreciar que el papel del parámetro β está en el mecanismo de adaptación. Por ejemplo, si β=1, el nuevo valor de $Q(s,a)$ no tiene en cuenta la historia anterior del valor de Q, sino que será la recompensa directa más la utilidad corregida por un factor constante.

El papel de γ es pesar la recompensa futura. Idealmente, este valor debería ser 1 ya que así se pesaría por igual la recompensa inmediata y la futura, pero en realidad se usan valores próximos a 1.

Los valores iniciales utilizados para la tabla de valores de Q tienen una influencia importante en el proceso de convergencia. Intuitivamente, si estos valores están cercanos a los valores óptimos, la convergencia será rápida. Por el contrario, determinadas configuraciones iniciales pueden hacer que la convergencia sea muy lenta en función de la situación relativa de los valores óptimos, o que incluso esta convergencia al valor óptimo no se produzca. De hecho, éste es uno de los principales inconvenientes del Q-learning, ya que la convergencia del algoritmo sólo se garantiza teóricamente para un número de visitas a cada estado infinito [Watkins, 1989].

Los principales problemas del Q-learning, son la sensibilidad del algoritmo con la elección de los parámetros que lo gobiernan, así como la complejidad en el espacio y

tiempo (necesidad de almacenamiento y gran número de iteraciones para garantizar la convergencia) debido al posible gran número de estados para un entorno real.

5.3 Jerarquía de clasificadores.

Empecemos a introducir ahora, las diferencias más notables entre los algoritmos RL y los CS. El mapa creado en un CS no es un homomorfismo, ya que se admiten reglas con elementos "don't care" o comodín en su parte de condición. Los elementos comodín en la parte de condición de los clasificadores aumentan el número de situaciones compatibles con el clasificador haciendo de él una regla más general. Si imaginamos la condición del clasificador como un conjunto de atributos del entorno que deben darse para la aplicación del clasificador, la sustitución de uno de los atributos por el comodín provoca que todas las situaciones del entorno que encajan con el resto de atributos (independientemente del valor que tome en el entorno el atributo que fue sustituido por el comodín) pasen a ser compatibles con el clasificador. De esta manera, se crean agrupaciones de estados asociados a un solo clasificador con su acción correspondiente.

Por consiguiente, la inclusión de comodines en la definición de los clasificadores provoca diferentes grados de generalidad en la aplicación de los mismos, estableciéndose así una jerarquía de clases, donde unas clases engloban a otras. Esta representación suele ser más compacta (requiere menos clasificadores) que los mapas completos producidos por los sistemas típicos RL. La jerarquía de clases se suele conocer como "*default hierarchies*", o jerarquía de situaciones por defecto, y fue teorizada en primer lugar por Holland en 1981 [Holland, 1981]. Más tarde, Goldberg [Goldberg, 1983], demostró la existencia de las mismas en la práctica en una de las primeras aplicaciones de los CS en el control de gaseoductos.

La utilización de las jerarquías por defecto presenta algunos problemas, cuando la selección del clasificador a aplicar depende de un valor de fuerza acumulativa asociado a cada clasificador y ganado durante el proceso de aprendizaje y aumentado o disminuido en función de la recompensa obtenida por ese clasificador del entorno. Es lo que se denomina ***aptitud basada en la fuerza*** (en inglés, *strength-based fitness*). Antes de enumerar los problemas mencionados, debemos distinguir entre reglas correctas y reglas incorrectas.

Una regla es correcta para un estado s, cuando la aplicación de la misma en ese estado supone un incremento de la fuerza de la regla. Por el contrario, una regla es incorrecta para un estado s, cuando la aplicación de la misma en ese estado supone un decremento de la fuerza de la regla.

Los problemas derivados de la utilización de un sistema de generalización como el descrito, junto con la aplicación de una aptitud basada en la fuerza de los clasificadores, son:

- **Reglas sobre-generales.** Una regla sobre-general es aquella que encaja en múltiples estados, pero es incorrecta en una minoría de ellos. Esta regla irá acumulando fuerza debido a que actuará correctamente en la mayoría de los estados con los que se encaja, lo que le lleva a que sea escogida muy probablemente para actuar en los estados con los que encaja pero es incorrecta. Esto va en detrimento del funcionamiento general del sistema.

- **El problema del clasificador acaparador.** La función de recompensa puede dar diferentes valores de recompensa para acciones correctas según el estado en el que se encuentre el entorno. De esta manera, aquellos clasificadores a los que se tiende a recompensar más por el mero hecho de encajar en estados "privilegiados" tendrán más fuerza que otros y tenderán a competir con ventaja con los demás. Si esta tendencia es muy fuerte, se pueden llegar a producir vacíos en el mapa de recubrimiento que se va generando. Es decir, en estados no especialmente privilegiados por la función de recompensa, el sistema puede dejar un hueco (ausencia de un clasificador que encaje en el estado).

- **Clasificadores sobre-generales fuertes.** Es la combinación de los dos problemas anteriores. Supongamos que una regla sobre-general actúa correctamente en un estado *A* con una alta recompensa, pero actúa incorrectamente en un estado *B* con una baja recompensa. Pensemos además, que hay una regla o clasificador particular que actúa correctamente en el estado *B*. Esto lleva a dos consecuencias. Como la selección de la acción se realiza en función de la fuerza del clasificador, la regla sobre-general tendrá más influencia en el estado de baja recompensa donde actuará incorrectamente. Por otra parte,

el problema del clasificador acaparador implica que la regla sobre-general tendrá mayores probabilidades de reproducirse, eliminando a la regla correcta completamente.

Esta problemática está relacionada con la elección de funciones de recompensa con sesgo. Una función de recompensa no tiene sesgo, si todas las acciones correctas devuelven la misma recompensa y todas las acciones incorrectas devuelven la misma penalización, independientemente del estado. Desgraciadamente, evaluar el nivel de sesgo en una función de recompensa para un sistema real no es una tarea nada sencilla.

Para comprender mejor el problema de los sobre-generales fuertes, vamos a describir el siguiente ejemplo sencillo extraído de [Kovacs, 2002].

Supongamos un entorno con sólo dos estados, en donde además se admiten sólo dos posibles acciones en cada estado. Vamos a construir una función de recompensa sin sesgo y otra con sesgo para comparar ambas situaciones.

Estado	Acción	Recompensa
0	0	1000
0	1	0
1	0	0
1	1	1000

Tabla 5.1. Ejemplo de función de recompensa sin sesgo.

Como vemos en la función de recompensa sin sesgo (tabla 5.1) todas las acciones correctas tienen la misma recompensa independientemente del estado (lo mismo ocurre con las acciones incorrectas).

En la tabla 5.2 se representa una función de recompensa con sesgo para el mismo sistema.

Estado	Acción	Recompensa
0	0	1000
0	1	0
1	0	0
1	1	200

Tabla 5.2. Ejemplo de función de recompensa con sesgo.

En este caso, la acción correcta para el estado 1 recibe una recompensa menor, incluyendo así un sesgo.

A continuación se presentan en la tabla 5.3 todos los clasificadores posibles que se pueden dar para este entorno.

Clasificador	Condición	Acción	Fuerza del clasificador E asumiendo que no hay sesgo	Fuerza del clasificador E asumiendo un sesgo
A	0	0	1000	1000
B	0	1	0	0
C	1	0	0	0
D	1	1	1000	200
E	#	0	500	500
F	#	1	500	100

Tabla 5.3. Lista de clasificadores posibles para el sistema ejemplo.

Los clasificadores E y F son sobre-generales ya que encajan con los dos estados, actuando correctamente en uno e incorrectamente en el otro. En las últimas dos columnas se ha estimado la fuerza de cada clasificador. Para los sobre-generales se supone que los dos estados posibles tienen aproximadamente la misma probabilidad, por lo que la fuerza se calcula como la media de las recompensas obtenidas de la situación correcta y de la situación incorrecta.

De esta manera, podemos ver que el clasificador E es un sobre-general fuerte en el caso de usar una función de recompensa con sesgo, ya que su fuerza resulta ser superior que la del clasificador D, que actúa correctamente en la situación recompensada con un valor más bajo. El clasificador E, a pesar de ser incorrecto en este estado sería probablemente escogido dado su mayor valor de fuerza.

Podemos profundizar un poco más en el papel que juega el sesgo de la función de recompensa, en la aparición del sobre-general fuerte. Sea “c” la recompensa obtenida por el sobre-general cuando actúa correctamente, sea “i” la recompensa obtenida por el sobre-general cuando actúa incorrectamente y sea “a” la recompensa obtenida por el clasificador correcto en la situación a la que la función de recompensa le ha asignado un valor menor. Si suponemos que el sobre-general va actuar correcta e incorrectamente con la misma probabilidad, por lo que su fuerza es un promedio de “c” e “i”, la condición para que aparezca el problema del sobre-general fuerte es $(c+i)/2>a$. Este cálculo se ha realizado en un ejemplo muy simplificado respecto de lo que puede ser un entorno más realista, pero deja claro el importante papel de la función de recompensa en la aparición de sobre-generales fuertes.

5.4 Estructura de un sistema clasificador.

Los algoritmos CS tienen sus comienzos en los trabajos Holland y Reitman en la década que va de 1975 a 1985. Fueron planteados por primera vez por John. H. Holland en un libro publicado en 1975 [Holland, 1975]. Como se ha descrito previamente, el objetivo del método planteado por Holland es buscar un sistema basado en reglas para determinar acciones sobre el entorno y que "aprendiese" a reaccionar en función de las situaciones encontradas. El modelo incluye un algoritmo genético, donde la población está constituida por reglas individuales, de las cuales se extrae un cierto número para constituir el sistema de reglas buscado. La idea de construir la población con partes de la solución completa en lugar de por muchas posibles soluciones recibe el nombre de "aproximación de Michigan" [Holland y Reitman, 1978], [Booker, 1982], utilizando el nombre de la Universidad donde estos investigadores desarrollaban su trabajo. En contraposición, tenemos la "aproximación de Pittsburgh" [Smith. 1980], [Smith, 1983], donde la población se compone de individuos que representan soluciones completas al problema. Vamos a estudiar cada uno de estos métodos por separado.

5.4.1 Sistemas tipo Michigan.

La obtención de un sistema clasificador mediante una aproximación de tipo Michigan requiere la generación automática de poblaciones de reglas (clasificadores) que **cooperen** en el desarrollo de una tarea deseada: nuevamente recalcamos que la población se compone de porciones de la solución total. Además, otras características fundamentales son: el paralelismo inherente a un sistema de reglas y el método de aprendizaje /adaptación basado en las técnicas de asignación de recompensas y descubrimiento de reglas [Holland, 1976]. En particular, los sistemas clasificadores tradicionales utilizan un algoritmo evolutivo como operador de descubrimiento e innovación para generar nuevos clasificadores. La asignación de recompensas es responsabilidad de algoritmos sustentados por una estrategia del tipo RL (Reinforcement Learning) como el algoritmo Q-learning.

Podemos encontrar en la literatura relacionada diferentes arquitecturas para la implementación de sistemas clasificadores. Explicaremos aquí el esquema básico propuesto por Holland, que está compuesto por tres partes principales: el sistema de actuación, el sistema de asignación de créditos y el sistema descubridor de

clasificadores. Estas partes se relacionan entre sí y con el entorno tal y como se muestra en la figura 5.2, extraída de [Cordón et al., 2001].

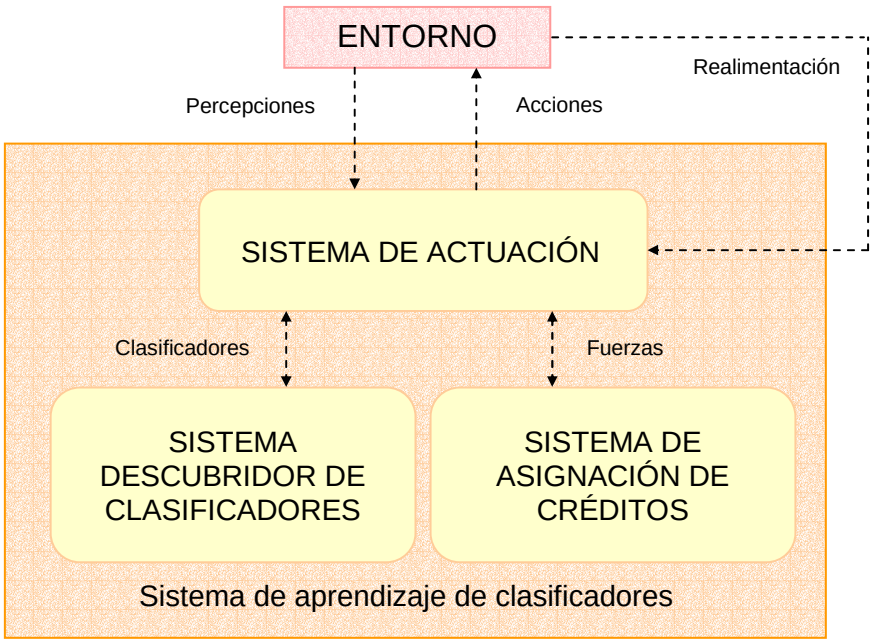


Figura 5.2. Sistema clasificador.

El sistema de actuación es la parte que interacciona directamente con el entorno. Podemos estructurarlo en tres áreas:

1. Área de interfaz. Su cometido es establecer una relación entre el sistema y el entorno. Esta área se subdivide en los siguientes módulos:

- *Interfaz de entrada*: En un sistema destinado a interactuar con un entorno, es fundamental la existencia de sensores para recoger los datos que forman la base de la percepción que del exterior se construye dentro del sistema. La interfaz de entrada contiene estos detectores que deben funcionar como transductores, realizando la codificación de las señales recogidas del exterior en **mensajes**. Denominamos mensajes a aquellas estructuras de datos externos o internos con las que las partes del sistema destinadas a construir una reacción y a aprender pueden realizar su tarea. En el caso del interfaz de entrada, tendremos mensajes externos.

- *Interfaz de salida*: De manera análoga al interfaz de entrada, este módulo debe contener uno o varios actuadores, además de transductores que conviertan los mensajes denominados de acción, en los correspondientes comandos para los actuadores.

2. Área de conocimiento adquirido y memoria de hechos. La misión de este área es el almacenamiento codificado de conocimiento en forma de reglas, además del equivalente a una memoria a corto plazo que refleje tanto la situación actual del entorno, como el estado del sistema en forma del conjunto de acciones dictaminadas por el área de decisión. Está formada por los siguientes elementos:

- El conjunto de reglas (*lista de clasificadores*), codificadas mediante un alfabeto compuesto por un número finito de símbolos que incluye el denominado comodín (*don't care*). Cada clasificador está estructurado como una regla *SI-ENTONCES,* con una parte de condición y una parte de acción. Hay al menos un parámetro asociado a cada clasificador denominado "fuerza", que se utiliza en la resolución de conflictos a la hora de establecer prioridades en la aplicación de un clasificador, o bien como base para estimar una "aptitud" para los individuos de la población de clasificadores.

- *Lista de mensajes*: Está compuesta por los mensajes externos provenientes del interfaz de entrada, así como por los últimos mensajes internos o mensajes de acción obtenidos a partir de los mecanismos de decisión encargados de "disparar" un subconjunto de clasificadores de la lista de reglas. De esta forma, el modelo de CS de Holland, se convierte en un **sistema recurrente**, ya que las decisiones sobre qué clasificadores van a ser utilizados depende de las últimas acciones realizadas por el sistema.

3. Área de decisión. Esta parte del sistema se encarga de establecer qué clasificadores deben ser activados para producir las acciones sobre el exterior. Esta tarea se realiza por dos módulos:

- *Módulo de búsqueda de encajes*: Su misión es identificar los clasificadores que encajan con el estado del entorno y del sistema. Para ello deben examinar la lista de mensajes y la parte de "condición" de los clasificadores.

- *Módulo de resolución de conflictos*: Tiene que decidir qué clasificador se dispara en caso de que existan varios que encajen con la situación actual del entorno) y realiza acciones sobre el entorno (aplica la acción del clasificador cuya parte de condición encaja con la situación del entorno). El módulo de resolución de conflictos estudia las fuerzas de los clasificadores candidatos para tomar la decisión de cual disparar.

Como ya se ha mencionado, el aprendizaje en un sistema clasificador se divide en dos procesos de aprendizaje distintos: el aprendizaje desarrollado por el sistema de asignación de créditos y el aprendizaje desarrollado por el sistema descubridor de clasificadores.

5.4.1.1 El sistema de asignación de créditos. Algoritmo "bucket brigade" estándar.

El sistema de asignación de créditos es el que controla y ajusta las fuerzas de las reglas. Existen muchas variantes en la realización de esta tarea. Se basan en procesos de recompensas y penalizaciones sobre las reglas, en función de lo beneficiosas que hayan resultado las acciones que realizaron sobre el entorno. Esto se traduce en aumentos o disminuciones en los valores de fuerza de las reglas. La función del sistema de asignación de créditos es estimar la utilidad /eficacia del clasificador en su actuación sobre el entorno.

El algoritmo de asignación de crédito clásico se denomina "*bucket brigade*" y fue propuesto por Holland. Vamos a describir ahora el denominado "bucket brigade estándar" tal y como se presenta en [Goldberg, 1989].

Una forma de verlo es como un sistema económico, en donde las entidades que lo componen son los clasificadores, y lo que se compra y vende es el derecho de enviar información. Los consumidores están en el interfaz de salida, esperando recibir información de los productores, que en este caso son los elementos clasificadores.

Existen dos elementos importantes que participan en este servicio económico: una casa de subastas y una cámara de compensación. Veamos en primer lugar el papel de la casa de subastas.

Cuando un clasificador encaja, no se envía directamente su mensaje. En lugar de ello, el privilegio ganado por el clasificador por haber encajado, es participar en una subasta. Su papel, entonces, vendrá regulado por la fuerza asignada al clasificador. El clasificador participa en la subasta con una oferta proporcional a la fuerza. Así, las reglas con una fuerza mayor tienen más preferencia que otras, ya que los clasificadores que hagan una oferta mayor son elegidos para enviar sus mensajes.

Ahora entra en juego la cámara de compensación. Cuando un clasificador es seleccionado para ser activado en la subasta, debe pagar su oferta en la cámara de compensación, reduciendo así su fuerza. A su vez, la cámara de compensación distribuye este pago entre aquellos clasificadores que participaron en llegar a la situación por la cual el clasificador elegido encajó, con lo que la fuerza de estos clasificadores es incrementada. De este modo, es posible crear subpoblaciones de reglas que cooperan, teniendo estos subconjuntos un número apropiado de reglas. Siguiendo a Goldberg [Goldberg, 1989], en un sistema de reglas no podemos buscar una regla maestra, sino un grupo de reglas coadaptadas que en conjunto tratan de cubrir una parte del comportamiento responsable del acercamiento a los objetivos establecidos para el sistema. Vamos a tratar ahora de explicitar un poco mejor este esquema.

Un clasificador varía su fuerza por varios motivos que podemos enumerar. Reduce la fuerza, por los pagos que debe realizar si su oferta les hace ganar en la subasta ($P_i(t)$). Incrementa su fuerza, gracias a recompensas recibidas del entorno, o bien, por ser partícipe de una acción previa que llevó a un clasificador ganador (mecanismo de compensación) ($R_i(t)$). Además, podemos incorporar un término general que haga disminuir la fuerza del clasificador y que podemos equiparar a un impuesto $T_i(t)$. De esta manera, la fuerza de cada clasificador varía según la fórmula:

$$S_i(t+1) = S_i(t) - P_i(t) - T_i(t) + R_i(t) \tag{5.3}$$

El clasificador oferta proporcionalmente a su fuerza, lo que puede representarse con una constante C_{oferta}

$$B_i = C_{oferta} S_i \tag{5.4}$$

Una forma de seleccionar a los clasificadores ganadores de la subasta puede ser determinista, y tener en cuenta solamente las ofertas. De esta manera se pueden seleccionar los *k* clasificadores con las ofertas más altas. Sin embargo, este método puede llevar a una situación de "status quo" [De Groot, 1970]. Para evitar esto, se introduce ruido aleatorio en el proceso de subasta, haciendo que ésta sea no determinista. Calculamos así una oferta efectiva, añadiendo a la oferta B_i un término aleatorio:

$$Eb_i = B_i + N(\sigma_b) \tag{5.5}$$

representando $N(\sigma_b)$ un proceso aleatorio caracterizado por un parámetro de desviación estándar σ_b.

Los ganadores del proceso de subasta pagan sus ofertas (los B_i) a la casa de compensación, y ésta distribuye el pago entre los clasificadores que enviaron un mensaje que contribuyó a la consecución de la situación actual.

Finalmente, cada clasificador debe ser penalizado independientemente de si ganaron o no, mediante un impuesto. Esto se hace para evitar que clasificadores ociosos permanezcan con valores altos de la fuerza, simplemente como consecuencia de su inactividad. Hay muchas posibilidades para definir este término, una de ellas es:

$$T_i = C_{impuesto} S_i \tag{5.6}$$

Podemos predecir algunas propiedades de este sistema de asignación de crédito. La fórmula, tras realizar las sustituciones pertinentes, es:

$$S_{t+1} = S_t - C_{bid} S_t - C_{tax} S_t + R_t \tag{5.7}$$

Agrupando términos:

$$S_{t+1} = (1 - K) S_t + R_t \tag{5.8}$$

donde

$$K = C_{oferta} + C_{impuesto} \tag{5.9}$$

Se trata de una ecuación en diferencias. Suponiendo por un momento que $R(t)$ es 0, tenemos la siguiente ecuación homogénea:

$$S_{t+1} = (1-K)S_t \tag{5.10}$$

La solución de esta ecuación es:

$$S_t = (1-K)^t S_0 \tag{5.11}$$

Esta es una solución estable (no tiende a infinito) para S_0 arbitrario cuando $0 \le K \le 2$, sin embargo en la práctica se mantiene $K \le 1$ para garantizar la no negatividad de las fuerzas. Esta propiedad de estabilidad nos permite prever el acotamiento de las fuerzas a lo largo de las iteraciones del algoritmo, pero podemos ir incluso más allá, examinando que ocurre en un clasificador activo que recibe una recompensa. Un clasificador que permanece activo, presenta una fuerza que responde a la ecuación 5.12.

$$S_t = (1-K)^t S_0 + \sum_{j=0}^{n-1} R_j (1-K)^{t-j-1} \tag{5.12}$$

Aquí se ha ignorado el hecho de que los clasificadores están en ocasiones activos y en otras no, lo cual se puede entender como una dependencia temporal en la constante (pasaríamos a tener K_t).

Supongamos que el clasificador recibe continuamente un valor constante $R_t = R_{ss}$. En este caso se llega a un valor de fuerza para el clasificador estacionario dado por:

$$S_{ss} = R_{ss} / K \tag{5.13}$$

La fuerza ha sido amplificada por el factor $(1/K)$. Por otra parte, la oferta del clasificador en el estacionario puede ser derivada como:

$$B_{ss} = (C_{oferta} / K)R_{ss} = R_{ss}C_{oferta} /(C_{oferta} + C_{imppuesto}) \quad (5.14)$$

Como $C_{impuesto}$ es mucho menor que C_{oferta} normalmente, en el estacionario la oferta del clasificador tiende a igualarse con la recompensa.

5.4.1.2 Relación entre el "bucket brigade" y el método "Q-learning".

Para entender la relación entre un algoritmo "bucket brigade" y el método de "Q-learning" vamos a usar el procedimiento expuesto en [Dorigo y Bersini, 1994]. Se trata de empezar con un CS muy simplificado denominado VSCS (*very simple classifier system*).

Las simplificaciones realizadas en el CS son:

1. Una sola condición y una sola acción en el clasificador.
2. La lista de mensajes se compone de 1 solo elemento (mensaje del entorno).
3. No se utiliza el elemento comodín en el alfabeto.
4. Hay una copia de todos los clasificadores posibles. Dado que se tienen en la población todos los pares posibles estado-acción, no hay necesidad de usar un algoritmo genético u otro sistema de búsqueda para modificar el recubrimiento del espacio de entradas.

De la restricción 2, se deduce que sólo será posible un mensaje en cada iteración, y este mensaje viene de los sensores que monitorizan el entorno. Por otra parte, la restricción 1 está relacionada con este hecho, pues sólo se tendrá una condición para el mensaje del entorno.

La restricción 3 elimina la capacidad de generalización del sistema. Será necesario tener un mapa completo correcto para que el sistema funcione, lo cual se establece en la restricción 4.

Veamos ahora el algoritmo en pseudocódigo:

1. Crear un clasificador para cada par estado-acción.
2. Inicializar t:=0.

3. Inicializar para cada clasificador c, la fuerza en el tiempo t, $S_t(c_c, a_c)$, donde hemos explicitado la dependencia con el clasificador, siendo c_c la parte de condición y a_c la parte de acción.
4. Repetir siempre:
    1. Leer el mensaje del entorno.
    2. Sea M, el conjunto de clasificadores que encajan con el mensaje.
    3. Elegir un clasificador $c \in M$, con una probabilidad establecida por $\frac{S_t(c_c, a_c)}{\sum_{d \in M} S_t(c_d, a_d)}$.
    4. Cambiar la fuerza de los clasificadores mediante el "bucket brigade" implícito (ver a continuación).
    5. $t = t + 1$
    6. Ejecutar a_c.

En el VSCS, la ecuación que regula el cambio en la fuerza del clasificador se basa en el denominado "bucket brigade" implícito que es una variante del "bucket brigade" estándar descrito más arriba.

En el "bucket brigade" implícito la ecuación es:

$$\begin{aligned} S_{t+1}(c_c, a_c) &= (1-\alpha) S_t(c_c, a_c) + R + \alpha S_{t+1}(c_d, a_d) = \\ &= S_t(c_c, a_c) + \alpha \left[\frac{R}{\alpha} + S_{t+1}(c_d, a_d) - S_t(c_c, a_c) \right] \end{aligned} \tag{5.15}$$

Según esta ecuación, si en el paso t se activa el clasificador c, su fuerza cambia en el paso $t+1$, y este cambio equivale a la suma algebraica de la recompensa externa R obtenida del entorno tras realizar la acción a_c, y la suma ponderada de la fuerza acumulada hasta el paso anterior en el clasificador c y la fuerza acumulada hasta el paso actual del clasificador activado en el paso $t+1$, d.

Este es el mismo sistema de recompensa que el propuesto en [Wilson, 1985] con la diferencia de que sólo un clasificador es activado cada vez. En [Goldberg, 1989] este algoritmo fue denominado "bucket brigade" implícito.

El sentido de la palabra implícito es que no hay una conexión directa entre el clasificador activado en el instante t, y el activado en el instante $t+1$. Es decir, no hay mensajes internos que lleven a la activación del clasificador en el instante $t+1$. Simplemente su activación se debe a los cambios en el entorno, fruto de la acción del clasificador activado en el instante t. Estos cambios provocan la activación del clasificador en el instante $t+1$, es decir, hay una relación implícita. En el "bucket brigade" estándar es posible que un mensaje externo active a un clasificador, por lo que está claro que el "bucket brigade" implícito está englobado dentro de éste.

Volvamos ahora al algoritmo Q-learning, revisándolo mediante la siguiente notación:

$$\begin{aligned} &Q_{t+1}(x,a) = (1-\alpha)Q_t(x,a) + \alpha\left[R + \gamma \underset{b}{\mathrm{MAX}} Q_t(y,b)\right] = \\ &Q_t(x,a) + \alpha\left[R + \gamma \underset{b}{\mathrm{MAX}} Q_t(y,b) - Q_t(x,a)\right] \end{aligned} \tag{5.16}$$

donde y es el estado obtenido tras ejecutar la acción a en el estado x.

En el VSCS tenemos que la condición c_c se puede identificar con el estado x, la acción a_c es identificable con a, la condición c_d con y y la acción a_d con b. En estos términos, la fórmula para la actualización de las fuerzas en el "bucket brigade" implícito del VSCS es:

$$S_{t+1}(x,a) = S_t(x,a) + \alpha\left[\frac{R}{\alpha} + S_{t+1}(y,b) - S_t(x,a)\right] \tag{5.17}$$

que modificada para adaptarse al esquema de Q-learning, queda:

$$S_{t+1}(x,a) = S_t(x,a) + \alpha\left[R + \gamma \underset{b}{\mathrm{MAX}} S_{t+1}(y,b) - S_t(x,a)\right] \tag{5.18}$$

Veamos las dos diferencias más importantes, entre estos algoritmos en el contexto del VSCS:

i) Q-learning evalúa el siguiente estado y eligiendo el valor de la mejor acción posible, mientas que en el "bucket brigade" implícito, la evaluación tiene en cuenta el par estado-acción usado realmente.

ii) En el "bucket brigade" implícito la contribución del siguiente estado al que se llega no sufre un descuento adicional respecto término negativo de la oferta pagada por el clasificador seleccionado ($S_{t+1}(y,b) - S_t(x,a)$). En el caso del Q-learning, el término correspondiente al siguiente estado sufre un descuento que viene dictado por la constante γ: ($\gamma \underset{b}{\text{MAX}} S_{t+1}(y,b) - S_t(x,a)$).

Respecto a la primera de las diferencias, en [Dorigo y Bersini, 1994] se comenta el uso de un operador generalizado para la evaluación del siguiente estado. El operador tiene como casos particulares el MAX (utilizado en el Q-learning) y el valor actual de la fuerza del clasificador, tal como se da en el "bucket brigade". Al parecer, los resultados experimentales tienen sin embargo a favorecer el uso del operador MAX.

En cuanto a la utilización de un factor de descuento $\gamma < 1$, ésta se ve justificada por el hecho de que efectivamente es necesario tener en cuenta las posibles recompensas futuras, pero también es conveniente favorecer aquellos caminos que sean más cortos. Pensemos en el uso de Q-learning sobre un espacio de estados con un estado de partida y un estado de finalización, existiendo diferentes caminos entre estos estados. Recorrer un camino u otro depende de la elección del conjunto de acciones sucesivas. Un Q-learning con un factor de descuento igual a 1, tenderá a estimar para todos los valores Q en los pares (estado, acción) el valor de la recompensa obtenida en la llegada al último estado (convergencia a un único valor), con lo que no habría forma de elegir entre las diferentes acciones de un estado. En cambio, la inclusión de un factor de descuento tiende a pesar menos la recompensa futura y más la recompensa inmediata, por lo que se favorecerán aquellos caminos que permitan obtener recompensa en los próximos pasos. Sin embargo, la utilización de $\gamma < 1$ o de $\gamma = 1$ también puede ser muy dependiente del problema. Por ejemplo en [Twardowski, 1993], se comprueba que si $\gamma = 1$, con lo que todos los valores van a converger a una misma cantidad, se favorece el que se produzca un comportamiento cíclico en el sistema, por lo que esta elección puede ser adecuada en problemas cuya solución tiene carácter cíclico.

Para entender un poco mejor la influencia del factor de descuento sobre el comportamiento del algoritmo vemos el siguiente ejemplo extraído de [Kovacs, 2002].

Supongamos el entorno descrito en la figura 5.3.

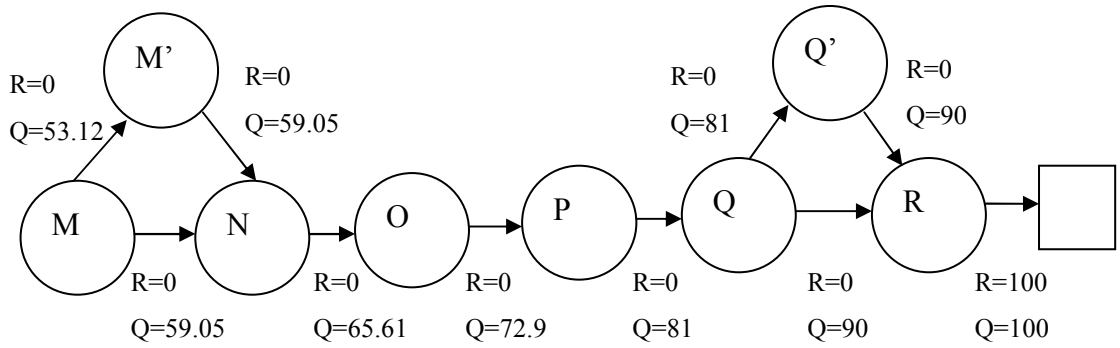


Figura 5.3. Entorno multi-paso para ejemplificar el efecto del descuento en el método de Q-learning.

Como vemos, en este entorno, la función de recompensas inmediatas se ha establecida de forma que todas las acciones producen una recompensa de 0 ($R = 0$), exceptuando la acción del estado R que produce una recompensa inmediata $R = 100$. Si no se utilizase una función Q, que tiene en cuenta las recompensas futuras, no se podría decidir qué acción tomar en los estados M y Q ya que las dos tendrían una fuerza de 0.

En el otro extremo, tenemos el uso de un factor de descuento $\gamma = 1$. Con este factor, la recompensa de la última transición iría pasándose completamente a las transiciones previas y sumándose a un valor de recompensa igual a 0. De esta manera, todas las transiciones adquirirían el mismo valor para Q igual a 100 (qué es la recompensa de la última transición). En esta situación, tampoco se podría favorecer una ruta particular en el grafo, ya que todas las acciones tomarían el mismo valor de fuerza.

Los valores de Q que se muestran en la figura se han conseguido con un factor de descuento de $\gamma = 0.9$. Como se puede comprobar, si partimos del estado M y en cada estado elegimos la acción con el mayor valor de Q, llegaremos al estado final por el camino más corto.

Sin embargo, la utilización del descuento sobre acciones futuras provoca fácilmente la posible aparición de clasificadores sobre-generales. Siguiendo con el ejemplo de la figura 5.3, supongamos una situación donde tenemos un clasificador X que encaja con el estado R y propone una transición al estado final (acción correcta), pero también encaja con el estado Q y propone una transición al estado Q' (acción incorrecta). De esta manera X es un clasificador sobre-general. Pensemos además que disponemos de otro clasificador que encaja con el estado Q y propone la transición a R.

Veamos que esta situación cumple la fórmula (c+i)/2>a, explicada anteriormente y que demuestra que X es un clasificador sobre-general fuerte: efectivamente (100 + 81)/2 > 90.

De forma más general, si c es la recompensa de la acción para la que el sobre-general es correcto, y la acción incorrecta se produce en un estado separado por un camino de m estados, recibirá una recompensa $i = c\gamma^m$. Además, si la acción correcta se produce en un estado separado por un camino de n estados, esta última recibirá una recompensa $a = c\gamma^n$. Entonces, para que el clasificador sobre-general sea fuerte, tendrá que cumplirse:

$$(c + c\gamma^m)/2 > c\gamma^n \tag{5.19}$$

Resulta que la condición es verdadera siempre que $c > 0$, $0 < \gamma < 1$ y $n \geq 1$. Esto se cumplirá casi siempre. Implica que la utilización del descuento en el Q-learning provoca funciones *Q* donde es posible la aparición de sobre-generales fuertes.

5.4.1.3 Del VSCS al CS.

El VSCS nos ha servido para comprender algunas diferencias entre el "bucket brigade" y el Q-learning, así como para comprender las implicaciones de estas técnicas en el ámbito de los clasificadores. Sin embargo, el VSCS es sólo una versión muy simplificada de un CS. Vamos a ir eliminando restricciones para explicar qué supone cada una de ellas. Para empezar, vamos a incluir más de una condición en los clasificadores y vamos a añadir a lista de mensajes nuevos "slots" o huecos.

A esta versión del VSCS lo denominaremos VSCS con memoria o VSCS-M. En el VSCS-M la lista de mensajes será:

a. Un hueco para el mensaje externo.
b. Un hueco para el mensaje de acción: se trata de un mensaje que especifica la acción a realizar sobre el entorno. El mensaje de acción, permite que el sistema tenga memoria de un paso.
c. El resto de la lista está destinada a mensajes internos, usados por ejemplo para describir diferentes estados internos del sistema.

Con esta lista de mensajes los clasificadores que encajan en un momento determinado producen dos clases de mensajes: los mensajes internos, y los mensajes de acción. Los mensajes de acción son reducidos a uno solo tras pasar por una etapa de resolución de conflictos. El mensaje ganador además de producir una acción se añade a la lista.

En el VSCS-M entendemos el papel que juegan los mensajes de acción y los mensajes internos en la construcción de una memoria a corto plazo, haciendo que el estado interno del CS juegue un papel explícito en la selección de los clasificadores.

Pasemos ahora a otro aspecto importante en el CS, su característica de no utilizar mapas completos. Esta capacidad se basa en poder construir clasificadores que *generalizan* sobre un conjunto de estados del sistema global equivalentes. El ingrediente que permite esta técnica es el elemento comodín o "*don't care*". Anteriormente ya se comentó que la utilización de clasificadores con comodines en sus condiciones crea una jerarquía donde puede manifestarse la problemática de cómo decidir cuando utilizar la regla más general o aplicar la excepción.

Cuando levantamos la última restricción impuesta sobre el VSCS, el cubrimiento total del espacio de estados, se hace necesario introducir un mecanismo de búsqueda. Este mecanismo de búsqueda tendrá una gran importancia en el sistema, especialmente en aquellos espacios de estados muy complejos. El mecanismo de búsqueda necesita de un conjunto de clasificadores que puedan adaptarse para recorrer las diferentes regiones del espacio de estados. Esta capacidad de los CS de modificar el conjunto de clasificadores da el carácter de *plasticidad* al sistema. En el caso del CS clásico de Holland, la plasticidad es implementada con un algoritmo genético, que es responsable de dos clases de cambios estructurales aquellos que afectan a la parte de la condición en el clasificador y aquellos que afectan a la parte de acción.

Este mecanismo de plasticidad basado en un algoritmo donde prima claramente el aspecto competitivo, y donde sobreviven sólo los mejores clasificadores es criticado por algunos autores [Dorigo y Bersini, 1994], que ven en otros marcos donde se incluyen aspectos cooperativos, como la generación de anticuerpos en el sistema inmunológico, técnicas más adecuadas [Bersini y Varela, 1994]. El aspecto cooperativo en la plasticidad también ha sido incorporado a sistemas basados en Q-learning [Bersini, 1993].

En resumen, tenemos que la memoria a corto plazo, el mecanismo de generalización y la plasticidad son las características añadidas al CS sobre el VSCS.

5.4.1.4 Descubrimiento de nuevas reglas.

Como ya hemos mencionado, es necesario proveer al sistema con algún método de innovación que introduzca reglas para atender a las situaciones novedosas que se van presentando y que de alguna forma se aproveche de los conocimientos previamente adquiridos durante el proceso de aprendizaje.

De esta tarea se responsabiliza al sistema descubridor de clasificadores, el cual tradicionalmente se compone de un algoritmo genético sobre los clasificadores que existen en la población en la iteración actual. Por lo tanto, el conjunto de reglas es la población, y cada regla es un individuo cuyo valor de aptitud está directamente relacionado con el valor de fuerza que tiene asociado. Aplicando operadores genéticos sobre esta población, se consiguen generar nuevas reglas o clasificadores, que se añaden al conjunto de reglas total, así como eliminar aquellos que han tenido menos éxito en su actividad de afrontar situaciones "leídas" del entorno. El sistema de descubrimiento puede contener otros métodos para el descubrimiento de reglas, como el denominado operador de recubrimiento que estudiaremos más adelante.

Este subsistema de descubrimiento hace que el enfoque descrito anteriormente y que engloba los CS en la corriente de los algoritmos RL no sea el único. En [Smith, 1992] se discuten los principales conceptos que definen un CS, y se toma una postura en la que se analiza al CS como una extensión de los algoritmos de optimización global basados en GA, discutiéndose las principales diferencias entre uno de estos algoritmos y un CS. Se argumenta que la diferencia proviene del carácter cooperativo de los procedimientos añadidos al mecanismo de descubrimiento. Así que tenemos estas dos visiones complementarias: por un lado el CS es un algoritmo de RL con un mecanismo de descubrimiento y por otro lado el CS es un sistema evolutivo que además incluye aspectos cooperativos entre los elementos de la población.

Pero, ¿qué entendemos por cooperación?. Una respuesta más precisa la tenemos en [Wilson, 1992]. Existen dos tipos de cooperación. La cooperación débil se establece cuando los clasificadores combinan sus efectos y ello redunda en la eficacia del sistema, sin afectar los valores de aptitud de los clasificadores. Es la situación que ocurre cuando

un clasificador cubre un conjunto de situaciones y otro clasificador cubre otro conjunto de situaciones disjunto con el anterior. Uno u otro clasificador encajará, dependiendo de la situación de la que se trate, y esto repercute en el comportamiento global del sistema. La cooperación fuerte se establece cuando los clasificadores combinan sus efectos, redundando en el comportamiento general del sistema y afectando a los valores de aptitud de los clasificadores.

Volviendo al ejemplo anterior, supongamos que se solapan los conjuntos de situaciones asociadas a los dos clasificadores. En las situaciones de solape, uno de los clasificadores puede actuar como la representación de una excepción de una regla general. La regla general vendría representada por el otro clasificador y su aplicación sin más en ciertas situaciones produciría un error. Gracias al clasificador excepción, el clasificador regla general no se aplica como tal, por lo que se está protegiendo a este último del error que se produciría cuando se presentan situaciones compatibles con la excepción. En este caso, la aptitud del clasificador más general se está viendo afectado por la existencia de la excepción.

La cooperación entre los elementos de la población marca diferencias importantes entre el algoritmo de optimización global basado en algoritmos genéticos y un CS. Para entender las diferencias, vamos a utilizar el experimento mental propuesto por Valenzuela-Rendón. Sea un problema de optimización a resolver mediante GA, donde la población se compone de soluciones x, teniendo cada solución una aptitud asociada que se calcula con una función f(x). En el esquema tradicional, cada solución x es codificada como una cadena de longitud L que representa una solución completa. Ahora imaginemos que cada elementos de la población se divide en n partes de longitud L/n. En cada subcadena marcamos los parámetros de la solución global a los que se refiere. De esta forma, una subcadena sirve para completar una parte de la solución. Imaginemos además que cada subcadena pasa a ser un miembro individual de la población. Tras este último paso hemos pasado del esquema tradicional a un esquema cooperativo. Bajo este esquema, los miembros de la población deben ser seleccionados y agrupados para dar cuenta de una solución completa, que pueda ser utilizada en la función f(x) para calcular la aptitud. Para obtener valores altos de la aptitud, los miembros de la población deben colaborar entre sí.

Usando este experimento, podemos dar cuenta de una primera diferencia. Los algoritmos de optimización tradicionales basados en GA deben evaluar cada una de las

soluciones completas que componen la población para calcular las aptitudes respectivas. En un GA podemos entender que hay un objetivo subyacente, que es producir estimaciones de la bondad de las subcadenas existentes dentro de las soluciones que forman la población, de manera que las mejores sobrevivan en el proceso evolutivo. Para lograr esto, es necesario ver su comportamiento en una gran variedad de *contextos* (en conjunción con otras subcadenas que forman parte de la solución), ya que si no, no se puede dar una estimación realista, puesto que la subcadena no actúa de forma aislada sino siempre formando parte de una solución. La variedad de contextos se consigue en los GA tradicionales mediante procesos de recombinación.

En un esquema cooperativo, las subcadenas son evaluadas en una variedad de contextos antes de aplicar el algoritmo genético. La variedad de contextos aquí se consigue mediante la selección de combinaciones de subcadenas. En determinadas aplicaciones se puede usar información particular del problema a resolver para evaluar la utilidad de la subcadena: por ejemplo, el parámetro X debe estar en el rango [0,R]. En el esquema cooperativo puede conseguirse una ventaja computacional, si se consigue reducir el número de evaluaciones de la aptitud a las estrictamente necesarias. En contraposición, hay que considerar si en el esquema cooperativo se mantiene el carácter global de la búsqueda realizada por el GA.

Además de la forma en que en cada estrategia se diversifican los contextos y las alternativas para estimar la bondad de las subcadenas, junto con las consecuencias computacionales que esto pueda suponer, otra diferencia clave está en la base sobre la que se realiza la búsqueda. La base para la búsqueda en el GA tradicional es el conjunto de soluciones completas. El GA tradicional explota las similaridades (esquemas) entre las soluciones con altos valores de aptitud. Por el contrario, en el caso de la estrategia cooperativa, la base para la búsqueda son las subcadenas o soluciones parciales. En este caso se explotan las similaridades entre soluciones parciales. Por ejemplo, si consideramos que cada solución parcial es una regla con su antecedente y consecuente, el algoritmo genético que se aplique tenderá a explotar similaridades relativas a los valores de los antecedentes que se utilizan (el concepto de esquema y similaridad se aplica al nivel de regla). Sin embargo, si se aplica el GA tradicional, la similaridad se busca a nivel de la solución completa. Como conclusión, si en un problema particular las soluciones parciales pueden beneficiarse mutuamente explorando sus similaridades, la estrategia cooperativa puede tener una ventaja añadida.

Esta discusión sobre los conceptos imbricados en un CS puede ser resumida en la definición de Wilson [Wilson, 1994]:

"Un sistema clasificador es un sistema de aprendizaje en el que un conjunto de reglas del tipo condición – acción compiten para controlar el sistema y ganar crédito basándose en la recepción de un refuerzo desde el entorno. El crédito acumulado por el clasificador, denominado fuerza, determina su influencia en el control de la competición y en los procesos evolutivos mediante algoritmos genéticos en los que nuevos, plausiblemente mejores, clasificadores son generados a partir de aquellos existentes que son fuertes, mientras que los más débiles son descartados."

5.4.1.5 Operaciones básicas en el CS.

A modo de resumen, esta es la secuencia de operaciones básicas desarrolladas en un CS.

1) Se crea un conjunto de clasificadores al azar o mediante algún algoritmo que tenga en consideración la estructura del problema. Se asigna la misma fuerza a todos los clasificadores.

2) El interfaz de entrada codifica las señales del entorno en mensajes.

3) Los mensajes del interfaz de entrada se añaden a la lista de mensajes.

4) El sistema determina el conjunto de clasificadores que encajan con los mensajes actuales de la lista.

5) Se resuelven los conflictos entre los clasificadores seleccionados y se determina el conjunto de clasificadores activos.

6) Se purga la lista de mensajes.

7) Los mensajes producidos por los clasificadores activos se colocan en la lista de mensajes.

8) La interfaz de salida se alimenta con los mensajes de acción de la lista. Si hubiera acciones incompatibles, se acude al módulo de resolución de conflictos para establecer la acción ganadora.

9) Se adquiere una señal de recompensa, que se pasa al sistema de asignación de créditos. Este sistema actualizará las fuerzas asociadas a los diferentes clasificadores.

10) Cuando el sistema de asignación de créditos alcanza el estacionario o está cerca de él, se aplica el sistema de descubrimiento (algoritmo genético) sobre el conjunto de clasificadores.

11) Regresamos al paso 2.

Como se desprende de estos pasos, el CS clásico se basa en la selección por el valor de la fuerza de los clasificadores, estimación de la bondad del clasificador mediante técnicas similares al aprendizaje con refuerzo, y por último, descubrimiento de nuevos clasificadores mediante algoritmos de búsqueda (por ejemplo, sistemas evolutivos). Uno de los puntos clave es el de la utilización de la estimación de la bondad del clasificador (fuerza) como base para la selección de los individuos de la población (aptitud basada en la fuerza).

En contraste con esta idea, tenemos el algoritmo XCS, que presenta como principal diferencia respecto al CS clásico la utilización de la estimación de la precisión de la predicción de la recompensa como aptitud.

5.4.1.6 El algoritmo XCS.

El concepto de CS ha sufrido extensiones, entre las que cabe destacar el algoritmo denominado XCS, propuesto en [Wilson, 1995]. Este algoritmo se diferencia del CS en varios aspectos. La fuerza de cada regla depende del error estimado en una predicción de la recompensa que el clasificador puede obtener. El algoritmo genético va a

funcionar sólo sobre subconjuntos de individuos de la población (conjuntos de acción) en lugar de hacerlo sobre toda la población.

Estas propiedades hacen del XCS un sistema con tendencia a encontrar clasificadores más precisos en cuanto a su adecuación a las diferentes situaciones del entorno. Además, este algoritmo incluye un mecanismo para recubrir de forma eficiente el espacio de los estados.

Se consideran aquí tres subsistemas: el programa de refuerzo, el entorno y el XCS. El entorno pasa al sistema una serie de situaciones sensoriales. Como respuesta, el sistema ejecuta un conjunto de acciones sobre el entorno. El resultado de cada acción es una recompensa escalar. En el diagrama de la figura 5.4, se muestra la interacción entre estos subsistemas.

En el caso de problemas de múltiples pasos, la bandera eop (*end of problem*) indica la finalización del problema. Mientras $\sigma(t)$ y $\alpha(t)$ son interacciones con el entorno en sí mismo, $\rho(t)$ y eop provienen de un subsistema que denominaremos "Programa de refuerzo" (PR). El PR determina la recompensa conforme a la entrada del entorno y a las acciones ejecutadas $(\sigma(t),\alpha(t))$. La separación del PR del entorno conceptualiza el hecho de que la recompensa viene dada por las propias características del entrenamiento.

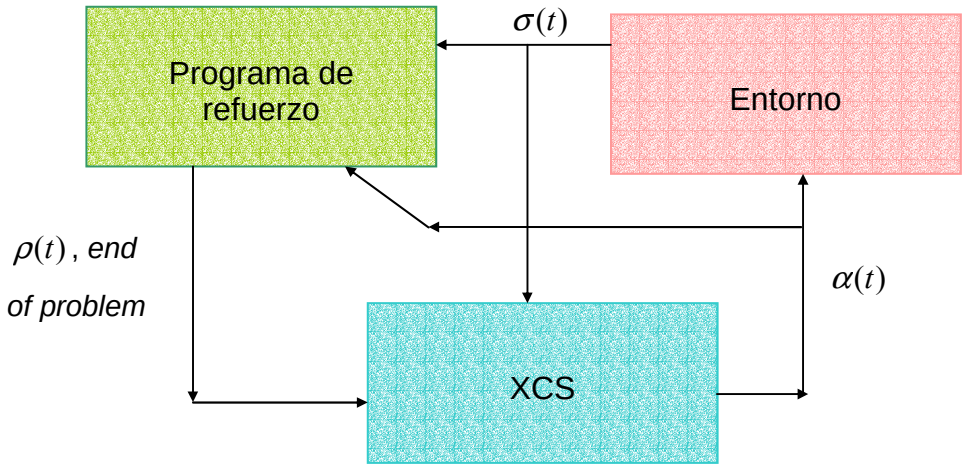


Figura 5.4. Estructura básica del algoritmo XCS.

Recordemos que en un problema de un paso las sucesivas situaciones no están relacionadas unas con otras. En ese caso, después de cada acción, el PR da lugar a la recompensa y señala con la bandera eop que el problema ha terminado. En un problema de múltiples pasos, como un laberinto, las situaciones sucesivas se relacionan entre sí.

La recompensa sólo se recibirá en situaciones particulares. El punto de finalización de la tarea, debe establecerse por el PR conforme a la propia definición de la misma.

El XCS utiliza los siguientes conjuntos en las diferentes fases del algoritmo:

1. La población $[P]$, formada por los clasificadores en la iteración t.

2. El conjunto de encaje $[M]$, formado a partir de $[P]$, por los clasificadores que encajan con la situación actual $\sigma(t)$.

3. El conjunto de acción (action set) $[A]$, formado a partir de $[M]$. Incluye a los clasificadores de $[M]$ que proponen la acción actual.

4. El conjunto de acción previo $[A]_{-1}$: el conjunto de acción en el último ciclo de ejecución.

Veamos ahora la definición de un clasificador en XCS. Está formado por las siguientes partes:

- La condición $C \in \{1,0,\#\}^L$, que especifica las situaciones sensoriales en las que el clasificador puede ser aplicado.

- La acción $A \in \{a_1, a_2, ..., a_n\}$ propuesta por el clasificador.

- La predicción p estima la recompensa esperada cuando el clasificador encaja y la acción propuesta por él es aplicada sobre el entorno.

Además, cada clasificador tiene los siguientes parámetros asociados:

- La estimación del error de predicción ε.

- La aptitud f del clasificador.

- La experiencia $\exp$, que cuenta el número de veces desde la creación del clasificador que éste ha pertenecido al conjunto de acciones.

- La marca de tiempo t_s que guarda el paso (iteración) en el que por última vez se aplicó un algoritmo genético en el conjunto de acciones al que perteneció el clasificador.

- El tamaño del conjunto de acción as, que estima el promedio de los tamaños de los conjuntos de acciones al que el clasificador perteneció.

- La *numerosidad* (*numerosity*) *num*, que refleja el número de microclasificadores representados por este clasificador.

Nos referiremos genéricamente a una propiedad del clasificador como x_{cl}, donde $x \in \{C, A, p, \varepsilon, f, \exp, t_s, as, num\}$.

La recompensa de un clasificador (*payoff*) no sólo se refiere a la recompensa inmediata ρ proporcionada por el entorno, sino que es una combinación de la misma y de la predicción de recompensa con la mejor acción posible en el siguiente estado. Para problemas de un paso, el payoff se reduce a la recompensa inmediata.

Una vez introducidos los diferentes elementos del algoritmo XCS, pasaremos a realizar su descripción algorítmica.

1. Inicialización.

Corresponde a la inicialización de los módulos: el entorno, el PR y el propio XCS. En el XCS hay que iniciar el contador de iteraciones t, los parámetros de aprendizaje y la población de clasificadores $[P]$. Esta última podría inicializarse como un conjunto vacío o al azar.

En pseudocódigo, los pasos de cualquier algoritmo XCS son:

1. Inicializar entorno.
2. Inicializar el programa de refuerzo PR.
3. Inicializar el XCS.
4. Ejecutar el experimento.

Ejecutar el experimento se refiere al lazo principal del programa.

2. Bucle principal del algoritmo.

En el bucle principal del algoritmo se realizan las siguientes tareas:

1. Se adquiere la situación actual en el entorno.
2. Se forma el conjunto de encaje $[M]$, con todos los clasificadores que se adecuan a la situación actual.
3. Se construye el denominado vector de predicción *VP* basado en los clasificadores del conjunto de encaje $[M]$. En este vector se trata de estimar el resultado de las acciones a_i de los clasificadores de $[M]$, es decir su *payoff*, considerado como se describió anteriormente.
4. Se utiliza el VP para elegir una acción a ejecutar. Con esta acción seleccionada se forma el conjunto de acción $[A]$, añadiendo todos los clasificadores de $[M]$ que den lugar a la acción. Cada vez que un clasificador se añade al conjunto $[A]$, sus parámetros exp , p , ε y f son modificados.
5. Se ejecuta la acción ganadora.
6. Si el problema es multipaso: el conjunto de acción previo $[A]_{-1}$ se modifica utilizando una cantidad P de payoff que es una combinación de la recompensa anterior ρ_{-1} y la mayor predicción en VP. Además se podría aplicar un algoritmo genético sobre $[A]_{-1}$.
7. Si un problema termina (en cada paso para problemas de un solo paso o en el último paso para problemas multi-paso), $[A]$ es modificado conforme a la recompensa ρ y es posible que se aplique un algoritmo genético sobre $[A]$.

Una vez alcanzado el paso 7, volvemos al paso 1. El lazo principal termina tras cumplirse un criterio de terminación.

Vamos a explicar ahora de forma más detallada, algunos subprocesos del lazo principal del programa.

3. Formación del conjunto de encaje.

La formación del conjunto de encaje toma con entrada la población actual $[P]$ y la situación en el entorno σ.

Los pasos para construir el conjunto de encaje son:

1. Inicializa el conjunto $[M]$ como un conjunto vacío.
2. Bucle que se repetirá siempre que $[M]$ esté vacío:
    a. Bucle que se repite para cada clasificador $cl \in P$.
        i. Si el clasificador cl encaja en la situación σ, añadir el clasificador cl en el conjunto $[M]$.
    b. Si el número de acciones diferentes en $[M]$ es menor que el valor del parámetro θ_{mna} entonces:
        i. Generar un clasificador de recubrimiento cl_c considerando $[M]$ y σ.
        ii. Añadir el clasificador cl_c al conjunto $[P]$.
        iii. Realizar una operación de borrado sobre la población $[P]$.
        iv. Vaciar el conjunto $[M]$.

En este pseudocódigo debemos aclarar las siguientes operaciones: determinación del encaje de un clasificador con la situación actual, generación de un clasificador de recubrimiento y la operación de borrado sobre la población.

Comencemos por la determinación del encaje. El pseudocódigo se muestra a continuación. Las entradas son cl, σ.

1. Para cada atributo x en el clasificador C_{cl}:

   a. Si $x \neq \#$ y $x \neq$ al atributo correspondiente en σ, se considera que no hay encaje y termina la comprobación.
2. Si se llega a este punto, se considera que se ha producido el encaje.

Como se muestra, el encaje incluye el uso del símbolo $\#$ como elemento comodín.

Veamos ahora como se genera un clasificador de recubrimiento. El pseudocódigo toma como entradas $[M]$ y σ. Los pasos son descritos a continuación.

1. Inicializar el clasificador cl. Tener en cuenta que la condición C_{cl}, debe tener la longitud de σ.
2. Para cada atributo x de C_{cl} haya que realizar lo siguiente:
   a. Con probabilidad $P_{\#}$ hacer que x sea un comodín $\#$.
   b. Si no se ha asignado x como $\#$, hacer que tome el valor del atributo correspondiente en σ .
3. Hacer que A_{cl} tome el valor de una acción no presente en $[M]$. La asignación se hace al azar.
4. Inicializar los valores p_{cl}, ε_{cl}, f_{cl} con los valores por defecto establecidos. Inicializar la experiencia del clasificador $\exp_{cl}$ a 0. Hacer que la marca de tiempo t_s tome el valor del tiempo actual t. Inicializar el número de acciones a_{cl} a 1, así como la numerosidad num_{cl} a 1.

Como se desprende del pseudocódigo, la generación de un clasificador de recubrimiento asegura un nuevo elemento para la población $[P]$ que encaja con la situación del entorno σ y con una parte de acción nueva en el conjunto $[M]$, aumentando así en 1 el número de acciones diferentes consideradas. El número de elementos comodín de la parte de condición del clasificador viene regulado por la probabilidad $P_{\#}$.

En cuanto a la operación de borrado de la población, se pretenden dos objetivos:

1) Asegurar aproximadamente el mismo número de clasificadores en cada conjunto de acciones.
2) Eliminar individuos con un valor de aptitud demasiado bajo.

Antes de pasar al pseudocódigo de la operación de borrado, vamos a tratar el concepto de macroclasificador. Sin el uso de este concepto, en una población se puede tener el clasificador *cl* repetido un número de veces N. El concepto de macroclasificador permite aumentar la eficiencia computacional del algoritmo, sustituyendo las N repeticiones del clasificador *cl*, por el denominado macroclasificador compuesto por los atributos de *cl* y el parámetro numerosidad *num* con valor N. Las diferentes operaciones realizadas sobre la población de clasificadores deben realizar los cambios oportunos en el parámetro de numerosidad para que el macroclasificador mantenga la coherencia con lo que representa. Cada clasificador "copia" representado por el parámetro numerosidad en el macroclasificador se denomina microclasificador.

Este es el pseudocódigo de la operación de borrado:

1. Si el número de microclasificadores ($\sum_{c\in[P]} num_c$) es inferior al parámetro N del algoritmo, no realizar la operación de borrado.
2. Calcular el promedio de la aptitud de la población: avFitness= $\sum_{c\in[P]} f_c / \sum_{c\in[P]} num_c$.
3. Inicializar el parámetro "suma de votos" a 0: voteSum=0.
4. Para cada clasificador $c \in [P]$,
    a. Incrementar el parámetro "suma de votos" (voteSum) con los denominados votos de borrado del clasificador c calculados a partir de la aptitud del clasificador y de la aptitud promedio de la población. Posteriormente veremos como se realiza este cálculo.
5. Calcular el punto de corte (choicePoint) como un punto del intervalo [0, voteSum) elegido al azar, y suponiendo una distribución uniforme.
6. Inicializar el parámetro "suma de votos" a 0: voteSum=0.
7. Para cada clasificador $c \in [P]$

a. Incrementar el parámetro "suma de votos" (voteSum) con los denominados votos de borrado del clasificador c calculados a partir de la aptitud del clasificador y de la aptitud promedio de la población.
b. Si voteSum es mayor que el punto de corte (choicePoint) borrar un microclasificador. Esto se realiza del siguiente modo para el macroclasificador c:
    i. Si la numerosidad de c es mayor que 1, disminuimos en 1 su parámetro de numerosidad.
    ii. En caso contrario eliminamos el macroclasificador de la población $[P]$.

El sistema de borrado descrito elige al macroclasificador que sufre la operación de borrado mediante un método de selección equivalente a una ruleta. Aquellos clasificadores con un mayor número de votos de borrado tendrán proporcionalmente más probabilidad de decrementar su numerosidad en una unidad.

Veamos ahora el pseudocódigo que permite calcular el número de votos de borrado para un clasificador. Los parámetros de entrada son el clasificador y el promedio de aptitud de la población avFitnessinPopulation.

1. Inicializamos la variable número de votos de borrado vote con el producto de as_{cl} (tamaño del conjunto de acción del clasificador cl) y la numerosidad num_{cl}.
2. Si la experiencia del clasificador cl es superior al valor del parámetro θ_{del} y la aptitud promedio de cada microclasificador de cl (f_{cl} / num_{cl}), es inferior al valor umbral obtenido a partir de la aptitud promedio de la población multiplicado por el parámetro δ, entonces el número de votos del clasificador se incrementa en el factor multiplicativo avFitnessinPopulation / (f_{cl} / num_{cl}).

Como vemos, el borrado es fomentado en aquellos clasificadores que tienen una experiencia suficientemente elevada (así se evita borrar a clasificadores recién creados que no han tenido tiempo aún de ser recompensados) pero sus microclasificadores tienen una aptitud promedio demasiado baja.

4. El vector de predicción.

El algoritmo XCS debe realizar una estimación de las recompensas que se pueden conseguir tras aplicar cada acción. La predicción para una acción es un promedio de la recompensa esperada por los clasificadores de $[M]$ que son productores de la acción pesada por la aptitud de cada uno. Si una acción no es invocada por ningún clasificador no se realiza una predicción y entonces se simboliza por un símbolo NULL.

El pseudocódigo toma como argumento de entrada el conjunto de encaje $[M]$.

1. Comenzamos por inicializar el vector de predicción VP a NULL. A continuación declaramos e inicializamos a cero, un vector que denominaremos vector de suma de fitnness (FSA).
2. Para cada clasificador $cl \in [M]$:
    a. Si $VP[A_{cl}] = NULL$, entonces
        i. $VP[A_{cl}] = p_{cl} * f_{cl}$
    b. en caso contrario
        i. $VP[A_{cl}] = VP[A_{cl}] + p_{cl} * f_{cl}$
    c. $FSA[A_{cl}] = FSA[A_{cl}] + f_{cl}$
3. Para cada posible acción A .
    a. Si $FSA[A]$ es diferente de 0
        i. $VP[A] = VP[A] / FSA[A]$

5. Elegir una acción.

El método de selección de acciones no está prefijado en el XCS. El siguiente pseudocódigo muestra la combinación de exploración pura y explotación pura. El primer método es búsqueda aleatoria, mientras que el segundo se basa en seleccionar la mejor de las acciones. El argumento de entrada para este pseudocódigo es el vector de predicción VP.

1. Realizar exploración pura con probabilidad $p_{\exp lr}$. Consiste en elegir una acción de entre aquellas en el vector de predicción que no tengan asociado un valor nulo.
2. Si no se realiza exploración pura, elegir la acción con el valor más alto en el vector de predicción.

6. Construcción del conjunto de acción.

Recordemos que el conjunto de acción $[A]$ está formado por los clasificadores de $[M]$ que producen la acción seleccionada act. Teniendo en cuenta este objetivo, el pseudocódigo que permite realizar esta tarea es bastante sencillo.

1. Inicializar $[A]$ como un conjunto vacío.
2. Para cada clasificador cl en $[M]$.
    a. Si $A_{cl} = act$
    b. Añadir el clasificador cl al conjunto $[A]$.

7. Actualización de los parámetros del clasificador.

Cada vez que un clasificador se añade al conjunto $[A]$, sus parámetros $\exp$, p, ε, as y f son modificados. Esta parte del algoritmo toma como entradas $[A]$, $[P]$ y el parámetro P. Si el problema es multipaso, P se calcula a partir de la recompensa obtenida en el paso anterior ρ_{-1} y la máxima recompensa en el vector de predicción, mediante la siguiente fórmula:

$$P = \rho_{-1} + \gamma \max[VP] \tag{5.20}$$

Para problemas de un paso, P es la recompensa en el paso actual ρ.

1. Para cada clasificador cl en $[A]$
    a. Incrementar su parámetro de experiencia: $\exp_{cl}++$

b. Ahora vamos a actualizar la predicción de la recompensa p_{cl}.
   i. Si la experiencia es suficientemente pequeña ($\exp_{cl} < 1/\beta$), cambiar la predicción usando la siguiente fórmula: $p_{cl} = p_{cl} + (P - p_{cl}) / \exp_{cl}$.
   ii. En caso contrario: $p_{cl} = p_{cl} + \beta * (P - p_{cl})$.
c. Pasamos a la actualización de la estimación del error de predicción ε_{cl}.
   i. Si la experiencia es suficientemente pequeña ($\exp_{cl} < 1/\beta$), cambiar el error de predicción usando la siguiente fórmula: $\varepsilon_{cl} = \varepsilon_{cl} + (|P - p_{cl}| - \varepsilon_{cl}) / \exp_{cl}$
   ii. En caso contrario $\varepsilon_{cl} = \varepsilon_{cl} + \beta * (|P - p_{cl}| - \varepsilon_{cl})$
d. Pasamos a la actualización de la estimación del tamaño del conjunto de acciones as_{cl}.
   i. Si la experiencia es suficientemente pequeña ($\exp_{cl} < 1/\beta$), $as_{cl} = as_{cl} + (\sum_{c \in [A]} num_c - as_{cl}) / \exp_{cl}$.
   ii. En caso contrario $as_{cl} = as_{cl} + \beta * (\sum_{c \in [A]} num_c - as_{cl})$.
e. Una vez actualizados los parámetros se recalculará la aptitud de los elementos de $[A]$ (esta parte se describe más abajo).
f. Este paso es opcional y se configura mediante una bandera. Realizar el proceso de subsumisión sobre $[A]$ manteniendo la coherencia con $[P]$.

Observamos que la predicción de los parámetros se basa en una aproximación lineal que realiza una corrección basada en el error actual. Si x_t es el valor del parámetro en el paso t, y $\hat{x}_t$ es la estimación realizada para el paso t, entonces la estimación para el paso $t+1$, se realiza del siguiente modo:

$$\hat{x}_{t+1} = \hat{x}_t + C(x_t - \hat{x}_t) \tag{5.21}$$

Este método para estimar la predicción de la recompensa, ocasiona una variabilidad considerable en la predicción de la estimación de la recompensa y un

aumento en la estimación del error, cuando el clasificador sobre el que se aplica es compatible con varios estados donde los valores de recompensa que se asignan son muy diferentes. Posteriormente analizaremos en mayor profundidad este efecto, de importantes consecuencias para los clasificadores sobre-generales y su relación con el sesgo en la función de recompensa.

Veamos ahora la actualización de la aptitud de los clasificadores incluidos en el conjunto de acciones $[A]$.

1. Inicializar el parámetro acumulador_de_precisión a 0.
2. Inicializar el vector de valores de precisión κ.
3. Para cada clasificador en $[A]$
    a. Si el error del clasificador ε_{cl} es menor que el parámetro ε_0 hacer $\kappa(cl) = 1$.
    b. En caso contrario $\kappa(cl) = \alpha(\varepsilon_{cl} / \varepsilon_0)^{-\nu}$.
    c. Incrementar el valor del parámetro acumulador_de_precisión en la cantidad $\kappa(cl) * num_{cl}$.
4. Para cada clasificador en $[A]$
    a. Actualizamos la aptitud del clasificador mediante la siguiente fórmula: $f_{cl} = f_{cl} + \beta * (\kappa(cl) * num_{cl} / \text{acumulador\_de\_precisión} - f_{cl})$.

Como vemos, la aptitud de cada clasificador se considera una función del error de precisión. La fórmula tiene en cuenta el concepto de macroclasificador, considerando que se asigna la misma medida de precisión ($\kappa(cl) / \text{acumulador\_de\_precisión}$) a cada microclasificador, de forma que la medida de precisión del macroclasificador se obtiene tras multiplicar esta cantidad por la numerosidad num_{cl}. La actualización de la aptitud se realiza con la misma fórmula de adaptación lineal, de forma que se considera que la aptitud de cada clasificador es un predicción de la medida de precisión en el siguiente paso. El parámetro de adaptación es β.

La medida de precisión del paso actual es una cantidad entre 0 y 1. Si la estimación del error de predicción está por debajo de cierto umbral preestablecido ε_0, la

medida de precisión toma su valor máximo que es 1. Si no es así, sufre un decaimiento exponencial dado por la fórmula $\kappa(cl) = \alpha(\varepsilon_{cl} / \varepsilon_0)^{-\nu}$.

Pasemos ahora a explicar el proceso de subsumisión. El conjunto de acción se revisa a la búsqueda del clasificador más general que sea además suficientemente experimentado y preciso. Entonces, el resto de clasificadores se revisan para ver si son englobados por el más general. Aquellos que son englobados serán eliminados de la población.

Veamos el proceso de subsumisión que se lleva a cabo sobre el conjunto de acción.

1. Inicializar un clasificador cl.
2. Búsqueda del clasificador subsumidor. Para cada clasificador c en $[A]$:
    a. Comprobar si el clasificador c reúne las condiciones previas para subsumir otros clasificadores: debe tener una experiencia suficientemente elevada ($\exp_c > \theta_{sub}$) y tener un error de predicción suficientemente pequeño ($\varepsilon_c < \varepsilon_0$). Si se cumplen estas condiciones:
        i. Ante cualquiera de estas circunstancias: cl está aún vacío o el número de comodines en la condición del clasificador c es mayor o igual que el número de comodines en el clasificador cl: entonces con probabilidad 0.5 asignar cl a c.
3. Eliminación de aquellos clasificadores que pueden ser subsumidos. Si cl no está vacío (se ha encontrado uno que reúne las condiciones), para cada clasificador $c \in [A]$:
    a. Si cl es más general que c (veremos después como se realiza esta comprobación):
        i. Adecuamos la numerosidad de cl: $num_{cl} = num_{cl} + num_c$.
        ii. Eliminamos el clasificador c del conjunto $[A]$.
        iii. Eliminamos el clasificador c del conjunto $[P]$.

La comprobación de si un clasificador cl_{gen} es más general que otro cl_{espec} se realiza del siguiente modo:

1. Si el número de comodines en Ccl_{gen} es menor que en Ccl_{espec} entonces decidimos que cl_{gen} no es más general que cl_{espec}.
2. Si la condición Ccl_{gen} y Ccl_{espec} encajan (coinciden los atributos correspondientes o el atributo de Ccl_{gen} es un comodín) entonces se concluye que cl_{gen} es más general que cl_{espec}.

8. El algoritmo genético en XCS.

La aplicación del algoritmo genético en XCS no se produce siempre. Es necesario que el denominado tiempo medio transcurrido desde la última modificación sea superior al parámetro θ_{GA}.

El siguiente cálculo permite estimar el tiempo medio transcurrido desde la última modificación:

$$\tau = t - \frac{\sum_{cl \in [A]} t_{s_{cl}} * num_{cl}}{\sum_{cl \in [A]} num_{cl}} \tag{5.22}$$

Si se da la condición para la aplicación del algoritmo genético los pasos a seguir son:

1. Para cada clasificador cl en $[A]$:
   a. Asignar el tiempo actual t al parámetro $t_{s_{cl}}$.
2. Seleccionar primer progenitor en $[A]$: progenitor1. El proceso de selección de progenitores se detallará posteriormente.
3. Seleccionar segundo progenitor en $[A]$: progenitor2.
4. Hacer una copia del clasificador progenitor1 en descendiente1.
5. Hacer una copia del clasificador progenitor2 en descendiente2.
6. Inicializar la numerosidad del descendiente1 y del descendiente2 a 1. $num_{desc1} = num_{desc2} = 1$.

7. Inicializar la experiencia del descendiente1 y del descendiente2 a 0. $\exp_{desc1} = \exp_{desc2} = 0$.
8. Con probabilidad χ:
    a. Realizar operación de cruce con descendiente1 y descendiente2. La operación de cruce se describirá posteriormente.
    b. Modificación de parámetros de los descendientes:
        i. La predicción de recompensa y la estimación del error de ambos clasificadores se toma como el promedio de los parámetros correspondientes de los progenitores.
        ii. La aptitud de ambos clasificadores descendientes se toma como el promedio de las aptitudes de los clasificadores progenitores reducido en un factor 0.1.
9. Para ambos descendientes:
    a. Aplicar operación de mutación usando el descriptor de la situación actual σ. La operación de mutación será descrita posteriormente.
    b. Este paso es opcional. Se trata de un proceso de subsumisión de los progenitores sobre los descendientes. Ocurre cuando un progenitor representa un clasificador cuya condición engloba a la del descendiente y además el padre es suficientemente preciso y con una experiencia alta. Si esto sucede el descendiente no es añadido a la población y el progenitor incrementa su numerosidad. El proceso se realiza del siguiente modo:
        i. Se comprueba si el progenitor1 puede subsumir al descendiente que se está considerando. Para ello es necesario que las acciones de ambos clasificadores sean iguales. También es necesario que la experiencia del progenitor sea superior al parámetro θ_{sub} y la estimación del error sea inferior a ε_0. Por último se comprueba que el progenitor1 sea más general que el descendiente (de forma análoga a como se hizo en el proceso de subsumisión aplicado al conjunto de acción). Si todas las condiciones se cumplen se incrementa la numerosidad del progenitor1.
        ii. Si no se cumple i por alguna de las razones se comprueba exactamente lo mismo para el progenitor2. Si se cumplen las

condiciones del proceso de subsumisión para el progenitor2 , incrementamos la numerosidad del progenitor2.

iii. Si no se cumpla ni i, ni ii insertamos al descendiente en la población

10. Realizamos un proceso de borrado de la población $[P]$. El proceso de borrado se explicará posteriormente.

Vamos a describir ahora las subtareas de los pasos descritos anteriormente. Comencemos con la selección de progenitores en el conjunto de acción $[A]$.

1. Inicializamos una variable denominada acumulador_fitness a 0.
2. Para cada clasificador cl en $[A]$
    a. Incrementar acumulador_fitness con el valor de aptitud de cl.
3. Inicializar la variable punto_de_corte con un número aleatorio entre 0 y acumulador_fitness usando una distribución uniforme.
4. Inicializar la variable acumulador_fitness a 0.
5. Para cada clasificador cl en $[A]$
    a. Incrementar acumulador_fitness con el fitness de cl.
    b. Si la variable acumulador_fitness es mayor que el valor de punto_de_corte
        i. Elegir cl como progenitor y salir de esta subtarea.

Como ya mencionamos anteriormente esta es una implementación del método de selección de la ruleta.

Pasemos ahora a la operación de cruce entre los clasificadores cl_1 y cl_2:

1. Elegir al azar dos puntos de corte x, y (índices sobre los atributos) en la parte de condición.
2. Si x>y intercambiar los valores de x e y.
3. Para todos aquellos atributos que ocupen la posición i con $x \leq i < y$ intercambiar los atributos $C_{cl_1}[i]$ y $C_{cl_2}[i]$.

A diferencia del cruce, que sólo tiene lugar en la parte de condición, la mutación puede tener lugar tanto en la condición como en la acción. La mutación en la parte de condición cambiará el valor del atributo seleccionado a una de las siguientes posibilidades: comodín o el valor del atributo correspondiente en σ. Veámoslo más detalladamente:

1. Para cada atributo de la parte de la condición del clasificador cl:
    a. Con probabilidad μ, si el atributo es un comodín, cambiar por el valor del atributo correspondiente en σ, y en caso contrario cambiar por un comodín.
2. Con probabilidad μ, cambiar la acción del clasificador por cualquiera de las otras acciones posibles.

Estos son los operadores genéticos usados en el XCS. Ahora terminemos la descripción con las subtareas generales "insertar clasificador en la población" y "proceso de borrado de la población".

La inserción de un clasificador en la población no puede limitarse a añadir un elemento al conjunto $[P]$, ya que hay que mantener la coherencia del parámetro numerosidad de los clasificadores. Para la inserción de cl en $[P]$ se realizan los siguientes pasos:

1. Para cada clasificador c de la población $[P]$:
    a. Si c es igual cl en la condición y la acción, incrementamos la numerosidad de c: num_c y terminamos la subtarea.
2. Añadimos cl a $[P]$.

Con esto, quedan descritos los aspectos más importantes del algoritmo XCS. Algunas de las técnicas presentadas aquí se han usado en el sistema tipo Michigan utilizado en la búsqueda del clasificador basado en una máquina de estados borrosa que posteriormente analizaremos, éste es el motivo por el que se ha revisado con detalle el algoritmo XCS.

5.4.1.7 Discusión sobre el algoritmo XCS.

En el punto anterior se ha realizado una descripción más o menos exhaustiva de este algoritmo. Aquí vamos a extraer algunas conclusiones de los puntos más importantes.

Como ya se comentó, la principal diferencia de XCS con los CS clásicos es la utilización de una aptitud basada en la precisión de la predicción de la recompensa, en lugar de en la propia recompensa. Este hecho ha sido analizado por diferentes autores en los últimos años, obteniendo algunas conclusiones relevantes:

1. La estrategia del XCS parece más adecuada para afrontar problemas multi-paso, todo lo contrario que los sistemas que basan la aptitud del clasificador en la recompensa esperada.

2. Se han obtenido resultados experimentales en los que los sistemas XCS tienen mejores propiedades de generalización, respecto a los CS tradicionales.

3. Los sistemas XCS parecen mejores respecto al denominado dilema exploración /explotación y al problema de los clasificadores sobre-generales.

Una primera consecuencia de usar la precisión de la predicción es que el algoritmo tenderá a mantener en la población los clasificadores con precisión y por tanto consistentes, tanto aquellos con una recompensa elevada, como aquellos con una recompensa baja. En el CS clásico, la población recogerá sólo a los clasificadores consistentes con altas recompensas. Este hecho, motiva que en el XCS sea normalmente necesario mantener una población con un mayor número de reglas, y por tanto se requiera mayor carga computacional.

Los clasificadores sobre-generales.

Para analizar comparativamente el XCS con el CS clásico, es preciso profundizar en las dificultadas encontradas en relación a la utilización de la aptitud basada en la fuerza.

Anteriormente, ya se mencionaron algunos de ellos: el clasificador sobre-general, el clasificador acaparador y el clasificador sobre-general fuerte. Fue entonces cuando se expuso que una causa para la aparición de estos problemas es la existencia de

funciones de recompensa con sesgo, es decir sistemas en los que la recompensa (penalización) depende fuertemente del estado particular.

Por aclarar el concepto, supongamos un clasificador sobre-general que actúa correctamente sobre 10 estados con recompensa c, e incorrectamente sobre un solo estado con recompensa i. Entonces su fuerza tenderá a ser (10c + i)/11. Se convertirá en un sobre-general fuerte si este valor es suficientemente grande (supera cierto valor a). Este valor será superior al valor de recompensa de otro clasificador que actúe correctamente sobre el estado donde el sobre-general fue incorrecto si los valores de recompensa en los estados donde el sobre-general actúa correctamente son superiores, es decir, si la función de recompensa tiene sesgo.

Existen otros factores relacionados con la problemática de los clasificadores sobre-generales:

- Para empezar, el hecho básico de que los clasificadores se aplican a un conjunto de estados en lugar de a uno solo como ocurre en los sistemas RL.

- La estrategia que combina exploración con explotación y que afecta al modo en el que se adaptan las recompensas de los clasificadores.

- La frecuencia con la que cada estado particular es visitado.

- Los mecanismos de selección, en concreto la importancia de la presión de selección que se establece en la reproducción y borrado de los individuos.

- La estructura del espacio de estados, que puede determinar hasta que punto los sobre-generales fuertes compiten con reglas correctas.

Entornos multi-paso.

Hemos visto que los problemas multi-paso implican la elección de acciones que deben tener en cuenta, no sólo la recompensa inmediata sino también la recompensa futura. Dos algoritmos que pueden afrontar este problema son el bucket brigade y el Q-learning.

En concreto, recordemos que el Q-learning calcula la estimación de la recompensa para la acción en un estado como la recompensa inmediata y la máxima de las recompensas del siguiente estado con un factor de descuento γ. A medida que este factor se aproxima a cero, se pesa más la recompensa inmediata favoreciéndose así los caminos más cortos, pero teniendo menos en cuenta la recompensa futura. Como hemos comentado anteriormente, este hecho tiene sus implicaciones sobre la aparición de clasificadores sobre-generales, ya que se obtienen funciones de recompensa con sesgo, ésta es la forma en que determinados caminos son favorecidos frente a otros.

Como resumen de las dificultades encontradas en los sistemas tradicionales con aptitud basada en la fuerza de los clasificadores tenemos:

- Las reglas son pesadas en proporción a la estimación de la recompensa inmediata o en entornos multipaso la recompensa inmediata y futura. Esto introduce el problema de la creación de clasificadores acaparadores.

- Aparecen clasificadores sobre-generales que son más fuertes que otros correctos con los que compiten. Además es difícil predecir en un entorno concreto la magnitud de este problema.

- En entornos de un paso con funciones de recompensa con sesgo, la aparición de clasificadores sobre-generales puede ser un efecto importante.

- También puede haber problemas en entornos multi-paso con funciones Q no triviales, dado que podrían surgir un gran número de clasificadores sobre-generales, como hemos descrito anteriormente.

Los clasificadores sobre-generales en los sistemas basados en la precisión de la predicción.

Es precisamente en la sensibilidad del sistema al sesgo de la función de recompensa donde los sistemas basados en la precisión de la predicción de la recompensa como el XCS adquieren sus mayores ventajas. Mientras en un CS clásico existe una dependencia

notable, como hemos visto, y esta dependencia causa la aparición de clasificadores sobre-generales, un XCS es insensible al sesgo de la función de recompensa.

El análisis de los sobre-generales fuertes en el caso del XCS se realizó por Kovacs [Kovacs, 2002], comenzando por una nueva interpretación del concepto de sobre-general para sistemas donde la aptitud se basa en la precisión de la predicción. Un clasificador se interpreta como sobre-general si el algoritmo XCS le asigna una variedad de valores de recompensa en diferentes estados. Tratemos de explicar esto mejor con un ejemplo.

Supongamos nuevamente un entorno con dos estados y sólo dos posibles acciones. La tabla 5.4 define su función de recompensa.

Estado	Acción	Recompensa
0	0	1000
0	1	0
1	0	1000
1	1	2000

Tabla 5.4. Definición de una función de recompensa.

Como se puede ver, se trata de una función de recompensa con sesgo: en el estado 1, la acción correcta, es decir aquella que recibe una recompensa mayor tiene asignado un valor diferente que la recompensa correspondiente en el estado 0. Lo mismo ocurre con las acciones incorrectas.

Sea el clasificador E: (#, 0). Esta regla encaja con ambos estados y produce la acción 0, por lo que es correcto en el estado 0 e incorrecto en el estado 1. Se trata por lo tanto de un clasificador sobre-general desde el punto de vista de la definición dada para los sistemas con aptitud basada en la fuerza.

Sin embargo no se trata de un clasificador sobre-general bajo la definición dada en el caso de sistemas con aptitud basada en la precisión: tanto en un estado como en otro, la recompensa recibida por este clasificador es la misma. Como vemos, la nueva definición de sobre-general es un poco más restrictiva, ya que exige además valores "suficientemente diferentes" para la recompensa obtenida en los diferentes estados.

El motivo de esta restricción adicional en la definición de clasificador sobre-general para un algoritmo como el XCS es que aquellos clasificadores que reciben diferentes valores de recompensa dependiendo del estado, tendrán una estimación del error posiblemente mucho mayor que en aquellos clasificadores que tienden a recibir

siempre valores muy parecidos de recompensa. La razón la podemos encontrar en la fórmula de actualización del error para el clasificador:

$$\varepsilon_{cl} = \varepsilon_{cl} + \beta * (|P - p_{cl}| - \varepsilon_{cl}) \quad (5.23)$$

Si la recompensa actual es muy diferente a la recompensa predicha, el error será elevado. Si el clasificador es aplicado en estados que producen recompensas diferentes, la predicción del error tenderá a oscilar entre estos valores diferentes y el error será acusado.

Por eso, en el XCS esta clase de clasificadores sobre-generales tienden a poseer una aptitud baja en comparación con los restantes clasificadores y no sobreviven al proceso de reproducción o borrado. En su artículo de comparativa, Tim Kovacs habla de una mayor insensibilidad del XCS a este efecto del sesgo de la función de recompensa.

Sin embargo, Kovacs añade que se mantiene otro efecto del sesgo de la función de recompensa, y es el hecho de que los clasificadores del XCS sólo pueden generalizar entre estados cuya recompensa se diferencia en una cantidad tolerable (suficiente, para que la estimación del error de predicción no sea demasiado elevada, y el clasificador pueda sobrevivir). En ese sentido, los sistemas tradicionales CS no tienen este problema, pudiendo expresarse en ellos generalizaciones que no podrían sobrevivir en un XCS. Esto a su vez es parte de la problemática ya descrita de los clasificadores sobre-generales fuertes y su incidencia en el CS.

Diferencias en las representaciones.

La utilización de una aptitud basada en la fuerza, o una aptitud basada en la predicción de la precisión tiene su incidencia en la forma en la que los clasificadores de la población representan los estados del entorno.

En los sistemas con aptitud basada en la fuerza, se tenderá a obtener un mayor número de clasificadores en aquellos estados con valores de recompensa mayores. Por el contrario, pueden aparecer estados que no queden representados por ningún clasificador como ya se ha mencionado anteriormente. Este comportamiento conduce a un mapa donde los estados con mayores recompensas vendrán representados por el

clasificador con la mejor de las acciones. Es lo que se denomina el mapa de las mejores acciones. En el caso de los sistemas con aptitud basada en la fuerza, éste es además un mapa parcial ya que podría haber estados que no fueran representados por ningún clasificador.

En contraste, la idea en la que se basa el XCS es encontrar una población de reglas tales que se consideren todas las acciones en todos los estados. Esto sería un mapa completo. En el XCS, cuando todas las acciones son consideradas, el mecanismo de selección de acciones es el único responsable de la acción que acabará ejecutándose, mientras que en el CS clásico la propia selección de los clasificadores juega un papel adicional.

5.4.2 Sistemas tipo Pittsburgh.

El problema de la aportación de crédito a las reglas individuales fue soslayado por la estrategia seguida por DeJong y sus estudiantes en la Universidad de Pittsburgh. [Smith 1980], [Smith, 1983].

La diferencia fundamental respecto a los sistemas tipo Michigan es la consideración de que el conjunto completo de reglas constituye un individuo de la población. Por ello, el algoritmo genético trabaja con una población de individuos candidatos a solución, sin necesidad de recurrir a un sistema de estimación del crédito para partes de la solución total. Este diferencia hace que los sistemas de tipo Pittsburgh se orienten a la competición entre individuos de la población, más que en la cooperación propia de los sistemas tipo Michigan. A efectos de seguir comparando ambas estrategias vamos a incluir un esquema bajo la misma perspectiva que en los sistemas tipo Michigan (figura 5.5).

La aptitud de cada individuo viene dada por la respuesta conseguida por esa solución en el entorno. Mientras en la aproximación Michigan un individuo es una regla aislada que tiene que cooperar con otras, tanto para formar el sistema como para adaptar su valor de recompensa, en la aproximación de Pittsburgh las soluciones completas compiten para actuar y reproducirse en la siguiente generación de soluciones.

Comparando los dos esquemas, apreciaremos en primer lugar que la realimentación obtenida del entorno por la acción de un sistema basado en reglas es evaluada directamente. El resultado servirá para asignar una aptitud al individuo de la población responsable en cada momento de las acciones del sistema. El sistema

descubridor usará directamente la población de individuos con sus aptitudes asociadas para seleccionar y producir nuevas bases de reglas.

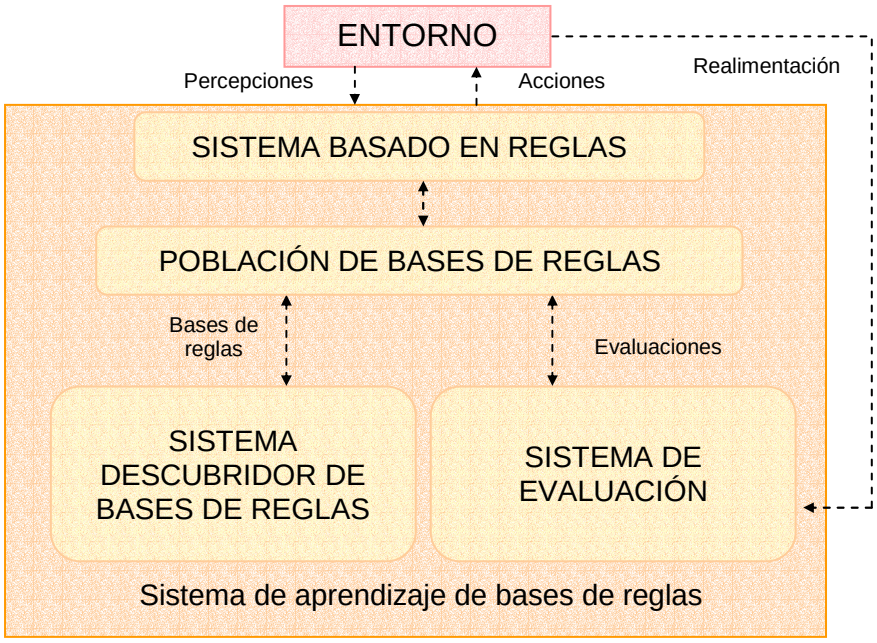


Figura 5.5. Esquema de Pittsburgh.

Un hecho muy importante es que la evaluación de la población de individuos requiere de la aplicación de cada base de reglas sobre el entorno para poder comprobar y medir su funcionamiento. Esto trae consigo algunas consecuencias importantes. Las bases de reglas son evaluadas de forma independiente, no cooperan sino que compiten. Por otra parte, es fácil comprender el coste computacional que supone en muchos casos la evaluación de poblaciones con un número relativamente elevado de individuos. Aparece aquí una dicotomía: la simplificación del la estructura del sistema en el caso de la estrategia Pittsburgh frente a la estrategia Michigan produce un incremento del esfuerzo computacional.

En el sistema de descubrimiento encontramos también una diferencia importante. En la estrategia Michigan el ritmo de reemplazo de individuos en la población debe ser suficientemente pequeño para que la cooperación entre individuos sea efectiva, ya que debe procurarse un mínimo de interacción entre las reglas. Dado el carácter meramente competitivo de la estrategia de Pittsburgh (no hay degradación del sistema mientras los mejores individuos estén presentes en la población), esta restricción no es tan fuerte en su caso. Aun así, hay que mantener el nivel de reemplazos

bajo cierto umbral para evitar la aparición de una convergencia demasiado temprana [Cordón et al., 2001].

Otro aspecto a considerar es el del ritmo relativo de evaluaciones y descubrimiento. En los algoritmos tipo Michigan, deben producirse un gran número de evaluaciones entre la aplicación del operador de descubrimiento y la evaluación, las suficientes para conseguir un situación estable en el proceso de asignación de recompensas. En cambio, en Pittsburgh, el esquema es diferente ya que tras la evaluación de todos los individuos es cuando se aplica inmediatamente el operador de descubrimiento.

5.5 Sistemas clasificadores con bases de reglas borrosas.

Los sistemas clasificadores se pueden considerar como algoritmos de búsqueda que pretenden obtener un conjunto adecuado de reglas para interaccionar con el entorno. En esta sección analizaremos los esfuerzos realizados encaminados a la utilización de reglas borrosas, en lugar de clasificadores como los descritos anteriormente. Este tipo de sistemas son denominados sistemas genéticos borrosos basados en reglas (en inglés *Genetic Fuzzy Rule-Based Systems*). Los denominaremos por sus siglas en inglés: GFRBS.

Desde ahora, debemos advertir que el sistema investigado en esta tesis presenta importantes diferencias respecto a los GFRBS que vamos a describir aquí. Sin embargo, parece conveniente describir las alternativas al método propuesto.

5.5.1 Algoritmos tipo Michigan.

La estructura de un GFRBS basado en una estrategia tipo Michigan es la que se muestra en la figura 5.6.

Como podemos apreciar en el esquema, se trata de una estructura muy similar al sistema clasificador convencional, solo que ahora estamos tratando con reglas borrosas y se utiliza un sistema de inferencia borroso para producir las acciones sobre el entorno en función de los datos obtenidos de los sensores.

En este caso, cada regla borrosa juega el papel de un clasificador. La población de clasificadores es la base de reglas. El sistema de inferencia borroso sustituye al sistema de producción.

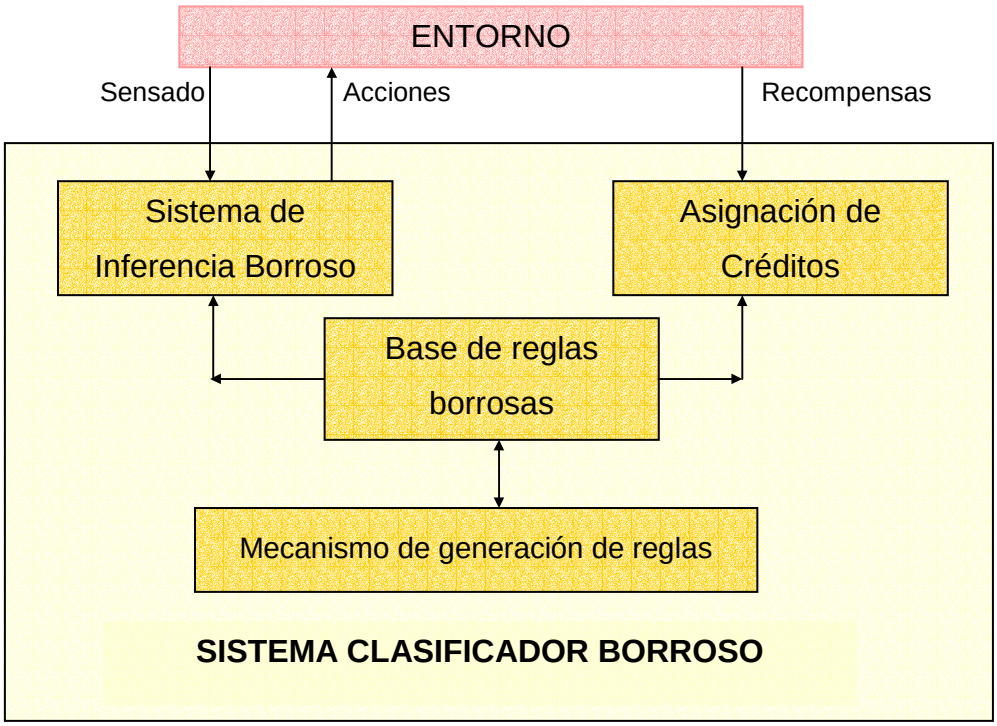


Figura 5.6. Esquema de un sistema clasificador borroso basado en una estructura tipo Michigan

5.5.1.1 Sistemas clasificadores borrosos para el aprendizaje de bases de reglas.

En general, este tipo de sistemas se ha utilizado considerando la evolución de la representación lingüística de la regla borrosa, o incluyendo en cada regla además la definición de las funciones de pertenencia utilizadas en esa regla. Como ejemplo, veamos el sistema de clasificación propuesto en [Valenzuela-Rendón, 1991], donde cada clasificador representa exclusivamente la estructura lingüística de la regla.

En este esquema se considera la definición previa de las variables lingüísticas (parámetros de las funciones de pertenencia asociadas a las valoraciones de cada variable). Entonces los cromosomas que integran la población representan reglas del tipo Si-Entonces, por ejemplo:

Si *X0* es {*Baja* o *Media*} y *X1* es {*Baja* o *Alta*} entonces *Y* es *Alta*

En el esquema propuesto por Valenzuela-Rendón la regla anterior se traduciría por:

00: 110, 01:101 / 11:001

Los primeros dos 0's identifican a la variable *X0*. Los tres bits a continuación indican cuales de las tres posibles valoraciones de la variables *X0* {*Bajo, Medio, Alto*} intervienen en la disyunción: en el ejemplo 110 significa {*Baja* o *Media*}. Después de la coma, tenemos la referencia a la variable *X1* (01) y el código del término {*Baja* o *Alta*}. Finalmente después del símbolo / tenemos el código de la variable de salida *Y* (11) y el código 001 referenciando a la valoración lingüística *Alto*.

El sistema de Valenzuela-Rendón utiliza además los denominados mensajes borrosos. Un mensaje borroso se codifica de forma similar a las reglas. Por ejemplo:

01:100

significaría, la variable *X1* toma el valor *Bajo*.

Los mensajes de entrada son los creados por la denominada unidad de entrada y hacen referencia al estado de las variables de entrada. Estas variables normalmente toman un valor real, que debe ser sustituido en la función de pertenencia de la valoración a la que se hace referencia en el mensaje borroso. El resultado de esta sustitución en la función de pertenencia devuelve el grado de activación del mensaje. Los mensajes con un grado de activación igual a 0 son eliminados de la lista de mensajes.

El encaje de los clasificadores que componen la población de reglas borrosas se realiza teniendo en cuenta el grado de activación de los mensajes. El procedimiento para realizar esta tarea incluye dos pasos.

Primero, se calcula el nivel de cumplimiento de cada una de las condiciones relativa a las variables de entrada del clasificador. Este nivel de cumplimiento se calcula como el máximo de los grados de activación de los mensajes borrosos que encajan con la variable de entrada considerada en la condición.

Después se determina el nivel de activación del clasificador, como el valor mínimo de todos los niveles de cumplimiento establecidos para las condiciones del clasificador.

Cada vez que un clasificador se dispara, su parte de acción es trasladada a la lista de mensajes. Este mensaje tendrá una activación proporcional al nivel de activación del clasificador que la generó. Si el mensaje ya se encuentra en la lista, no es duplicado sino que su nivel de activación es aumentado. Es importante recordar, que los clasificadores pueden producir mensajes que afecten a las variables de entrada u otras variables internas, lo que provoca la posible concatenación de activaciones de un conjunto de clasificadores. Por este motivo, es necesario un mecanismo de asignación de crédito como el "bucket brigade", donde se contabiliza la influencia de unos clasificadores en la activación de otros.

La unidad de salida del sistema se encarga de calcular los conjuntos borrosos que van a ser asignados a las variables de salida. Para cada variable de salida, busca los mensajes de la lista que hacen referencia a ella y los agrega utilizando inferencia max-min: cada conjunto borroso resultante es recortado utilizando el nivel de activación del mensaje y luego son agregados mediante un operador max. El conjunto borroso resultante es desborrosificado mediante el método del centro de gravedad.

El sistema de Valenzuela-Rendón utiliza el algoritmo de "bucket brigade" para la asignación de créditos. Cada clasificador tiene una fuerza asociada, que se usará después en la parte evolutiva del algoritmo para considerar el conjunto de clasificadores seleccionados. Inicialmente, todos los clasificadores tienen la misma fuerza asociada. Cada vez que se obtiene una salida y se aplica al entorno, éste devuelve una recompensa. La recompensa se distribuye entre todos aquellos clasificadores que contribuyeron a la generación de esa salida, de forma que aquellos clasificadores con una salida más similar a la que realmente se produce reciben una recompensa mayor. Aquellos clasificadores que no estuvieron directamente involucrados en la producción de la salida, pero que previamente enviaron mensajes que activaron los clasificadores que sí se implicaron en la salida, reciben como recompensa una fracción de la que éstos últimos obtuvieron. Además el sistema de Valenzuela-Rendón descuenta en cada clasificador un cantidad fija de fuerza a todos los clasificadores.

El mecanismo de descubrimiento de nuevos clasificadores, se implementa en el sistema de Valenzuela-Rendón, mediante un GA del tipo "estado-estacionario". Es

decir, sólo una fracción de la población, los clasificadores mas débiles, es sustituida. El clasificador más débil es reemplazado por un nuevo clasificador obtenido de dos padres seleccionados en función de su fuerza. El GA no se realiza en todas las iteraciones sino tras un cierto número de pasos.

Es importante resaltar que el esquema de codificación impuesto para los clasificadores hace que la aplicación de los operadores genéticos pueda producir individuos que son sintácticamente incorrectos. En ese caso, el individuo es descartado y se procede nuevamente a la generación de un nuevo individuo.

5.5.1.2 Sistemas clasificadores borrosos para el aprendizaje de bases de reglas borrosas.

La principal diferencia con el sistema descrito anteriormente, es que en este caso el clasificador debe codificar los parámetros de las funciones de pertenencia asociadas a los antecedentes de la regla, en lugar de la estructura lingüística de la misma.

Los primeros investigadores en desarrollar un esquema de este tipo, fueron Parodi y Bonelli [Parodi y Bonelli, 1993]. Presentan un sistema clasificador borroso que aprende una base de reglas borrosas del tipo Mandani. La lista de mensajes sólo incluye mensajes de entrada, ya que los clasificadores sólo se refieren en su parte de acción, a las variables de salida del sistema. De esta forma, lo que el sistema clasificador realiza realmente es aproximar una función que relaciona las entradas con las salidas, en lugar de un proceso de decisión que puede encadenar la activación de una serie de clasificadores. La dinámica de este sistema clasificador borroso es más simple con lo que se evita la necesidad de un mecanismo de asignación de crédito más sofisticado como el "bucket-brigade".

En el algoritmo de Parodi y Bonelli las funciones de pertenencia son triangulares simétricas, requiriendo dos parámetros: el centro y el ancho del triángulo. Así, para la entrada i en la regla k-ésima el centro es x_{cik} y el ancho es x_{wik}. Una regla queda representada como:

$$R_k : ((x_{c1k}, x_{w1k}), (x_{c2k}, x_{w2k}), ..., (x_{cik}, x_{wik}), ..., (x_{cnk}, x_{wnk})) \rightarrow (y_{ck}, y_{wk})$$

El descubrimiento de nuevas reglas (o lo que es lo mismo, nuevas combinaciones de parámetros), se realiza mediante un GA donde sólo se codifican los valores de los centros de los triángulos. Los valores de las desviaciones se mantienen constante para las diferentes reglas.

Otro ejemplo de sistema clasificador borroso para la generación de reglas borrosas es el propuesto por Velasco y Magdalena [Velasco y Magdalena, 1995]. Una de las características del sistema es que es adaptativo, capaz de cambiar sus base de reglas borrosas mientras interactúa controlando un sistema real.

Una particularidad que podemos encontrar aquí, es que las nuevas reglas generadas por el GA son almacenadas aparte antes de ser incluidas en la población de clasificadores "on-line". En este "limbo", las reglas son evaluadas "off-line" y sólo si se encuentra que la regla puede ser útil, es llevada al conjunto de reglas que actúa "on-line". El algoritmo asigna una fuerza a cada clasificador que tenga una activación diferente de 0, independientemente de si la regla está en el conjunto "on-line" o en el limbo. La evaluación de un clasificador se realiza en términos de la similitud de la acción que propone respecto a la de la acción que actualmente se está ejecutando sobre el sistema real.

Las reglas borrosas utilizan funciones de pertenencia trapezoidales (4 parámetros). Cada regla tiene un número variable de antecedentes y un solo consecuente. Además cada regla tiene asociado un conjunto de parámetros: fuerza, promedio de su fuerza a lo largo del proceso, tiempo de vida, etcétera.

La codificación de las reglas se realiza mediante cromosomas de longitud variable, quedando definidos como una lista de parejas: el primer valor es un entero que representa a la variable, y el segundo es un conjunto de parámetros que define a la función de pertenencia del conjunto borroso. La última pareja especificada en el cromosoma se refiere a la variable de salida.

Es interesante ver como se usa el limbo para producir el aprendizaje on-line de la base de reglas borrosas. Como se ha descrito, el GA produce nuevos clasificadores que son almacenados en un conjunto intermedio, para ser evaluados off-line. El algoritmo de evaluación adapta la fuerza de las reglas en el limbo, aunque estas no contribuyan a la decisión que se está produciendo. La fuerza de una regla candidata se estima basándose en una medida de similaridad entre su acción propuesta y la acción

que el sistema está llevando a cabo actualmente. Para pasar una regla del limbo al conjunto de reglas activas se tienen en cuenta los siguientes parámetros:

- Edad de la regla. Número de iteraciones desde que la regla fue puesta en el limbo.

- Activaciones de la regla. Número de iteraciones en los que la regla hubiera sido disparada si hubiera estado en el sistema on-line.

- Evaluación equivalente de la regla. Valor constante que se debería haber añadido a la medida de eficacia de la regla, para alcanzar su valor actual tras el número de activaciones de la regla.

- Edad del limbo. La edad del limbo se establece a priori. Representa el número de iteraciones tras el cual la regla debe ser pasada necesariamente al conjunto de reglas on-line o definitivamente descartada.

- Número mínimo de activaciones. Parámetro fijado a priori. Se trata del mínimo número de activaciones que debe tener una regla para que se considere que puede pasar del limbo al conjunto de reglas activas.

- Evaluación equivalente mínima. Valor mínimo de la evaluación equivalente de la regla que ésta debe poseer para poder pasar del limbo al conjunto de reglas activas.

El sistema tiene en cuenta estos parámetros para ordenar las reglas. Las reglas peores son eliminadas y el resto son filtradas conforme a ciertos criterios. Aquellas reglas que finalmente cumplan con los criterios son pasadas del limbo al conjunto de reglas activas. Los criterios establecen que aquellas reglas cuyo período de evaluación expiró y no han tenido la actividad suficiente u obtuvieron una mala evaluación cuando estuvieron activas son eliminadas del limbo. Por el contrario, las reglas que se han activado suficientes veces y obtuvieron una buena evaluación son pasadas al conjunto

de reglas on-line. El resto de las reglas permanece en el limbo, para ser evaluadas un mayor número de veces antes de tomar una decisión.

5.5.2 Algoritmos tipo Pittsburgh.

La estructura general de un GFRBS basado en un esquema de Pittsburgh es el que se muestra en la figura 5.7 [Cordón et al., 2001].

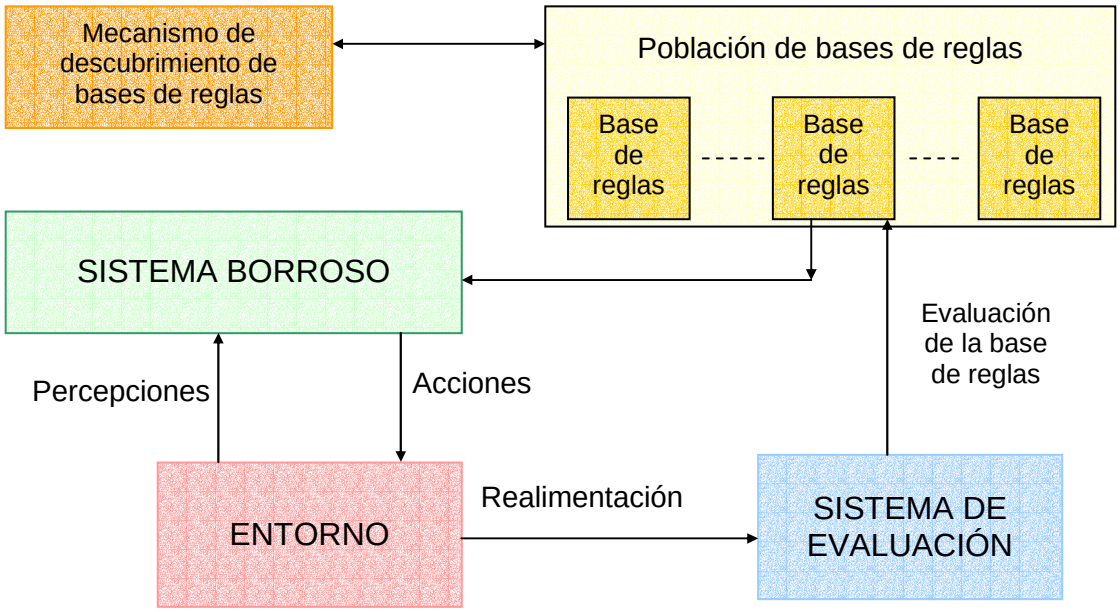


Figura 5.7. Representación esquemática de un GFRBS de tipo Pittsburgh.

Como se desprende de la figura, el mecanismo de aprendizaje de los GFRBS de tipo Pittsburgh actúa sobre una población de conjuntos de reglas borrosas. La evaluación de los individuos de la población requiere la utilización de un sistema borroso que interactúa con el entorno y se configura con la base de reglas borrosas que se está evaluando. El resultado de las acciones sobre el entorno es analizado por un sistema de evaluación, que determina la aptitud de la base de reglas correspondiente.

Igual que en el caso de los sistemas tipo Michigan, debemos distinguir los casos en los que el cromosoma se limita a representar la información lingüística de la base de reglas (quizás con alguna información añadida sobre las funciones de pertenencia o el escalado de las variables de entrada y salida) y aquellos en los que el cromosoma almacena información tanto de la base de reglas como de las funciones de pertenencia. Es evidente que en el caso de los sistemas de tipo Pittsburgh, el aspecto de la

representación es más complejo y juega un papel mucho más importante que en los sistemas tipo Michigan.

El aprendizaje en los sistemas tipo Pittsburgh garantiza la cooperación de las reglas que forman parte del sistema ya que el mecanismo evolutivo obliga al sistema de reglas a configurarse de manera conjunta para obtener un valor óptimo. Sin embargo, el precio a pagar es el aumento de la dimensionalidad del espacio de búsqueda, con lo que se vuelve más difícil encontrar soluciones adecuadas.

5.5.2.1. Codificación de los sistemas borrosos.

Vamos a centrar la discusión de esta clase de sistemas en los aspectos relativos a la representación de las bases de reglas borrosas. Comenzaremos con la codificación del contenido lingüístico de la base de reglas.

A la hora de codificar la estructura de una base de reglas, se tienen dos alternativas a considerar: cromosomas de longitud fija y cromosomas de longitud variable. La primera de las alternativas se debe tomar con sistemas de reglas con una estructura prefijada. En estos casos, la codificación podrá ser posicional, si el papel de cada gen en el cromosoma viene dada por su posición dentro del mismo, o no posicional, si esto no sucede de esta forma. En el caso de la codificación de una lista de reglas, tendremos un código no posicional, ya que el orden de las reglas puede alterarse sin modificar el resultado final de la inferencia.

Cuando la lista de reglas no tenga un número fijo de reglas habrá que recurrir a una codificación de longitud variable. A su vez, cada regla dentro de la lista puede ser codificada con un código de longitud fija o variable, dependiendo de si el número de antecedentes y consecuentes se mantiene constante o se hace variable.

En el caso de reglas donde se codifica la valoración lingüística de las diferentes variables que componen los antecedentes y consecuentes, podrá usarse un código posicional, donde la posición del gen dentro de la parte del cromosoma relativa a la regla establece la variable que se está representando.

Otra alternativa muy usada, si se codifican los parámetros de las funciones de pertenencia, es que una parte del código indicará la variable a la que asignar la función de pertenencia y, a continuación, otro código representará los parámetros de la misma.

Este código será no posicional, ya que no es la posición dentro del cromosoma lo que indica la función del gen, sino otro código que le precede.

Vamos a comenzar describiendo la forma usual en la que se utilizan los códigos posicionales. Éstos se aplican, sobre todo, en sistemas borrosos basados en tablas de decisión.

Las tablas de decisión borrosas definen una relación concisa entre una colección de conjuntos borrosos correspondientes a las entradas del sistema y conjuntos borrosos correspondientes a las salidas. Una tabla de decisión puede ser utilizada para representar un conjunto de reglas Si-Entonces, con variables de entrada y variables de salida. Por ejemplo:

Si *X1* es *pequeño* y *X2* es *corto* entonces *Y* es *malo*.
Si *X1* es *pequeño* y *X2* es *mediano* entonces *Y* es *malo*.
Si *X1* es *mediano* y *X2* es *corto* entonces *Y* es *regular*.
Si *X1* es *grande* y *X2* es *mediano* entonces *Y* es *regular*.
Si *X1* es *grande* y *X2* es *largo* entonces *Y* es *bueno*.

Podemos construir una tabla de decisión en la forma mostrada en la tabla 5.5.

X2, X1	Pequeño	Medio	Grande
Corto	*Malo*	*Medio*	
Medio	*Malo*		*Medio*
Largo			*Bueno*

Tabla 5.5. Tabla de decisión.

El cromosoma correspondiente a una tabla de decisión puede obtenerse sin más que recorrer la tabla por filas y codificar cada salida como un entero. Se puede utilizar un etiqueta “nula” para representar la ausencia de salida. En el ejemplo que se ha presentado, el cromosoma correspondiente sería (1,2,0,1,0,2,0,0,3).

El operador genético de cruce, en sus diversas modalidades, opera en estas cadenas de la forma normal, aunque los códigos no son binarios, sino elementos de un conjunto finito. En la tabla 5.6 tenemos un ejemplo de la actuación del operador de cruce con dos puntos.

Progenitor A	1	2	0	1	0	2	0	0	3
Progenitor B	0	0	3	2	1	2	0	2	1
Descendiente A	1	0	3	2	0	2	0	0	3
Descendiente B	0	2	0	1	1	2	0	2	1

Tabla 5.6. La líneas más gruesas indican los dos puntos de referencia para el cruce.

El operador de mutación simplemente cambia el entero que representa a la valoración lingüística por otro adecuado a la representación.

Las tablas de decisión pueden extenderse para representar sistemas borrosos de tipo TSK. En este caso, la principal diferencia estará en el contenido de las celdas de la tabla. En lugar de ser etiquetas que referencian a valoraciones lingüísticas de la variable de salida, tendremos aquí los parámetros de una función lineal.

Supongamos que el sistema TSK tiene n variables de entrada $X_1, X_2, ..., X_n$ con $N = \{N_1, N_2, ..., N_n\}$ valoraciones lingüísticas en cada variable y una sola variable de salida Y, que será un número. Podemos representar la base de reglas como una tabla de decisión con $L_r = \prod_{i=1}^{n} N_i$ celdas. Cada celda va a contener $n+1$ parámetros, representando los coeficientes de la ecuación lineal $Y = w_0 + \sum_{i=1}^{n} w_i X_i$, que constituye la salida del sistema. Por consiguiente la base de reglas del sistema TSK puede ser descrita como una lista de $L_r(n+1)$ números reales. Cada número de la lista puede entonces codificarse como una cadena de bits de longitud fija. Los operadores genéticos usuales pueden entonces aplicarse sobre la cadena en la forma convencional.

La principal desventaja de los esquemas de codificación mencionados es que si el número de entradas y salidas crece, o el número de valoraciones lingüísticas es elevado, el tamaño de la tabla de decisión se incrementa rápidamente. Esto lleva a la idea de trabajar con una lista o conjunto de reglas donde no se representan todas las posibles combinaciones de posibles entradas. En este caso, es posible que parte del espacio de entradas no sea "cubierto" por reglas incluidas en la lista.

Las reglas borrosas elementales son las que vienen representadas por las celdas de una tabla de decisión. En el caso de trabajar con una lista de reglas, podemos usar una variante denominada regla DNF (*disyuntive normal form* o forma normal disyuntiva). En esta variante, pueden aglutinarse varias reglas elementales en una sola regla. Por ejemplo:

Si *X* es {*Co* o *Cp*} entonces *Y* es {*Dq* o *Dr*}

En este caso se representa mediante el operador "o" lo equivalente a varias reglas elementales agregadas. En el ejemplo, si el sistema considerado solo tiene una entrada *X* y una salida *Y*, se estarían reemplazando 2 reglas elementales:

Si X es Co Entonces Y es Dq o Dr
Si Y es Cp Entonces Y es Dq o Dr

Sea el sistema con 3 variables de entrada *X1, X2* y *X3* y la salida *Y*. La siguiente regla DNF:

Si *X1* es {*C13* o *C14*} y *X3* es {*C31* o *C32*} entonces *Y* es {*D14* o *D15*}

reemplaza a 12 reglas borrosas elementales (debe calcularse como el producto de las reglas sustituidas por cada variable en la parte de los antecedentes (2 x 3 x 2)) .

La utilización de reglas DNF alivia el problema del crecimiento excesivo de las tablas de decisión cuando el número de entradas y salidas es elevado.

La codificación de la lista de reglas, se puede realizar de muchos modos, pero normalmente se realiza la concatenación del código de cada regla. Cuando no se use un código de longitud fija para codificar cada regla individual es preciso conectar las reglas mediante un código de concatenación, que indica el final de una regla y el comienzo de la siguiente.

Un ejemplo de código de longitud fija es el propuesto en [González et al., 1993], [Magdalena y Monasterio, 1995]. Para cada variable de entrada Xi, el código contiene una subcadena de Ni bits. Cada bit representa la inclusión o ausencia de una de las valoraciones lingüísticas asociadas a la variable Xi en una disyunción. Así la regla DNF:

Si *X1* es {*C12* o *C13*} y *X2* es *C23* y *X3* es {*C34* o *C35*} entonces *Y1* es {*D16* o *D17*}

con {*C11, C12, C13, C14,C15*} el conjunto de valoraciones lingüísticas de *X1*, {*C21, C22, C23*} el conjunto de valoraciones lingüísticas de *X1*, {*C31, C32, C33, C34, C35*} el conjunto de valoraciones lingüísticas de *X3* y {*D11, D12, D13, D14, D15, D16*, *D17*} el conjunto de valoraciones lingüísticas de *Y1*, se codificaría:

{0 1 1 0 0} {0 0 1} {0 0 0 1 1} {0 0 0 0 0 1 1}

Como vemos, se trata de un código posicional de longitud fija, donde un 0 indica la ausencia de la valoración correspondiente en la disyunción, mientras que un 1 representa la inclusión.

La base de reglas se codifica como la concatenación de los códigos de las reglas que intervienen: $(r_1 \mid r_2 \mid ... \mid r_k)$. El número de reglas a incluir es normalmente variable, con el límite superior dado por el mayor número posible de reglas elementales (tamaño de la tabla de decisión).

Veamos ahora como la definición de los operadores genéticos para bases de reglas con códigos de longitud fija. Distinguiremos entre los operadores clásicos y los operadores no posicionales.

El operador de cruce clásico aplicado sobre una lista de reglas se aplica al nivel de las reglas. Dados dos cromosomas representando dos bases de reglas:

$$r_i = \{r_{i1}, r_{i2}, ..., r_{ik}\}$$
$$r_j = \{r_{j1}, r_{j2}, ..., r_{jl}\}$$

se definen dos puntos de cruce al azar, uno para cada base de reglas (obsérvese que cada base de reglas puede tener un número diferente de reglas).

La operación de cruce nos lleva de:

$$r_i = \{r_{i1}, r_{i2}, ..., r_{i\alpha} \mid r_{i\alpha+1}, ..., r_{ik}\}$$
$$r_j = \{r_{j1}, r_{j2}, ..., r_{j\beta} \mid r_{j\beta+1}, ..., r_{jl}\}$$

a

$$r_i = \{r_{i1}, r_{i2}, ..., r_{i\alpha} \mid r_{j\beta+1}, ..., r_{jl}\}$$
$$r_j = \{r_{j1}, r_{j2}, ..., r_{j\beta} \mid r_{j\alpha+1}, ..., r_{jk}\}$$

Este tipo de operación de cruce tiene una desventaja importante, ya que en las listas de reglas la posición del gen no está relacionado con su función. Por este motivo, la recombinación posicional se puede considerar una alteración puramente aleatoria.

El operador de mutación clásico actúa en las listas de reglas a nivel de los bits o valores enteros que componen la regla .

Además de los operadores de cruce y mutación, se suelen incluir otros operadores, como el de reordenación y el de alineación. El operador de reordenación modifica el ordenamiento de las reglas en la lista. En realidad, ya se ha comentado que la posición de la regla en la lista es irrelevante. Sin embargo, cambiar el orden de las reglas sí puede tener un efecto para los resultados obtenidos en la aplicación de operaciones de cruce, formándose nuevos subconjuntos de reglas o rompiendo algunos ya existentes. Una forma típica de un operador de reordenamiento es la rotación circular de la lista de reglas, llevando la regla que ocupa la posición $\gamma+1$ hasta la primera posición de la lista.

Aunque el operador de reordenamiento aumenta la diversidad de los descendientes que se pueden obtener en operaciones de cruce, subsiste el problema de que dentro del cromosoma no hay una ubicación en la que genes contiguos tengan características similares. Para ello, en algunas aplicaciones se utilizan operadores de alineación, que toman dos cromosomas progenitores y producen dos descendientes donde aquellos genes (reglas) con similaridades en la parte de los antecedentes ocupan posiciones similares en los cromosomas descendientes.

Pasemos ahora a examinar los denominados operadores de cruce no posicionales, que pretenden resolver en la propia definición del operador de cruce, los problemas atacados anteriormente mediante la operación de reordenamiento y la operación de alineamiento.

El operador de cruce no-posicional propuesto en [Magdalena, 1998] se basa en reordenar las reglas conforme al orden que asumirían en la tabla de decisión correspondiente. Sea por ejemplo, el conjunto de reglas:

R1*:* Si *X1* es {*C11* o *C12* o *C13*} y *X2* es {*C21* o *C22* o *C23*} entonces *Y* es *D5.*
R2*:* Si *X1* es {*C11* o *C12* o *C13*} y *X2* es {*C23* o *C24* o *C25*} entonces *Y* es *D4.*
R3*:* Si *X1* es {*C13* o *C14* o *C15*} y *X2* es {*C21* o *C22* o *C23*} entonces *Y* es *D2.*
R4*:* Si *X1* es {*C13* o *C14* o *C15*} y *X2* es {*C23* o *C24* o *C25*} entonces *Y* es *D1.*

En la tabla de decisión correspondiente (tabla 5.7) se ha marcado en cada celda qué regla la está representando:

X1/X2	C11	C12	C13	C14	C15
C21	R1	R1	R1	R3	R3
C22	R1	R1	R1	R3	R3
C23	R1	R1	R4	R4	R4
C24	R2	R2	R4	R4	R4
C25	R2	R2	R4	R4	R4

Tabla 5.7. Tabla de decisión representando las celdas que son asumidas por R1, R2 o R3.

El operador propuesto por Magdalena se basa en realizar la operación de cruce sobre la tabla de decisión usando una "superficie de corte". A continuación, vemos la actuación de esta operación de cruce sobre dos tablas de decisión progenitoras (figura 5.8).

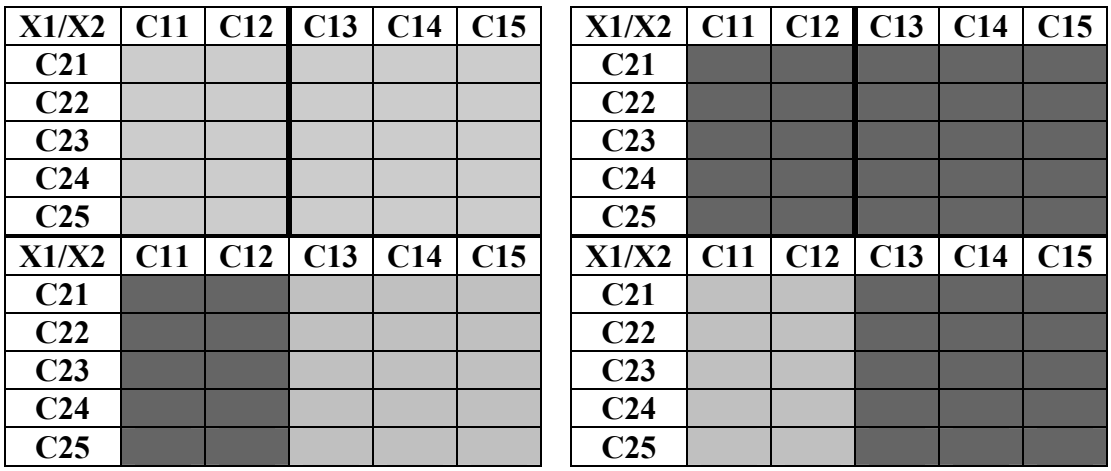


Figura 5.8. Operación de cruce mediante una superficie de corte aplicada a una tabla de decisión.

Como vemos en al figura 5.8, la superficie de corte divide las tablas de decisión de los progenitores en dos zonas que son intercambiadas. Esta idea se encuentra con el problema de que determinadas reglas pueden quedar atravesadas por la superficie de corte.

En ese caso, hay varias opciones. La primera y más evidente es la de descomponer la regla en otras, con la desventaja de la fragmentación creciente con tendencia a la producción de reglas elementales.

En realidad, una superficie de corte divide el código en dos fragmentos. El operador de cruce se encarga de intercambiar estos fragmentos en los individuos

descendientes. La decisión sobre a qué fragmento del código pertenece una regla cuando es atravesada por la superficie de corte y una parte de la misma queda a un lado y a otro de la división se puede realizar de diferentes maneras.

Una de las técnicas es estocástica. Se establecen probabilidades para cada fragmento en función de la porción de volumen de cada fragmento cubierto por la regla dividida. Tras un experimento aleatorio con las probabilidades mencionadas se determina del resultado del mismo, a qué fragmento debe pertenecer la regla.

La técnica puede ser determinista, tomando como por ejemplo el criterio de que la regla será asignada a aquel fragmento con un mayor volumen cubierto por la regla.

Hasta ahora se han introducido técnicas para la representación de bases de reglas, donde cada una tiene una longitud fija. Sin embargo, para determinadas aplicaciones puede resultar conveniente la utilización de reglas especificadas con códigos de longitud variable. Dado que en este trabajo no se han usado reglas de este tipo, no vamos a incidir en este aspecto. Una revisión adecuada de este tema puede ser encontrada en [Cordón et al., 2001].

Sí vamos a incidir un poco más en la codificación de reglas de tipo aproximado o reglas donde los antecedentes y consecuentes no vienen dadas por la etiqueta correspondiente a la valoración lingüística sino por los parámetros de la función de pertenencia de un conjunto borroso. Es decir, las cláusulas de la forma "xi es Cj" se refieren a un conjunto borroso Cj cuyos parámetros son parte íntegra del cromosoma.

Debemos mencionar aquí, por su proximidad con el método utilizado en esta tesis, la representación propuesta en [Kang et al., 2000], que opera con sistemas del tipo TSK. Cada regla de longitud fija, utiliza cuatro parámetros por antecedente para representar una función de pertenencia trapezoidal y un parámetro para representar la constante utilizada en el consecuente. La lista de reglas es codificada, concatenando los códigos de las reglas individuales. Además el cromosoma contiene una matriz bidimensional, cuyas filas se corresponden con las reglas borrosas y sus columnas a las variables de entrada. Cada elemento $M_{ij} \in [0,1]$ de la denominada matriz de conexión M especifica la importancia relativa de la variables j en la regla i. De esta manera, un valor de 0 en la matriz significa que el valor de la variable de entrada correspondiente en la regla debe ser ignorado a la hora de calcular la salida del sistema.

Por último hay que mencionar las técnicas multi-cromosómicas utilizadas para la codificación de los sistemas borrosos basados en reglas del tipo lingüístico, es decir,

aquellos donde la denominada base de conocimiento se divide en la base de reglas y la base de datos. Esta división, que tiende a hacer más comprensible en términos lingüísticos el sistema resultante, tiene la desventaja de introducir el problema de manejar información no homogénea, ya que el contenido de la base de datos (definición de las funciones de pertenencia asociadas a las valoraciones lingüísticas de las variables) es muy diferente al de la base de reglas (conjunto de reglas del tipo Sí – Entonces). Una descripción más amplia de las técnicas más frecuentemente usadas la podemos encontrar en [Cordón et al., 2001].

Capítulo 6

Las máquinas finitas de estados borrosas como parte de un sistema clasificador.

6.1 Introducción.

El objetivo de esta investigación es estudiar el diseño de un sistema clasificador basado en una máquina finita de estados borrosa. La tarea de este clasificador es discriminar series de datos producto de procesos estocásticos con características diferentes. Entenderemos una serie de datos como la realización de un proceso estocástico, es decir un experimento aleatorio que produce un conjunto de valores numéricos ordenados tomando como base cierta variable independiente (el tiempo, distancias espaciales, etcétera) donde se establece una relación de orden.

Una característica de estos vectores de datos, es que las muestras no tienen por qué ser independientes, sino que normalmente se ven influenciadas por los valores de las muestras previas en el ordenamiento. Muchas veces, son estas relaciones las que

permiten diferenciar las series de datos provenientes de procesos estocásticos de características diferentes, más que los valores en sí mismos. Por consiguiente, es natural que se busquen clasificadores con funciones discriminantes que incidan en esta característica. Un ejemplo típico son los modelos de Markov, que serán descritos en el próximo capítulo, y donde la probabilidad de la ocurrencia del valor de un dato depende solamente del valor obtenido en el dato previo.

Los modelos recurrentes, como la máquina de estados borrosa descrita en este trabajo, incorporan una memoria a corto plazo que puede ser útil en la tarea de clasificación de estas series de datos, ya que las relaciones entre datos sucesivos pueden ser expresadas. En este capítulo, se describirá el procedimiento por el cual la máquina de estados borrosa puede incorporarse a la función discriminante de un clasificador, así como las técnicas utilizadas para el diseño del clasificador en dos arquitecturas diferentes: sistemas tipo Pittsburgh y sistemas tipo Michigan.

6.2 Clasificación de series temporales.

Para poder entender el concepto de proceso estocástico es necesario primero definir lo que es un *espacio de probabilidad*.

El concepto abstracto de espacio de probabilidad se refiere a una terna (Ω, F, P). En esta terna, Ω es el espacio de las muestras o conjunto de sucesos posibles, F es el conjunto de las partes de Ω y P es una función que asigna a cada elemento de F un número real cumpliendo los siguientes axiomas de probabilidad.

i) $P(A) \geq 0 \ \forall A \in F$ (6.1)

ii) $P(A_1 \cup A_2 \cup ... \cup A_n) = P(A_1) + P(A_2) + ... + P(A_n)$, si $A_i \cap A_j = \varnothing$ (6.2)

iii) $P(\Omega) = 1$ (6.3)

Un proceso estocástico es una familia de variables aleatorias $\{X(t,\omega), t \in T, \omega \in \Omega\}$, definidas sobre el espacio de probabilidad (Ω, F, P), y una relación de orden definida sobre T, de forma que si $t_1, t_2 \in T, t_1 \leq t_2$ o $t_1 > t_2$.

Con esta definición, si fijamos el valor de ω, $X(.,\omega)$ se convierte en una función de T en $\Re$ que denominaremos *serie de datos*. Cuando T es un conjunto discreto de puntos, por ejemplo $T = Z$ (conjunto de los números enteros), la serie se denomina serie temporal.

Es lógico pensar que un aspecto esencial en el análisis de una serie de datos es la posible relación entre los datos en función de la posición relativa de los mismos dentro de la serie. Por ello, en la clasificación de series temporales, no sólo deben tenerse en cuenta los valores aislados de las muestras, sino que también deben considerarse las posibles influencias entre los mismos. La aparición de este tipo de relaciones es muy común en la series temporales obtenidas de la Naturaleza, y son un reflejo de las propiedades de los sistemas con memoria.

La nomenclatura utilizada, es decir, el término *serie temporal*, puede llevar a engaño, ya que el tiempo no es la única variable independiente que se puede utilizar para ordenar los componentes de un vector de datos. Otras magnitudes, como por ejemplo, medidas de distancias espaciales, se pueden usar para producir series de datos, incluso variables más abstractas pueden intervenir en el ordenamiento.

En el terreno propio de la clasificación, la consideración del vector de características como una serie de datos tiene como objetivo la construcción de funciones discriminantes adecuadas, donde las relaciones entre datos sucesivos pueden explotarse para mejorar la clasificación.

Normalmente, la clave de este proceso consiste en considerar la dependencia existente entre datos cuya distancia relativa medida sobre la serie ordenada pertenece a un conjunto fijo de enteros $Z = \{z_1, z_2, ..., z_P\}$. De esta manera, las funciones discriminantes incluyen los siguientes bloques funcionales escritos de forma general:

$$d_i(\mathbf{x}) = g(f_i^1(x_1, x_{1-z(1)}, x_{1-z(2)}, ..., x_{1-z(P)}), ..., f_i^j(x_j, x_{j-z(1)}, x_{j-z(2)}, ..., x_{j-z(P)}), .., h_i(\mathbf{x})) \quad (6.4)$$

donde $f_i^j(x_j, x_{j-z(1)}, x_{j-z(2)}, ..., x_{j-z(P)})$ representa la interrelación entre los datos ordenados según alguna variable independiente y $h_i(\mathbf{x})$ representa otras contribuciones a esta función discriminante.

La construcción de esta función discriminante suele realizarse partiendo del ajuste de un *modelo* para la serie temporal. Entonces, la función discriminante se puede

obtener a partir de K modelos ajustados que representan las K clases de la clasificación. La medida de la bondad de la adecuación de cada modelo a los datos suele ser utilizada para la definición de la función discriminante. También suelen ser utilizadas estimaciones estadísticas, fundamentalmente la estimación de la probabilidad a posteriori $P(M_i \mid O)$, donde M_i es uno de los modelos utilizados para definir las clases y O son las observaciones de los datos.

Un ejemplo de esta técnica ya se comentó anteriormente en esta tesis, al describir las aplicaciones de los modelos ocultos de Markov.

Un sistema de inferencia borroso recurrente, como la máquina discreta de estados borrosa ya descrita, se puede usar para expresar relaciones del tipo $f_i^j(x_j, x_{j-z(1)}, x_{j-z(2)}, ..., x_{j-z(P)})$. La naturaleza recurrente de la máquina introduce una memoria a corto plazo en el sistema al considerar el concepto de estados y activaciones de los estados. Esto hace de este modelo un buen candidato para capturar información relevante concerniente a la dependencia entre datos sucesivos ordenados por una variable independiente. Por lo tanto, nuestro propósito es utilizar la máquina finita de estados borrosa como parte de una función discriminante usada para clasificar series temporales. Al tratar de clasificar en dos clases, se considera la siguiente función discriminante:

$$d(\mathbf{x}) = d_n(\omega(\mathbf{x})) - d_a(\omega(\mathbf{x})) \tag{6.5}$$

En esta fórmula $\mathbf{x}$ es el vector de características (serie temporal) y $\omega(\mathbf{x})$ es el resultado de procesar el vector de características, utilizando por ejemplo el algoritmo basado en la máquina de estados borrosa y que describiremos después. De esta manera, tenemos un espacio de características y un espacio de características procesado. En el espacio de características procesado se definen dos puntos: n y a, que representan respectivamente cada una de las clases en las que se desea realizar la clasificación. De esta manera, $d_n(\omega(\mathbf{x}))$ es una medida de distancia entre el vector de características procesado $\omega(\mathbf{x})$ y el punto n. De forma análoga se define $d_a(\omega(\mathbf{x}))$. Con estas definiciones, si $d(\mathbf{x}) \geq 0$ consideraremos que x pertenece a la primera clase y si $d(\mathbf{x}) < 0$, se considera que x pertenece a la segunda clase.

El espacio procesado de características es una transformación del espacio de características al que pertenece *x*. La transformación se puede ver como la concatenación de dos transformaciones más sencillas. La primera toma como entrada el espacio de características original compuesto por vectores de características de dimensión *N*. La máquina finita de estados borrosa con N_s estados realiza la transformación. Para cada vector de características, se obtiene una matriz $\mathbf{M}_{N_s \times N}(\mathbf{x})$ (N_s filas y N columnas). El elemento m_{rs} es el elemento de $\mathbf{M}_{N_s \times N}(\mathbf{x})$ de la fila *r* y la columna *s*, y representa la activación del estado *r* cuando la componente *s* del vector de características ha pasado por la máquina finita de estados borrosa. Por lo tanto, la primera transformación $T_1 : \mathbf{X} \rightarrow \Omega_1$ va desde el espacio de características de entrada X al espacio Ω_1, compuesto por las matrices con N_s filas y N columnas que se pueden obtener al procesar los vectores de características con la máquina seleccionada.

La segunda transformación es una medida de la matriz $\mathbf{M}_{N_s \times N}(\mathbf{x})$. En el presente trabajo, el parámetro seleccionado intenta medir la reactividad de la máquina. La idea de esta aproximación es evitar máquinas que ofrezcan **baja reactividad** ante el vector de características de entrada, asegurando que la información capturada por la máquina está relacionada con la dependencia entre muestras sucesivas. Por lo tanto, se elige el estado N como *estado de detección* y se establece un umbral $th \in [0,1]$. La reactividad se mide como el número de veces que la activación del estado de detección (última fila de la matriz $\mathbf{M}_{N_s \times N}(\mathbf{x})$) sobrepasa el umbral dividido entre *N*. Esta cantidad normalizada en el intervalo $[0, 1]$ tiene principalmente dos ventajas.

Primero, si la máquina finita de estados borrosa se comporta bien como clasificador, reaccionará, en el sentido de que ocurrirán activaciones /desactivaciones en, al menos, un estado y este comportamiento diferenciará cada una de las clases.

Segundo, este proceso es de cómputo fácil y rápido. Esta segunda característica es importante para el método de diseño propuesto en este trabajo, que está basado en computación evolutiva y puede requerir un número elevado de evaluaciones del algoritmo.

Por lo tanto, la segunda transformación es $T_2 : \Omega_1 \rightarrow \Omega_2$, donde Ω_2 es el espacio del *parámetro de reactividad* calculado a partir de la matriz $\mathbf{M}_{N_s \times N}(\mathbf{x})$.

La composición de ambas transformaciones genera la parte básica de la función discriminante: $\omega(\mathbf{x}) = T_2(T_1(\mathbf{x}))$. Ésta es una transformación desde el espacio del vector de características hasta el espacio del parámetro de reactividad: $\omega : \mathbf{X} \to \Omega_2$. En las secciones siguientes se describirá el proceso de diseño del clasificador basado en búsqueda evolutiva. Este método incluye la obtención de los puntos $\mathbf{n}$ y $\mathbf{a}$, y una discusión sobre la elección del último estado como estado de detección así como del valor particular del umbral th.

6.3 Proceso de diseño del clasificador. Sistemas tipo Pittsburgh y tipo Michigan.

6.3.1 Introducción.

La búsqueda de la máquina finita de estados borrosa, que hará las veces de función discriminante del clasificador para clasificar series de datos, se realiza mediante algoritmos genéticos. Se han desarrollado búsquedas con dos sistemas evolutivos basados respectivamente en arquitecturas tipo Pittsburgh y tipo Michigan.

Ambas estrategias evolutivas son procesos de aprendizaje supervisados, por lo tanto, es necesario extraer previamente las series de datos que constituirán el conjunto de entrenamiento para las máquinas. El objetivo es clasificar estas series de datos o "trazas" en dos conjuntos o clases diferenciadas. Es necesario que las trazas del conjunto de entrenamiento estén previamente catalogadas según su pertenencia a una clase u otra.

En este trabajo se han tratado de clasificar distintos tipos de series de datos. En primer lugar, se clasifican series de datos simulados (series de datos correspondientes a modelos ocultos de Markov) para estudiar la calidad del aprendizaje de los algoritmos utilizados y comprobar el efecto de los parámetros en el algoritmo. En segundo lugar, se clasifican series de datos reales (serie de datos que pretenden describir la distribución de cromatina en núcleos celulares a partir de imágenes médicas obtenidas con citologías) como un ejemplo de aplicación de estos sistemas borrosos de clasificación encontrados mediante los algoritmos que se describen en las siguientes secciones.

Como ya se ha comentado, el conjunto de entrenamiento está constituido por un conjunto de estas trazas previamente catalogadas. En el caso de la clasificación de las series procedentes de modelos simulados (modelos ocultos de Markov), podemos catalogar las trazas por haber sido generadas artificialmente, por lo que se sabe a qué modelo pertenece realmente cada una. En el caso de la clasificación de núcleos de células cancerígenas, las trazas corresponden a células catalogadas previamente por el especialista.

La extracción de las trazas se realiza de modo diferente según el problema planteado. Estos procesos de extracción serán comentados en los capítulos dedicados a pruebas y resultados, en los que se describe cada problema en detalle. Una vez analizado cada problema, se construyen las trazas que representan la información extraída del dominio y se aplica un algoritmo genético para tratar de encontrar la máquina de estados borrosa capaz de clasificar y reconocer los distintos patrones.

6.3.2 Sistemas tipo Pittsburgh.

6.3.2.1 Introducción.

En la implementación tipo Pittsburgh, el algoritmo trabaja con una población de individuos, donde cada uno es una posible solución completa. Recordemos que se trata de un proceso puramente competitivo donde se pretende llegar a una solución globalmente óptima. Los individuos de la población son máquinas de estados borrosas. A grandes rasgos, el proceso de búsqueda se desarrolla como sigue:

Inicialmente, se crea una primera generación de la población de individuos aleatoriamente. En cada iteración, se evalúa cada máquina de estados borrosa con las trazas del conjunto de entrenamiento y se asigna un valor de aptitud a cada máquina, que indica la bondad de la clasificación que presenta la máquina sobre estas trazas. Se selecciona una fracción de las mejores máquinas para iniciar un proceso de repoblación, y se sustituyen los individuos no seleccionados por los nuevos individuos generados en este proceso de renovación de la población. La repoblación se lleva a cabo por medio de la aplicación de tres operadores genéticos (reproducción, mutación y cruce) sobre los individuos seleccionados.

Este proceso se repite hasta encontrar una máquina cuyo valor de aptitud esté por debajo de un cierto umbral especificado por el usuario. En ese momento, el algoritmo detiene su ejecución.

Una de las cuestiones claves en la utilización de un esquema tipo Pittsburgh es la representación de cada individuo. Cada cromosoma de la población contiene una máquina codificada (un conjunto de reglas: antecedentes y consecuentes) en forma matricial. La parte de los antecedentes y la parte de los consecuentes de las reglas se codifican en dos matrices distintas. En este trabajo, se han asignado para todos los antecedentes funciones de pertenencia gaussianas. Los antecedentes y consecuentes de las reglas de cada máquina se codifican como se indica a continuación.

El número de reglas (parámetro *num_reglas*) y el número de estados (parámetro *num_estados*) deben fijarse a priori. La matriz de antecedentes es de dimensión *num_reglas* × (*num_estados* + 1): cada fila representa una regla, la primera columna representa el antecedente de la regla para la entrada externa y las siguientes columnas representan los antecedentes de las reglas para cada estado. La matriz de consecuentes es de dimensión *num_reglas* * *num_estados*, donde las filas representan las reglas y las columnas representan los consecuentes de las reglas para cada estado de la máquina. Ambas matrices se codifican de modo distinto.

En esta investigación hemos simplificado el espacio de búsqueda estableciendo a priori uno de los parámetros de las funciones de pertenencia de los antecedentes: la desviación de las gaussianas, de forma similar a [Cordón et al., 2001]. Además se considera que los centros de las gaussianas sólo pueden tomar valores provenientes de un conjunto finito $C = \{c_1, c_2, ..., c_{N_c}\}$. De esta forma, cada antecedente de una regla se representa con un número entero que codifica una de las distintas combinaciones posibles de valores (media, desviación) de las funciones de pertenencia asociadas a la entrada externa o al estado correspondiente. En las pruebas realizadas, existen 5 combinaciones distintas, codificadas con números del 1 al 5, tal y como se muestra en la tabla 6.1.

Centro	Desviación	Código
0.1	0.2	1
0.3	0.2	2
0.5	0.2	3
0.7	0.2	4
0.9	0.2	5

Tabla 6.1. Codificación para los antecedentes de las reglas de la máquina finita de estados borrosa en la implementación tipo Pittsburgh.

Por lo tanto, la matriz de antecedentes se codifica con números enteros pertenecientes al intervalo [1,5]. Cada número representa un nivel distinto de activación de la señal externa (primera columna) o de la activación de los estados (restantes columnas). En la tabla 6.2 se presenta un ejemplo de una matriz de antecedentes codificada de esta manera, con 10 reglas y 4 estados.

La definición de la matriz de consecuentes está ligada a la tipología TSK de los sistemas de inferencia borrosos que componen la máquina de estados. Esta matriz, se compone de constantes reales con valores en el intervalo [0,1]. Estos valores representan el nivel de activación de cada estado, que la regla correspondiente establece como consecuente. En la tabla 6.2 se presenta un ejemplo de una matriz de consecuentes codificada de esta manera.

Matriz de antecedentes					Matriz de consecuentes			
5	1	2	3	5	0.2753	0.0330	0.3235	0.3682
5	2	2	1	5	0.0376	0.1427	0.2541	0.5656
2	3	3	1	5	0.3740	0.3870	0.0133	0.2258
2	5	1	2	4	0.2277	0.1615	0.1573	0.4535
3	5	2	3	3	0.0218	0.2858	0.2055	0.4869
1	1	5	2	1	0.1908	0.3408	0.1578	0.3106
1	4	5	4	4	0.2813	0.1742	0.2829	0.2615
1	5	2	2	1	0.0542	0.1922	0.1121	0.6415
2	5	4	2	4	0.3086	0.3610	0.0083	0.3221
2	1	2	3	5	0.0045	0.1568	0.1152	0.7235

Tabla 6.2. Ejemplo de una máquina codificada en la implementación Pittsburgh.

6.3.2.2 Funcionamiento detallado del algoritmo.

Este algoritmo consta de 5 pasos, como se puede apreciar en el esquema presentado en la figura 6.1. Además, es necesario una inicialización previa que se detalla a continuación.

1. Se especifica el conjunto de entrenamiento: se suministra al algoritmo el conjunto de las N_t trazas catalogadas $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{N_t}}\}$. Cada una de estas trazas constituye la entrada externa de la máquina finita de estados borrosa. Además de las trazas, es necesario introducir la catalogación de las mismas. Esto se realiza en los vectores $G_1 = \{g_1^1, g_2^1, ..., g_{l_n}^1\}$ y $G_2 = \{g_1^2, g_2^2, ..., g_{l_a}^2\}$, que contienen los índices de las trazas del conjunto de entrenamiento X que pertenecen a la clase **A** y a la clase **N** respectivamente.

2. Además, hay que proporcionar los conjuntos $P_i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, ..., \mathbf{p}_{N_{o_i}}^i\}$, donde cada $\mathbf{p}_j^i$ representa un conjunto de parámetros elegible para las funciones de pertenencia del antecedente i.

3. Declaración de parámetros. Los parámetros que se van a usar durante el algoritmo son los siguientes:
   - Parámetros relacionados con la generación de máquinas de estado borrosas: número de reglas y número de estados de cada máquina (*num_reglas*, *num_estados)*
   - Tamaño de la población: *num_maquinas*.
   - Parámetros utilizados en el proceso de evaluación de las máquinas de estados borrosas: niveles de activación iniciales de los estados antes de comenzar a evaluar el primer dato de una traza (*stini)*.
   - Parámetros utilizados en el proceso de cálculo de la aptitud de las máquinas de estados borrosas. Hay que fijar el estado que será considerado como estado de detección (aquel sobre el que se medirá el parámetro de reactividad) y el umbral de detección que intervienen en el cálculo del parámetro de reactividad de cada máquina: respectivamente *num_detec* y *param_alta*.
   - Parámetros utilizados en el proceso de selección de los individuos de la población. Fracción de los mejores individuos que sobreviven y pasan a la siguiente generación: *alfa*.
   - Parámetros utilizados en el proceso de repoblación. Por una parte, las probabilidades de aplicación de las operaciones de reproducción, cruce y

mutación, respectivamente *p1, p2, p3*. Y por otra, los parámetros que regulan la operación de mutación sobre las máquinas de estado borrosas, que son el número de reglas a mutar y el número de elementos dentro de cada regla a mutar, respectivamente *mutar_reglas* y *mutar_elementos*.

Una vez que se han realizado estos pasos previos, el algoritmo comienza su ejecución normal. A continuación, se describen con detalle todos los pasos del algoritmo Pittsburgh.

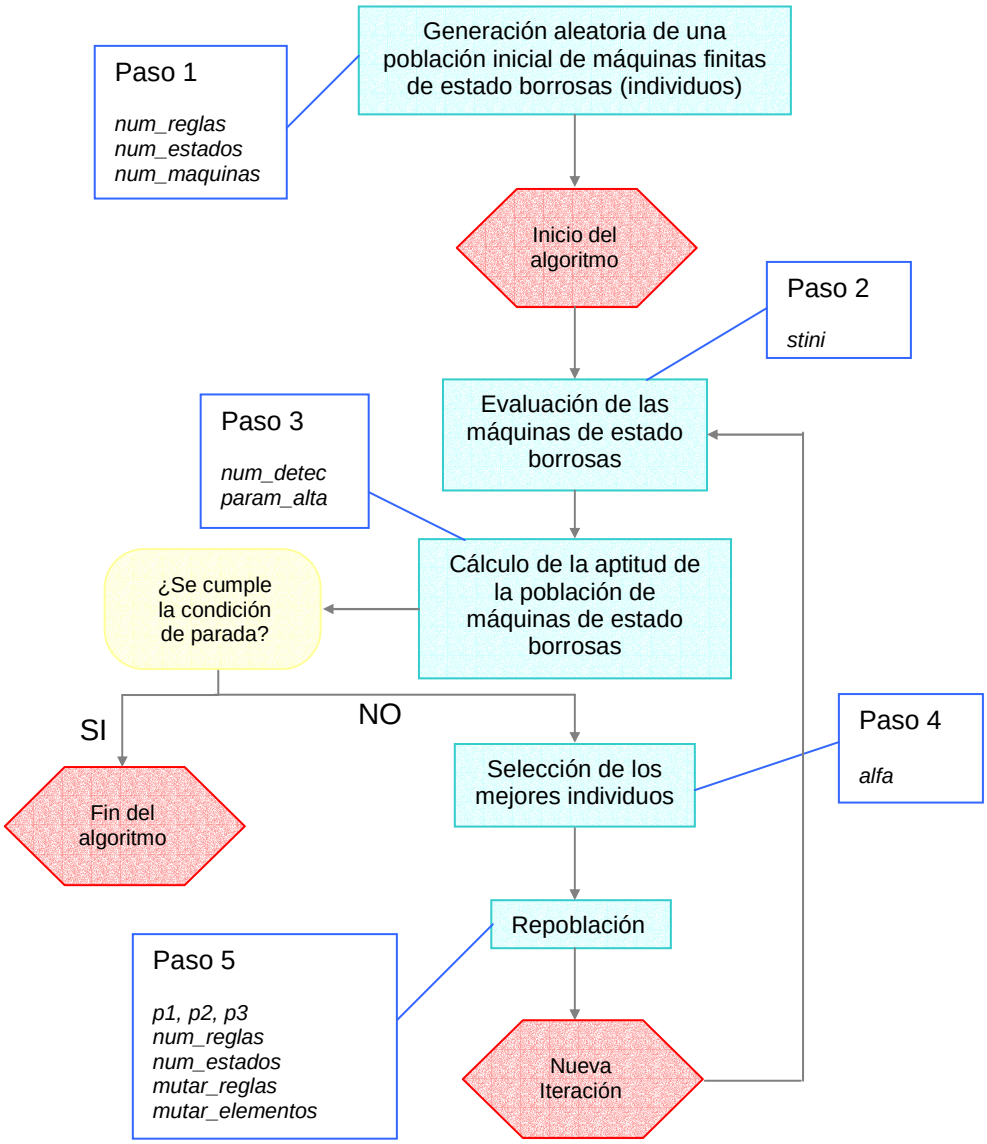


Figura 6.1. Esquema de la implementación Pittsburgh.

Paso 1: Generación inicial aleatoria de una población de individuos.

Los individuos son máquinas de estados borrosas. Como se explicó anteriormente, cada individuo de la población está representado por una matriz de antecedentes y una matriz de consecuentes. En este paso se generan aleatoriamente los individuos de la población inicial de la que se parte.

La función que genera la población inicial acepta como entrada:

- El número de reglas que debe tener cada máquina de la población (parámetro *num_reglas*).
- El número de estados que debe tener cada máquina de la población (parámetro *num_estados*).
- La configuración de la máquina de estados, es decir la cardinalidad del conjunto de posibles valores para los parámetros de la función de pertenencia a utilizar en cada antecedente.
- El número total de máquinas que componen la población, es decir, el número de máquinas que se deben generar aleatoriamente (parámetro *num_maquinas*).

La matriz de antecedentes se forma utilizando los índices de los elementos incluidos en los conjuntos P_i de parámetros admisibles para cada antecedente. Estos índices se escogen al azar de entre los posibles. Por otra parte, los elementos de la matriz de consecuentes se toman como constantes reales en el intervalo [0,1] generadas aleatoriamente con una distribución de probabilidad uniforme en dicho intervalo.

Paso 2: Evaluación de los individuos.

En esta etapa, el algoritmo evalúa cada máquina de estados borrosa incluida en la población con todas las trazas del conjunto de entrenamiento. El proceso supone la evaluación de cada dato de la traza, tomada como entrada externa, y usa la realimentación proporcionada por los niveles de activación de los estados. La evaluación del primer dato de la traza requiere un conjunto inicial de niveles de activación *stini,* fijado a priori en el algoritmo y común para todos los individuos de la población. Para evaluar es necesario utilizar la matrices de antecedentes para obtener los parámetros de las funciones de pertenencia de cada regla borrosa. Por cada traza con

la que es evaluada la máquina, se guarda la matriz de activaciones de los estados y la matriz de activaciones de las reglas.

La función que realiza la evaluación de los individuos de la población acepta como entrada:

- Conjuntos de parámetros posibles $P_i = \left\{ \mathbf{p}_1^i, \mathbf{p}_2^i, ..., \mathbf{p}_{N_{o_i}}^i \right\}$ asociados a los antecedentes de la máquina de estados borrosa.
- Las matrices de antecedentes y consecuentes de las máquinas de la población.
- La variable que contiene el conjunto de trazas de entrenamiento.
- El vector que contiene la activación inicial de los estados (*stini*).

La función devuelve a la salida el conjunto de las matrices de activación de los estados y el conjunto de las matrices de activación de las reglas de las máquinas, tras evaluar cada máquina con cada traza. La matriz de activación de los estados para la traza *k-ésima* procesada por la máquina *l-ésima* de la población $\mathbf{S}_{lk}$, tiene tantas columnas como estados ($\mathbf{n_s}$) y tantas filas como datos contiene la traza ($\mathbf{n}_d$). Así el elemento s_{ij}^{lk} representa la activación del estado j, tras el procesamiento del *i-ésimo* dato de la traza. Por otra parte, la matriz de activación de las reglas para la traza *k-ésima* en la máquina *l-ésima* de la población, $\mathbf{P}_{lk}$, se define de la forma usual para un sistema borroso: cada fila está relacionada con el procesamiento de un vector de entradas (en nuestro caso, entrada externa y activaciones de los estados) y cada columna se refiere a al nivel de activación de cada regla del sistema borroso. Las dimensiones de esta matriz serán: *num_reglas* columnas (número de reglas) y $\mathbf{n}_d$ filas. De esta manera, el elemento p_{ij}^{lk} representa la activación de la parte de antecedentes de la regla *j* al procesar el dato *i-ésimo* de la traza.

Paso 3: Cálculo de la función de aptitud.

En este paso se calcula la aptitud de cada individuo, es decir, se calcula una medida de la bondad de la clasificación que realiza cada máquina finita de estados borrosa sobre las trazas del conjunto de entrenamiento.

La función que calcula esta aptitud para la máquina *l-ésima*, acepta como entrada:

- El conjunto de las matrices de activación de los estados $\{\mathbf{S}_{l1}, \mathbf{S}_{l2}, ..., \mathbf{S}_{lN_t}\}$ de la máquina tras evaluar la máquina con cada traza.
- El número del estado de la máquina que actúa como estado de detección (parámetro *num_detec*).
- El umbral que tiene que sobrepasar el valor de la activación del estado de detección para considerar que está activado a alta (parámetro *param_alta*).
- Los vectores que contienen información sobre la clasificación de las trazas, *G1* y *G2* (recordemos que se trata de un proceso de aprendizaje supervisado).

Para calcular la aptitud de una máquina, se analizan las matrices de activación de los estados de la máquina resultantes de la evaluación la máquina con cada traza. Una vez fijado el estado de detección, *num_detec*, se contabiliza el número de muestras de una traza para las que la activación de este estado ha superado el umbral *param_alta* y se divide por el número total de muestras de la traza. Esta medida de reactividad del estado de detección se realiza para cada traza (en cada matriz de activación de los estados).

Es decir, para la máquina *l-ésima* de la población y la traza *k-ésima* de longitud $\mathbf{n_d}$, el parámetro de reactividad es:

$$p_r^{lk} = \frac{\sum_{i=1}^{\mathbf{n_d}} (s_{i\text{num\_detec}}^{lk} > \text{param\_alta})}{\mathbf{n_d}} \tag{6.6}$$

donde se supone que $(s_{i\text{num\_detec}}^{lk} > \text{param\_alta})$ vale 1 si se cumple la condición y 0 en caso contrario.

Como se ha comentado anteriormente, uno de los objetivos es que la máquina de estados borrosa resultante del proceso de búsqueda presente una reactividad bien diferenciada a las dos clases de trazas. Con este objetivo, se construye la medida de aptitud siguiente.

Se aplica un algoritmo clustering borroso no supervisado (*fuzzy c-means clustering*) para establecer una partición borrosa sobre el espacio del parámetro de reactividad. Si los valores resultantes de este parámetro para los dos tipos de clases ocupan zonas diferenciadas en este espacio, la máquina de estados borrosa estará funcionando conforme a nuestras expectativas, y el algoritmo de clustering podrá producir con éxito dos conjuntos borrosos convexos con funciones de pertenencia $\mu_A(\mathbf{x})$ y $\mu_N(\mathbf{x})$, que definen cada una de las clases. Diremos que $\mathbf{x} \in \mathbf{A}$, si $\mu_A(\mathbf{x}) \geq \mu_N(\mathbf{x})$, y $\mathbf{x} \in \mathbf{N}$, si $\mu_N(\mathbf{x}) > \mu_A(\mathbf{x})$.

El algoritmo de clustering borroso utilizado no devuelve funciones de pertenencia parametrizadas, sino los valores de pertenencia a cada una de las clases de los parámetros de reactividad calculados para las trazas. Es decir, para cada parámetro de reactividad calculado p_r^{lk} tendremos los pares (p_r^{lk}, μ_A^{lk}) y (p_r^{lk}, μ_N^{lk}), que especifican el valor del parámetro y su pertenencia a cada clase. La estimación de la pertenencia de un valor arbitrario x al conjunto borroso que define la clase $\mathbf{A}$ o al conjunto borroso que define la clase $\mathbf{N}$ se realiza entonces por interpolación lineal usando los pares descritos anteriormente.

El algoritmo de clustering borroso utilizado devuelve los centros representativos de cada clase, definido para la clase $\mathbf{A}$ como:

$$c_A^l = \frac{\sum_{j=1}^{N_r} p_r^{lj} \mu_A^{lj}}{\sum_{j=1}^{N_r} \mu_A^{lj}} \tag{6.7}$$

y con la definición análoga para la clase $\mathbf{N}$.

Una vez realizado el clustering supervisado se establece por convenio que el conjunto borroso obtenido cuyo centro representativo tenga un valor más próximo a 0 será el correspondiente a la clase $\mathbf{N}$ y el otro corresponderá a la clase $\mathbf{A}$. Entonces, para cualquier máquina de la población:

$$c_{\mathbf{N}}^l \leq c_A^l \tag{6.8}$$

Con este convenio y utilizando los valores de pertenencia a ambas clases de las trazas procesadas por la máquina de estado, se realiza la clasificación. A continuación contabilizamos el número de aciertos y fallos en la clasificación de las trazas de cada clase.

La aptitud final de la máquina se calcula según la expresión $1-\frac{m}{L}$, donde *m* es el número de trazas que se han clasificado correctamente en las dos clases, y *L* es el número de trazas total.

Tal y como se ha diseñado el proceso de cálculo de la aptitud, se puede observar que cuanto menor sea este valor, mejor es la clasificación de la máquina finita de estados borrosa.

Llegados a este punto, el algoritmo comprueba si la aptitud de alguna de las máquinas que componen la población está por debajo del parámetro introducido por el usuario *umbral_fitness*. Si es así, el algoritmo ha encontrado una máquina con un valor de la aptitud aceptable en los límites introducidos por el usuario y el algoritmo para su ejecución. Si no ocurre esto, el algoritmo continúa con su ejecución normal.

Paso 4: Selección de los mejores individuos.

En este paso se selecciona una porción de los mejores individuos de la población y se descarta el resto, con el objetivo de utilizar estos individuos en la posterior repoblación. Este tipo de algoritmo se denomina *algoritmo genético de estado estacionario* y se caracteriza porque una parte de la población pasa de una generación a la siguiente. Una de las ventajas de este algoritmo es que se puede reducir la carga computacional, evitando reevaluar aquellos individuos que pasan de una generación a la siguiente, aunque la tendencia a la convergencia prematura es mayor. El algoritmo de selección utilizado ordena todos los individuos de mejores a peores (de valores de la aptitud menores a mayores) y descarta un porcentaje de los peores. Este porcentaje se especifica con el parámetro *alfa* introducido por el usuario, regulando de esta manera el grado de solapamiento de las poblaciones.

La función que implementa la selección acepta como entrada:

- Las matrices de antecedentes y consecuentes de las máquinas que componen la población.
- Los valores de la aptitud de las máquinas de la población.
- La fracción de solapamiento (parámetro *alfa*).

La función devuelve a la salida las matrices de antecedentes y consecuentes de las máquinas que han sido seleccionadas para el proceso de repoblación. Estas máquinas pasan a formar parte de la población en la siguiente iteración, y además, serán utilizadas como individuos "padres" para generar nuevas máquinas en la repoblación.

El usuario introduce el parámetro *alfa* para controlar el porcentaje de individuos que se conservan. Por ejemplo, si *alfa* = 0.5, se descarta la peor mitad de la población y se selecciona la mejor mitad para la repoblación. Cuanto menor sea el valor de este parámetro, más diversidad se consigue en la población en la siguiente iteración, ya que se obliga al proceso de repoblación a generar más individuos nuevos.

Paso 5: Repoblación.

En este paso, se generan nuevos individuos por medio de la aplicación de operadores genéticos sobre los individuos supervivientes al proceso de selección, hasta alcanzar de nuevo el tamaño original de la población. Los operadores genéticos que se aplicarán son la reproducción, la mutación y el cruce.

La función que realiza la repoblación acepta como entrada:

- Las matrices de antecedentes y consecuentes de las máquinas seleccionadas para ejercer de "padres" en la repoblación.
- El número total de máquinas que compone la población (parámetro *num_máquinas*).
- El número de reglas de cada máquina de la población (parámetro *num_reglas*).
- El número de estados de cada máquina de la población (parámetro *num_estados*).
- Las probabilidades de selección de los operadores genéticos reproducción, mutación y cruce (parámetros *p1, p2* y *p3*, respectivamente).
- Número de reglas que se desean mutar sobre una máquina cuando está activa la operación de mutación (parámetro *mutar_reglas*).

- Número de elementos que se desean mutar sobre una regla de la máquina cuando está activa la operación de mutación (parámetro *mutar_elementos*).
- La configuración de la máquina de estados, es decir la cardinalidad del conjunto de posibles valores para los parámetros de la función de pertenencia a utilizar en cada antecedente.

La función devuelve como salida las matrices de antecedentes y consecuentes de las nuevas máquinas creadas. Posteriormente, la población para la siguiente iteración se construye uniendo los individuos seleccionados en el Paso 4 para realizar la repoblación con estos nuevos individuos.

En la repoblación, se elige probabilísticamente qué operador se va a aplicar. A continuación, se elige aleatoriamente un individuo "progenitor" (en el caso de la reproducción y la mutación) o dos individuos "progenitores" (en el caso del cruce) sobre los que aplicar el operador genético seleccionado. Recordemos que los progenitores se elegirán siempre del conjunto de individuos seleccionados para sobrevivir en la generación siguiente.

El efecto de los operadores genéticos se describe a continuación:

1. Reproducción. Este operador genera una nueva máquina que es una copia exacta de la máquina elegida como "progenitor". Ambas máquinas se introducirán posteriormente en la nueva población.

2. Mutación. Este operador genera una nueva máquina mutando el individuo progenitor. Ambas máquinas se introducirán posteriormente en la nueva población.

La operación de mutación utiliza los elementos siguientes:

- Las matrices de antecedentes y consecuentes de la máquina seleccionada para ser mutada.
- El número de reglas de cada máquina de la población (parámetro *num_reglas*).
- El número de estados de cada máquina de la población (parámetro *num_estados*).
- Número de reglas que se desean mutar sobre la máquina (parámetro *mutar_reglas*).

- Número de elementos que se desean mutar sobre una regla de la máquina (parámetro *mutar_elementos*).

Para mutar una máquina, se selecciona al azar un número determinado de reglas que se van a mutar. El número de reglas a mutar viene indicado en el parámetro *mutar_reglas*. Por lo tanto, la mutación no afectará a todo el cromosoma, sino a parte de él. Dentro de cada regla, se mutarán aleatoriamente un número determinado de elementos (antecedentes y/o consecuentes). El usuario debe especificar mediante el parámetro *mutar_elementos* cuántos elementos desea mutar dentro de una regla.

Cuando el elemento a mutar es un antecedente, se elige aleatoriamente un código de los posibles existentes para codificar los antecedentes (ver tabla 6.1). Cuando se muta un consecuente, se cambia el valor de ese elemento por un número real con valor entre [0,1], que representa el nivel de activación de ese estado.

3. Cruce. Una vez escogidas dos máquinas progenitoras para ser cruzadas, se elige al azar una regla de cada máquina. A continuación, se eligen al azar dos puntos o posiciones distintas permitidas dentro de las reglas como puntos cruce (para implementar un cruce de dos puntos). El operador de cruce intercambia el contenido entre estos puntos de las reglas de cada máquina. Estas dos máquinas nuevas resultantes se integran en la población de la siguiente iteración, junto con las máquinas progenitoras.

Por ejemplo, supongamos que se desean cruzar dos máquinas como las presentadas en la tabla 6.3.

Supongamos que, aleatoriamente, se elige para cruzar la regla número 2 de la primera máquina y la regla número 8 de la segunda máquina. Los puntos de corte se eligen también al azar, y resultan estar entre la posición número 3 y la posición número 8, tal y como se indica en la tabla 6.3. Se intercambia el contenido de estas reglas entre estas posiciones. Las máquinas resultantes se muestran en la tabla 6.4.

Se puede apreciar que el cruce implementado tiene efectos suaves sobre la población en la siguiente iteración, ya que las máquinas resultantes del cruce son ligeramente distintas a las máquinas padres de las que surgieron. Por lo tanto, la diversidad de la población en la siguiente iteración viene influenciada en mayor medida por la operación de mutación.

Máquina de estados borrosa 1								
Matriz de antecedentes					Matriz de consecuentes			
5	1	2	3	5	0.2753	0.0330	0.3235	0.3682
5	2	2	1	5	0.0376	0.1427	0.2541	0.5656
2	3	3	1	5	0.3740	0.3870	0.0133	0.2258
2	5	1	2	4	0.2277	0.1615	0.1573	0.4535
3	5	2	3	3	0.0218	0.2858	0.2055	0.4869
1	1	5	2	1	0.1908	0.3408	0.1578	0.3106
1	4	5	4	4	0.2813	0.1742	0.2829	0.2615
1	5	2	2	1	0.0542	0.1922	0.1121	0.6415
2	5	4	2	4	0.3086	0.3610	0.0083	0.3221
2	1	2	3	5	0.0045	0.1568	0.1152	0.7235
Máquina de estados borrosa 2								
Matriz de antecedentes					Matriz de consecuentes			
1	5	4	3	3	0.6154	0.0579	0.0153	0.9501
4	5	3	2	2	0.2311	0.7919	0.3529	0.7468
1	1	5	3	5	0.6068	0.9218	0.8132	0.4451
5	2	2	2	2	0.4860	0.7382	0.0099	0.9318
2	3	3	5	2	0.8913	0.1763	0.1389	0.4660
1	4	4	2	4	0.7621	0.4057	0.2028	0.4186
3	5	1	5	1	0.4565	0.9355	0.1987	0.8462
4	2	2	5	3	0.0185	0.9169	0.6038	0.5252
2	1	4	4	3	0.8214	0.4103	0.2722	0.2026
1	4	4	3	3	0.4447	0.8936	0.1988	0.6721

Tabla 6.3. Ejemplo de cruce entre dos máquinas de estados borrosas en la implementación Pittsburgh. Máquinas seleccionadas como “padres”.

Una vez completada la nueva población, la siguiente iteración del algoritmo comienza en la etapa 2. El algoritmo se detiene si encuentra una máquina que cumple las especificaciones del usuario, como ya se ha indicado anteriormente.

Una vez encontrado el mejor individuo (la mejor máquina de estados borrosa), el clasificador o función discriminante final estará compuesto por esta máquina junto con los datos de pertenencia a las clases borrosas de los parámetros de reactividad y centros provenientes de la aplicación del clustering borroso.

Máquina de estados borrosa 1									
Matriz de antecedentes						Matriz de consecuentes			
5	1	2	3	5		0.2753	0.0330	0.3235	0.3682
5	2	2	5	3		0.0185	0.9169	0.6038	0.5656
2	3	3	1	5		0.3740	0.3870	0.0133	0.2258
2	5	1	2	4		0.2277	0.1615	0.1573	0.4535
3	5	2	3	3		0.0218	0.2858	0.2055	0.4869
1	1	5	2	1		0.1908	0.3408	0.1578	0.3106
1	4	5	4	4		0.2813	0.1742	0.2829	0.2615
1	5	2	2	1		0.0542	0.1922	0.1121	0.6415
2	5	4	2	4		0.3086	0.3610	0.0083	0.3221
2	1	2	3	5		0.0045	0.1568	0.1152	0.7235
Máquina de estados borrosa 2									
Matriz de antecedentes						Matriz de consecuentes			
1	5	4	3	3		0.6154	0.0579	0.0153	0.9501
4	5	3	2	2		0.2311	0.7919	0.3529	0.7468
1	1	5	3	5		0.6068	0.9218	0.8132	0.4451
5	2	2	2	2		0.4860	0.7382	0.0099	0.9318
2	3	3	5	2		0.8913	0.1763	0.1389	0.4660
1	4	4	2	4		0.7621	0.4057	0.2028	0.4186
3	5	1	5	1		0.4565	0.9355	0.1987	0.8462
4	2	2	1	5		0.0376	0.1427	0.2541	0.5252
2	1	4	4	3		0.8214	0.4103	0.2722	0.2026
1	4	4	3	3		0.4447	0.8936	0.1988	0.6721

Tabla 6.4. Ejemplo de cruce entre dos máquinas de estados borrosas en la implementación Pittsburgh. Máquinas resultantes "hijas".

6.3.3 Sistemas tipo Michigan.

6.3.3.1 Introducción.

En el capítulo anterior se ha descrito en detalle la arquitectura de este tipo de sistemas. En esta investigación se ha utilizado este esquema para implementar un algoritmo de búsqueda en el espacio de los clasificadores borrosos concretos con los que trabajamos (máquinas finitas de estado borrosas).

Usando la nomenclatura para los esquemas tipo Michigan, el entorno bajo estudio va a ser el sistema clasificador borroso que se está adaptando. Esta es una diferencia importante respecto a otras técnicas de diseño de sistemas borrosos (GFRBS) que hacen uso de estrategias tipo Michigan: lo más frecuente es que el conjunto de reglas coadaptadas en el sistema clasificador sea la propia base de reglas borrosas del sistema, sin embargo en este caso se investiga la alternativa de situar la base de reglas

del sistema borroso como parte del entorno, que sufre modificaciones debido a la aplicación de meta-reglas incluidas en un sistema clasificador.

Continuando con la descripción del sistema desde el punto de vista de una arquitectura similar a la del algoritmo XCS, el programa de refuerzo será un sistema encargado de evaluar a este clasificador borroso con las trazas del conjunto de entrenamiento y de estudiar el error cometido en dicha clasificación. En la figura 6.2. podemos ver la arquitectura del sistema objeto de la investigación.

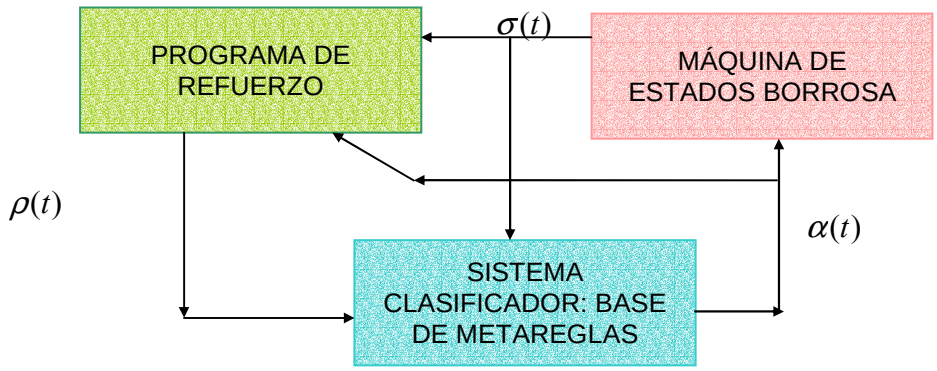


Figura 6.2. Arquitectura del sistema evolutivo de tipo Michigan.

Las meta-reglas del sistema clasificador se componen de dos partes. La primera parte, la condición de la meta-regla, es una descripción de la máquina de estados borrosa sobre la que esa meta-regla puede ser aplicada. Esta descripción comprende tanto los antecedentes como los consecuentes de la máquina y admite la utilización de comodines como medio de generalización sobre el espacio de todas las posibles máquinas de estados borrosas. La segunda parte se refiere al consecuente, donde la meta-regla propone acciones sobre una de las reglas de la máquina. Estas acciones se refieren a cada uno de los antecedentes y consecuentes de forma individual y son de dos tipos: dejar el elemento sin variación y cambiar el valor del antecedente /consecuente por otro.

El programa de recompensas premiará al parámetro *fuerza* de la meta-regla que se aplique en función de la mejora /empeoramiento que produzca en la máquina de estados borrosa. La selección de la meta-regla se producirá entre aquellas que encajen con la situación del entorno (es decir, la máquina de estados borrosa actual). La meta-regla aplicada es seleccionada, y sólo si se produce una mejora en la evaluación de la máquina, ésta es sustituida por el resultado de la modificación.

Esta última decisión tiene sus inconvenientes y sus ventajas. Por una parte, se puede apreciar que una de las principales dificultades con la que se puede encontrar este algoritmo es la de verse atrapado en un mínimo local. Supongamos que se ha modificado la máquina de estados borrosa hasta el punto de que no existen acciones que afecten a una sola de las reglas y que produzcan una mejora en la evaluación de la máquina. No quiere decir esto que la máquina obtenida sea la mejor de las soluciones, el óptimo global, sino que son necesarios *cambios simultáneos* en más de una regla para lograr una mejora. En esta situación el algoritmo se vería atrapado en un mínimo local. En resumen, los mínimos locales donde el sistema puede verse atrapado se caracterizan por ser máquinas de estado borrosas mejorables con modificaciones que afecten simultáneamente a más de una regla.

Por otra parte, se ha experimentado con el método y se ha encontrado que la convergencia hacia soluciones "aceptables" es más rápida y eficiente si se opta por sustituir la máquina de estados borrosa sólo en el caso de que se produzcan mejoras en la misma mediante las acciones propuestas por la meta-regla seleccionada.

Para entender mejor el proceso, vamos a representar el espacio de búsqueda por una rejilla de puntos como la que se muestra en la figura 6.3. Cada punto tiene un color en la escala de grises, suponemos que los colores más claros representan valores más elevados. En un algoritmo de búsqueda que no se base en el gradiente y que en cada paso de la búsqueda pueda acceder a cualquier posición del espacio, el mínimo global en B sería fácilmente alcanzable (por ejemplo, mediante una búsqueda aleatoria). Sin embargo, para mantener el paralelismo con lo que sucede en el algoritmo propuesto que dado un punto del espacio, supongamos que sólo es posible acceder a los puntos vecinos a través de las líneas continuas dibujadas (es decir, desde un punto solo se tiene acceso a los 4 primeros vecinos) y además ese acceso sólo se puede realizar si el punto destino tiene un valor inferior o igual al punto de partida.

En la situación que se plantea en la figura 6.3 a la izquierda, si el algoritmo de búsqueda se sitúa en el punto A, quedará atrapado en un mínimo local, ya que todos los puntos a los que el algoritmo de búsqueda tiene acceso, sus 4 primeros vecinos, tienen un valor superior. En la situación de la derecha, esto no sucede, ya que existen primeros vecinos a los que el sistema puede trasladarse.

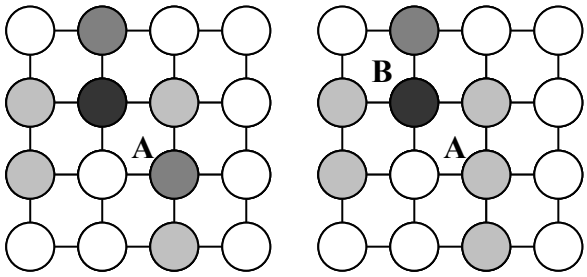


Figura 6.3. A la izquierda: mínimo local en A. A la derecha: en esta situación podemos acceder al mínimo global B.

Es decir, la hipótesis que estamos haciendo en este caso, es que dada cualquier máquina de estados en el espacio de búsqueda, es probable la existencia de un camino que implica la modificación regla a regla de la máquina de estados en el que la evaluación de la máquina se mantiene constante o va decreciendo y llega hasta un mínimo local aceptable.

La reducción del espacio de búsqueda cuando se utiliza esta hipótesis es considerable, no sólo porque se restringe la búsqueda a cambios en reglas individuales sino además porque no se permite la exploración de puntos del espacio de búsqueda que empeoren el valor de la evaluación.

Veamos ahora como es la inicialización del algoritmo. En los sistemas de tipo Michigan convencionales se genera aleatoriamente un conjunto de reglas inicial. En este algoritmo, la base de meta-reglas inicial es generada por un sistema de recubrimiento de forma similar al método empleado en el algoritmo XCS. En la primera iteración, este sistema crea meta-reglas que encajan con la máquina existente. En las iteraciones restantes, en el caso de que no exista en la base de meta-reglas un número mínimo de meta-reglas que encajen con la máquina de la iteración considerada, este sistema se ocupa de generar nuevas meta-reglas que encajen con la citada máquina.

En el algoritmo XCS, el mantenimiento de los clasificadores en la población se basa en una aptitud calculada a partir de una estimación del error en la predicción de la recompensa. Como ya se comentó en el capítulo anterior, esta forma de actuar tiene algunas ventajas respecto a una aptitud basada en la fuerza de los clasificadores, especialmente cuando la función de recompensa tiene sesgo y en problemas multipaso, como por ejemplo, la resolución de un laberinto. Sin embargo, nuestro planteamiento inicial para atacar el problema del diseño automático de la máquina de estados borrosa es considerarlo como un problema de un paso: dada una máquina de estados borrosa,

establecer la modificación que ha de hacerse en sus reglas para obtener el mejor discriminante posible en nuestro esquema. Esta aproximación no conduce a una solución óptima, puesto que presenta el riesgo de caer en mínimos locales como ya se ha comentado, pero se simplifican las dificultades asociadas a los problemas multipaso, como por ejemplo, la tendencia de producir sobre-generales fuertes e inicialización de parámetros además de simplificar el espacio de búsqueda. Por todo esto, la selección de las meta-reglas emplea una aptitud basada en la fuerza y hace descansar la capacidad de generalización del sistema en la utilización de comodines en las meta-reglas.

El algoritmo propuesto realiza las tareas del sistema de asignación de créditos mediante módulos que gestionan recompensas y penalizaciones sobre las fuerzas de las meta-reglas. Además, se introducen módulos que se ocupan del borrado y depurado del conjunto de meta-reglas, para eliminar aquellas consideradas no beneficiosas en la población. Estos últimos módulos realizan funciones similares a las descritas para los módulos equivalentes en el algoritmo XCS.

El sistema descubridor de clasificadores se implementa con un algoritmo genético que actúa sobre las meta-reglas que encajan con la máquina en la iteración actual (conjunto de encaje). El espacio de búsqueda del sistema descubridor es el de las meta-reglas que provocan modificaciones en reglas individuales de la máquina de estados, que como ya se ha comentado es bastante más reducido que el espacio de todas las posibles máquinas. La selección correcta de las meta-reglas depende de estimar unos valores adecuados de las fuerzas, para lo que es necesario que el sistema de asignación de créditos (recompensas) actúe durante un número de iteraciones. Por ello, el algoritmo genético no se dispara en todas las iteraciones del algoritmo, existe un módulo encargado de controlar su disparo cuando se cumplen ciertos requisitos, que tienen que ver con la antigüedad de las meta-reglas de la población o el número de iteraciones que han transcurrido desde la última vez que se procedió a ejecutar el algoritmo genético.

Antes de terminar esta discusión general sobre el algoritmo que se ha planteado, debemos comentar un detalle adicional sobre la aplicación de las meta-reglas. La condición de la meta-regla incluye una descripción de la máquina de estados borrosa, pero hay que precisar que en ella no es importante el orden de las reglas: el “encaje” se producirá con una máquina de estados borrosa que sea compatible con la que se describe en la condición de la meta-regla independientemente del ordenamiento de las reglas. Por otro lado, en la parte donde se describe la acción a realizar sobre la máquina

de estados borrosa se incluye un campo que especifica el número de la regla sobre la que se va a aplicar la acción. Este detalle puede parecer poco coherente con el hecho de que la condición de la meta-regla encaja independientemente del orden de las reglas, pero no es así.

El motivo es que los cambios en la máquina de estados borrosa se producen de forma suave y gradual a nivel de las reglas individuales y no implican el cambio en el orden de las mismas, por lo que imponer un número de regla sobre la que aplicar la acción sólo supone que estamos adaptando el sistema para la situación concreta en la que se encuentra el entorno, en este caso nuestra máquina de estados borrosa, y se usa la condición de la meta-regla de forma consistente con el hecho de que los cambios de orden de las reglas en la máquina no afectan a su funcionamiento.

A continuación, se describe en profundidad el algoritmo de Michigan implementado en este trabajo para encontrar las máquinas finitas de estado borrosas capaces de clasificar patrones.

6.3.3.2 Funcionamiento general.

El algoritmo se divide en 12 pasos, tal y como se puede apreciar en el diagrama de bloques presentado en la figura 6.4. Las flechas en trazo discontinuo indican el flujo del algoritmo durante su primera iteración y las flechas en trazo continuo indican el flujo en el resto de las iteraciones. Siguiendo el esquema de la figura 6.4, se presenta un resumen del funcionamiento del algoritmo.

El objetivo de este algoritmo es encontrar una máquina finita de estados borrosa que presente una eficiencia determinada en la clasificación de patrones, siguiendo un esquema de búsqueda tipo Michigan. El sistema de búsqueda tipo Michigan es considerablemente más rápido en nuestra implementación que el sistema de búsqueda tipo Pittsburgh. En la implementación tipo Pittsburgh, presentada en la sección anterior, el algoritmo trabaja con una población, donde cada individuo es una máquina de estados borrosa, y la determinación de la aptitud de los individuos exige la evaluación de todas ellas en cada iteración. En la implementación tipo Michigan, el algoritmo trabaja con una máquina, de modo que sólo se realiza una evaluación por iteración, aunque requiere de un número de iteraciones elevado para hacer converger las fuerzas asociadas a las meta-reglas y presenta el problema de los mínimos locales ya mencionado.

El algoritmo comienza generando aleatoriamente una máquina finita de estados borrosa con un número determinado de reglas y de estados (paso 1). La codificación de esta máquina de estados borrosa es igual a la explicada para el método tipo Pittsburgh. Los antecedentes de cada regla se representan con números que codifican las distintas combinaciones posibles de valores (media, desviación) de las funciones de pertenencia asociadas a la entrada externa y a cada uno de los estados (tabla 6.5).

Centro	Desviación	Código
0.1	0.2	1
0.3	0.2	2
0.5	0.2	3
0.7	0.2	4
0.9	0.2	5

Tabla 6.5. Ejemplo de codificación para los antecedentes de las reglas de la máquina finita de estados borrosa en la implementación tipo Michigan.

Los consecuentes de cada regla también se codifican con números. Se divide el intervalo [0,1] en un número determinado de puntos equidistantes. Cada uno de estos puntos es un posible valor para el consecuente y se codifica con un número distinto. Por ejemplo, tomamos 6 puntos equidistantes, de modo que los números que aparecen en la representación del consecuente de la regla representan uno de los valores recogidos en la tabla 6.6.

Valor del consecuente	Código
0	1
0.2	2
0.4	3
0.6	4
0.8	5
1	6

Tabla 6.6. Codificación para los consecuentes de las reglas de la máquina finita de estados borrosa en la implementación tipo Michigan.

Un ejemplo de máquina inicial generada al azar según lo descrito anteriormente se muestra en la tabla 6.7.

Antecedentes					Consecuentes			
entrada externa	estado 1	estado 2	estado 3	estado 4	estado 1	estado 2	estado 3	estado 4
2	4	3	5	3	3	3	2	2
3	3	3	5	4	4	3	4	5
5	1	4	2	5	2	1	1	4
4	1	5	4	1	4	6	1	4
4	2	2	5	4	2	1	6	1
3	3	2	5	1	6	3	1	4
4	3	3	1	5	6	6	1	1
4	2	3	5	1	5	4	3	3
1	2	2	4	5	6	5	2	1
5	5	4	2	2	4	3	2	2

Tabla 6.7. Ejemplo de máquina de estados borrosa codificada en la implementación tipo Michigan.

Tras evaluar la máquina (paso 2) con las trazas del conjunto de entrenamiento y calcular el valor de la función objetivo (paso 3) en base a la clasificación que realiza sobre las trazas de entrenamiento (error de clasificación), el algoritmo comprueba si este valor de función objetivo de la máquina está por debajo de un cierto umbral introducido por el usuario como condición de parada del algoritmo. Si es así, ya se ha encontrado una máquina aceptable (según las especificaciones del usuario) y se detiene la ejecución del algoritmo. Si no, el algoritmo continúa su ejecución normalmente.

Dependiendo de si se encuentra en la primera iteración o en cualquier otra, el algoritmo seguirá ejecutándose por caminos distintos. Supongamos que nos encontramos en la primera iteración (en la figura 6.4, el flujo indicado por las flechas discontinuas). La siguiente acción del algoritmo será aplicar una meta-regla sobre la máquina finita de estados borrosa. Al estar en la primera iteración, la base de meta-reglas aún está vacía, por lo tanto, el algoritmo primero genera una base de meta-reglas inicial mediante un proceso de recubrimiento (paso 8). Este recubrimiento genera un número determinado de meta-reglas que encajan con la máquina considerada en esta iteración.

A continuación, se explica con más detalle cómo es la estructura de estas meta-reglas.

Una meta-regla consta de 6 campos distintos:

1. El **campo de comparación**, que es lo que se compara con la matriz de antecedentes de la máquina finita de estados borrosa, elemento a elemento, para saber si hay encaje entre la máquina y la meta-regla.

Esta parte tiene las mismas dimensiones que la matriz de antecedentes de la máquina. Sus elementos se codifican del mismo modo, pero el libro de códigos incluye un símbolo más: "0", que hace la función de *don't care*. Por lo tanto, el campo de comparación es una matriz compuesta por los números permitidos que codifican los antecedentes de las reglas de la máquina y por símbolos *don't care*. Para ver si una meta-regla encaja con una máquina determinada, se compara la matriz de antecedentes de la máquina con la matriz definida en el campo de comparación de la meta-regla. Si ambas matrices coinciden, se dice que la meta-regla encaja con la máquina. En la tabla 6.8 se muestra un ejemplo en el que existe encaje entre el campo de comparación de una meta-regla y la matriz de antecedentes de una máquina de estados borrosa.

Matriz de Antecedentes					Parte de comparación de la meta-regla				
entrada externa	estado 1	estado 2	estado 3	estado 4	entrada externa	estado 1	estado 2	estado 3	estado 4
2	4	3	5	3	0	0	0	0	3
3	3	3	5	4	0	3	0	5	4
5	1	4	2	5	5	1	0	2	0
4	1	5	4	1	0	0	5	4	1
4	2	2	5	4	4	2	2	0	4
3	3	2	5	1	3	0	0	0	1
4	3	3	1	5	0	3	0	0	5
4	2	3	5	1	4	0	3	5	0
1	2	2	4	5	1	2	0	0	0
5	5	4	2	2	5	0	4	0	2

Tabla 6.8. Ejemplo de encaje entre el campo de comparación de una meta-regla y la matriz de antecedentes de una máquina finita de estados borrosa.

Aunque en el ejemplo de la tabla 6.8 el ordenamiento de la matriz de comparación en la meta-regla y la matriz de antecedentes coinciden, no es un requisito como ya se comentó previamente. Basta que cada regla de la máquina de estados borrosa esté representada por una descripción compatible en la matriz de comparación de la meta-regla.

2. El **campo de acción sobre los antecedentes de la máquina**. Una vez que se comprueba que una meta-regla encaja con la matriz de antecedentes de la máquina, si ésta es seleccionada para aplicarse sobre la máquina, esta parte de la meta-regla define los cambios a realizar sobre los antecedentes de una de las reglas de la máquina. El número de la regla de la máquina de estados borrosa

sobre la que se aplicarán los cambios viene dado en el último elemento del campo de acción sobre los consecuentes de la máquina (tercer campo).

La parte de acción sobre los antecedentes de una meta-regla es un vector que codifica las funciones de pertenencia asociadas a la señal externa y cada uno de los estados de la máquina del mismo modo en que se codifica el antecedente de la regla para la máquina finita de estados borrosa. La única diferencia es que se incluye un símbolo más en la codificación, el "0", que de nuevo, es un *don't care*.

La acción que se realiza sobre el antecedente de la regla elegida de la máquina consiste en cambiar elemento a elemento esa parte de la regla por los elementos de la parte de acción sobre los antecedentes de la meta-regla. Sólo en el caso en que haya un "0" en la parte de la meta-regla, se mantiene el valor que existía en ese elemento de la regla de la máquina.

Por ejemplo, la siguiente parte de acción sobre los antecedentes de la meta-regla:

[1 0 1 5 0]

cambiaría el antecedente de la regla siguiente:

[3 5 3 3 2]

por:

[1 5 1 5 2]

3. El **campo de acción sobre los consecuentes de la máquina**. Una vez que se comprueba que una meta-regla encaja con la matriz de antecedentes de la máquina, si es seleccionada para aplicarse sobre la máquina, esta parte de la meta-regla define los cambios a realizar sobre los consecuentes de una de las reglas de la máquina. La regla de la máquina sobre la que se aplicarán los cambios está codificada en el último elemento de este campo.

   La parte de acción sobre los consecuentes de una meta-regla es un vector que codifica los posibles valores de los consecuentes de la máquina del mismo modo en que se codifica el consecuente de la regla para la máquina finita de

estados borrosa. La única diferencia es que se incluye un símbolo más en la codificación, el "0", que de nuevo, es un *don't care*.

La acción que se realiza sobre el consecuente de la regla elegida de la máquina consiste en cambiar elemento a elemento esa parte de la regla por los elementos de la parte de acción sobre los consecuentes de la meta-regla. Sólo en el caso en que haya un "0" en la parte de la meta-regla, se mantiene el valor que existía en ese elemento de la regla de la máquina.

Por ejemplo, la siguiente parte de acción sobre los consecuentes de la meta-regla:

[5 0 5 2 3]

cambiaría el antecedente de la regla siguiente (suponiendo que es el consecuente correspondiente a la regla número 3 de la máquina):

[2 1 1 4]

por:

[5 1 5 2]

4. El **campo de la fuerza** de la meta-regla. Todas las meta-reglas tienen un valor de "fuerza", que es una estimación de la bondad de su comportamiento. Este campo será modificado en el algoritmo por el sistema de asignación de recompensas. Una meta-regla presenta un buen comportamiento si, cuando es aplicada sobre la máquina, introduce cambios en ella que hacen que mejore el valor de la función objetivo (que disminuya su error en la clasificación de las trazas de entrenamiento). Cuanto mejor sea el comportamiento de una meta-regla, mayor valor de fuerza tendrá en este campo.

5. El **campo de la experiencia** de la meta-regla. La "experiencia" de una meta-regla es el número de veces que se ha aplicado sobre una máquina. Este campo contabiliza las aplicaciones de la meta-regla, y por lo tanto, será modificado en la evolución del algoritmo dependiendo del número de veces que se aplique la meta-regla.

6. El **campo de la antigüedad** de la meta-regla. En este campo se guarda el número de iteración *k* en la que la meta-regla ha sido creada. En algunas etapas del algoritmo se usará este campo para calcular la antigüedad que tiene la meta-regla como la diferencia entre el valor guardado en este campo (número de iteración en que fue creada la meta-regla) y el número de la iteración actual en la que se considera.

Una vez generada una base inicial de meta-reglas, que encajan con la máquina, se selecciona la que se va a aplicar sobre ella (paso 9). La aplicación varía la estructura de la máquina (paso 10), y se da comienzo a una nueva iteración en el algoritmo.

A partir de la segunda iteración, el algoritmo sigue el flujo indicado por las flechas de trazo continuo de la figura 6.4. La nueva máquina (máquina con variaciones introducidas por una meta-regla aplicada en la iteración anterior) se evalúa con las trazas del conjunto de entrenamiento (paso 2).

Se calcula el valor de función objetivo que presenta en la clasificación de dichas trazas (paso 3), y se comprueba si se ha llegado al error de clasificación deseado. Si es así, el algoritmo deja de ejecutarse y ofrece como resultado la máquina encontrada. Si no, continúa su curso normal, realizando un proceso de recompensa (paso 4). El proceso de recompensa compara el valor de la función objetivo que mide la eficacia de la máquina en esta iteración con el valor de la función objetivo que presentaba en la anterior iteración, antes de ser modificada por la meta-regla que se aplicó sobre ella. Si este valor ha mejorado, la meta-regla aplicada ha mejorado la estructura de la máquina, y por tanto, se recompensará positivamente aumentando su valor de fuerza (cuarto campo de la meta-regla). Si la eficiencia de la máquina ha empeorado, la meta-regla no ha introducido cambios beneficiosos, por lo que será recompensada negativamente, disminuyendo su valor de fuerza. Si el valor de la función objetivo no sufre variación, la meta-regla será penalizada, disminuyendo su valor de fuerza, por no haber introducido cambios apreciables en la máquina.

En el siguiente paso (paso 5) se realiza un proceso de depurado sobre la base de meta-reglas. En este proceso, se recorre toda la base buscando meta-reglas que tengan todos los elementos a “0” (*don´t care*) en los campos de acción sobre los antecedentes y sobre los consecuentes (campos segundo y tercero) y se eliminan de la base, por no

introducir ninguna acción sobre los antecedentes y consecuentes de las reglas de la máquina.

En el paso 6 se realiza un proceso de borrado, en el que se recorre toda la base de meta-reglas y se eliminan las peores meta-reglas existentes en la misma, es decir, las meta-reglas con más experiencia (que han sido aplicadas más veces) con menos fuerza (han sido recompensadas negativamente o penalizadas varias veces por no haber introducido cambios beneficiosos sobre la máquina cuando se aplicaron).

Las siguientes acciones del algoritmo están orientadas a modificar la máquina actual mediante la aplicación de una meta-regla. Lo primero que se hace es buscar las meta-reglas de la base de meta-reglas que encajan con la matriz de antecedentes de la máquina (paso 7). Si el número de encajes es menor que un mínimo de encajes exigido por el usuario, entonces se ejecuta un proceso de recubrimiento (paso 8) que genera tantas meta-reglas nuevas (que encajen con la matriz de antecedentes) como hagan falta para llegar al número mínimo de encajes exigido. En este subconjunto de meta-reglas que encajan, *M*, se elige la meta-regla a aplicar (paso 9) y se aplica (paso 10).

Antes de finalizar la iteración y comenzar otra nueva, se analiza si se debe disparar una algoritmo genético para renovar la población de meta-reglas (paso 11). Este algoritmo se llevaría a cabo sobre las meta-reglas de *M* (meta-reglas que encajan con la máquina en esta iteración). El proceso de decisión de arranque del algoritmo genético se ha implementado de dos modos distintos. En el primero, esta decisión se toma teniendo en cuenta la antigüedad de las meta-reglas. Si un porcentaje de las meta-reglas de *M* tienen suficiente antigüedad (su antigüedad es mayor que un umbral introducido por el usuario), se considera que es necesario disparar un algoritmo genético para renovar la base. En el segundo modo, se disparan los algoritmos genéticos a una frecuencia constante, determinada por el usuario mediante un parámetro que indica el número de iteraciones fijas que debe existir entre un algoritmo genético y otro. En caso de tener que ejecutarse (paso 12), se seleccionan las mejores meta-reglas de *M* (población a repoblar) y a partir de éstas, se generan nuevas meta-reglas mediante la aplicación de tres operadores genéticos: la reproducción, la mutación y el cruce.

A continuación, se explican en profundidad los pasos y procesos realizados en el algoritmo Michigan, así como los parámetros involucrados en cada uno.

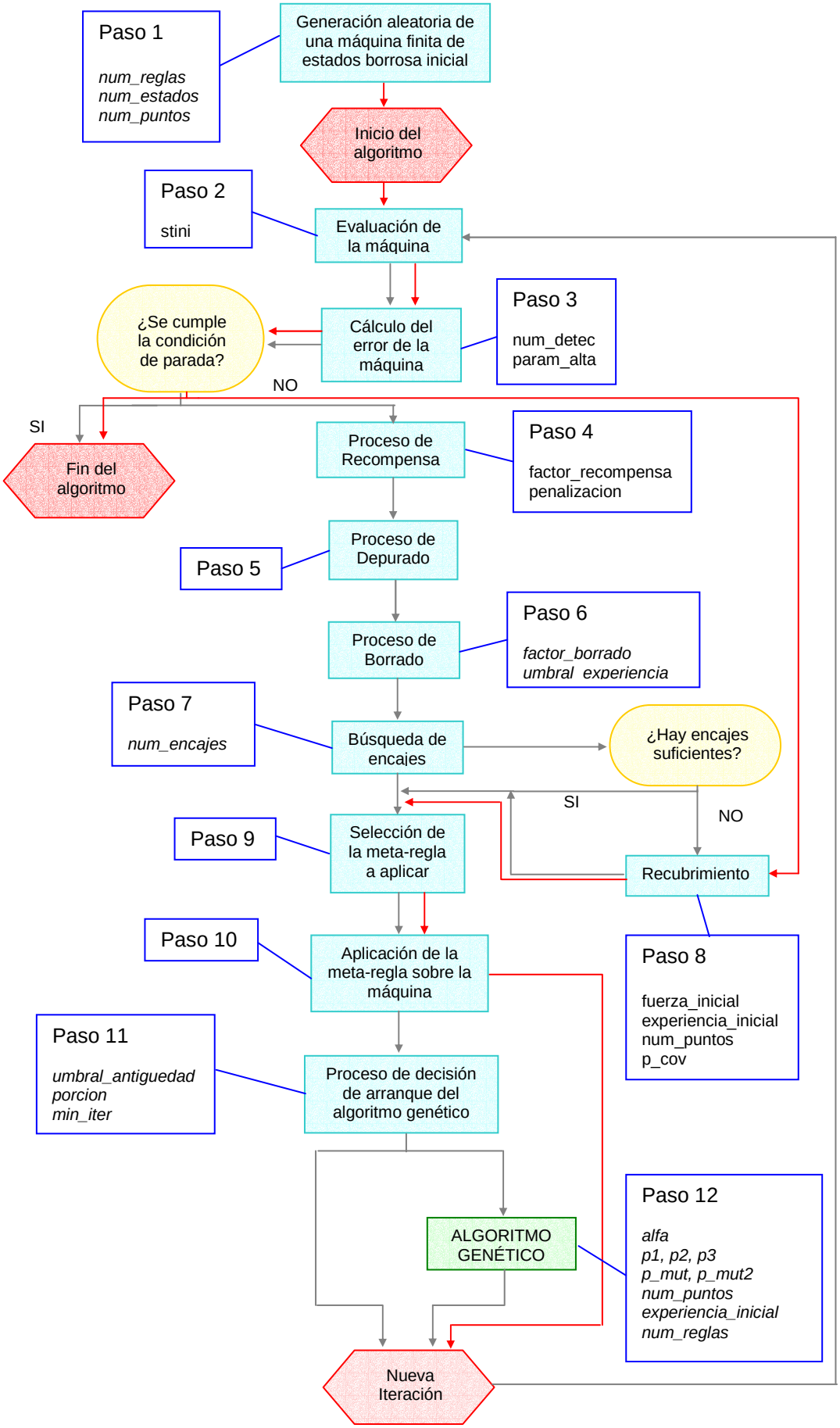


Figura 6.4. Esquema de la implementación tipo Michigan.

6.3.3.3 Funcionamiento detallado del algoritmo.

Los pasos previos a la ejecución del algoritmo son los siguientes:

1. Se especifica el conjunto de entrenamiento: se suministran las trazas catalogadas que queremos clasificar en dos clases. Estas trazas serán las entradas externas de la máquina finita de estados borrosa. El aprendizaje será supervisado por lo que en estos vectores, debemos pasar información al algoritmo de la verdadera clasificación de las trazas que constituyen el conjunto de entrenamiento.

2. Se inicializan los vectores que contienen la información sobre los parámetros de las funciones de pertenencia asociadas a la entrada externa y a los estados de la máquina, para la parte de los antecedentes.

   Las máquinas finitas de estados borrosas que utilizamos tienen asociadas a sus estados funciones de pertenencia gaussianas.

3. Inicialización de parámetros. A continuación se presenta la lista de parámetros que se usarán en el algoritmo. Junto con el nombre del parámetro, se acompaña una breve descripción que se ampliará en la descripción detallada del algoritmo.

   - Parámetros utilizados en el proceso de generación inicial de una máquina de estados borrosa (paso 1): número de reglas, número de estados y número de intervalos en los que se divide el intervalo [0,1] para la asignación de consecuentes (*num_reglas*, *num_estados*, *num_puntos)*.
   - Parámetros utilizados en el proceso de evaluación de las máquinas de estados borrosas (paso 2): activación inicial de los estados (*stini)*.
   - Parámetros utilizados en el proceso de cálculo del valor de la función objetivo de las máquinas de estados borrosas (paso 3): estado de detección y umbral para el cálculo del parámetro de reactividad (*num_detec*, *param_alta)*.
   - Parámetros utilizados en el proceso de recompensa y penalización (paso 4): factor de proporcionalidad para determinar las recompensas postivas y negativas y constante de penalización (*factor_recompensa, penalización)*.

- Parámetros utilizados en el proceso de borrado (paso 6): número máximo de elementos que se desea borrar (no se eliminarán más candidatos aunque hayan más individuos con las condiciones requeridas) (*N*), factor que se aplica para obtener la fuerza umbral que se utiliza en la asignación de votos (*factor_borrado*), experiencia mínima de una meta-regla para ser considerada para el borrado (*umbral_experiencia*).
- Parámetros utilizados en el proceso de búsqueda de encajes (paso 7): número mínimo de meta-reglas que se requieren para formar el conjunto de encaje (*num_encajes*).
- Parámetros utilizados en el proceso de recubrimiento (paso 8): fuerza que se asigna a las nuevas meta-reglas que se crean en el proceso (*fuerza_inicial*), experiencia que se asigna a las nuevas meta-reglas que se crean en este proceso (*experiencia_inicial*), parámetro usado para la creación aleatoria de los consecuentes (*num_puntos*), probabilidad de que un antecedente de la máquina copiado a la parte de condición de la meta-regla pase a ser un comodín (*p_cov*).
- Parámetros utilizados en el proceso de decisión de arranque de un algoritmo genético (paso 11): número de iteraciones desde la última aplicación del GA tras el cual se volverá a disparar el proceso evolutivo (regla de disparo periódica) (*min_iter*), fracción de meta-reglas que deben superar un umbral de antigüedad establecido en el conjunto de encaje para que se desencadene el algoritmo genético (regla de disparo no periódica) (*porción*), umbral de antigüedad utilizado en la regla de disparo no periódica (*umbral_antigüedad*).
- Parámetros utilizados en el algoritmo genético (paso 12): grado de solape entre poblaciones sucesivas (*alfa*), probabilidad de aplicar el operador de curce (*p1*), probabilidad de aplicar el operador de mutación *(p2*), probabilidad de aplicar el operador de reproducción (*p3*), parámetro que establece la probabilidad de que un elemento de la meta-regla sufra una mutación (*p_mut*), parámetro que permite regular la probabilidad de que la mutación resulte en un comodín o bien se produzca una sustitución al azar del elemento (*p_mut2)*.

- Parámetros de control del bucle: parámetro que establece el criterio de parada del algoritmo (*umbral_fitness*).

4. Inicialización de variables. Las variables que se declaran e inicializan a vectores vacíos al comienzo del algoritmo son las siguientes:
   - *población*: variable en la que se guarda la base de meta-reglas de cada iteración.
   - *M*: variable en la que se guardan las meta-reglas que encajan en cada iteración.

5. Inicialización de registros. Los registros que se declaran e inicializan a vectores vacíos al comienzo del algoritmo son los siguientes:
   - *registro_fitness*: registro en el que se guarda el valor de la función objetivo de la máquina finita de estados borrosa en cada iteración. Se consulta para determinar si la nueva máquina modificada es mejor que la última o por el contrario se debe restituir esta última por ser mejor.

Una vez que se han realizado estos pasos previos, el algoritmo comienza su ejecución normal. A continuación, se describen con detalle todos los pasos del algoritmo Michigan:

Paso 1: Generación aleatoria de una máquina finita de estados borrosa inicial.
La generación al azar de una máquina de estados borrosa requiere:

- Cardinalidad de los conjuntos de parámetros posibles $P_i = \left\{\mathbf{p}_1^i, \mathbf{p}_2^i, ..., \mathbf{p}_{N_{o_i}}^i\right\}$ asociados a los antecedentes de la máquina de estados borrosa.
- El número de reglas (parámetro *num_reglas*).
- El número de estados (parámetro *num_estados*).
- El número de valores representativos equidistantes en el intervalo [0,1] deseados para generar los consecuentes de las reglas de la máquina finita de estados borrosa (parámetro *num_puntos*).

El algoritmo comienza generando aleatoriamente una máquina finita de estados borrosa con un número determinado de reglas (*num_reglas)* y de estados (*num_estados*).

Los antecedentes de cada regla se van generando mediante la elección al azar de un índice sobre los conjuntos de parámetros posibles $P_i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, ..., \mathbf{p}_{N_{oi}}^i\}$ para las funciones de pertenencia asociadas a la entrada externa y a cada uno de los estados de la máquina (en la tabla 6.5 se muestra un ejemplo de esta codificación). Este número es el que aparece en la representación del antecedente de la regla.

Para generar los consecuentes de cada regla, se divide el intervalo [0,1] en *num_puntos* equidistantes, y se elige al azar uno de estos puntos para el consecuente. Cada punto se codifica con un número entre 1 y *num*_puntos. Para el caso particular de *num_puntos* = 6 se muestra la codificación en la tabla 6.6.

Paso 2: Procesamiento de las trazas mediante la máquina finita de estados borrosa. Esta evaluación requiere de los siguientes elementos.

- La matriz que contiene los antecedentes de las reglas y la matriz que contiene los consecuentes de las reglas de la máquina finita de estados borrosa a evaluar.
- El conjunto de trazas de entrenamiento previamente catalogadas.
- El conjunto con los posibles parámetros para las funciones de pertenencia de cada antecedente (posibles valores de las medias y desviaciones de las gaussianas para la entrada externa y para cada uno de los estados).
- El vector que contiene la activación inicial de los estados (*stini*).
- El número de puntos utilizado para generar los consecuentes de las reglas de la máquina finita de estados borrosa (parámetro *num_puntos*).

Esta función devuelve como salida el conjunto de las matrices de activación de los estados de la máquina y el conjunto de las matrices de activación de las reglas de la máquina, tras evaluar la máquina con cada traza.

La función evalúa la máquina con cada traza del conjunto de entrenamiento. Para ello, previamente debe acceder a la matriz de antecedentes y la matriz de consecuentes para construir la máquina de estados borrosa. Tras la evaluación de la traza, se obtienen las matrices de activación de los estados y de las reglas de la máquina.

Paso 3: Evaluación de la máquina finita de estados borrosa.

En este paso se evalúa la máquina de estados borrosa, es decir, se calcula una medida de la bondad de la clasificación que realiza la máquina en su estado de diseño actual, sobre las trazas del conjunto de entrenamiento. El cálculo a realizar es el mismo que el que ya se describió en la evaluación del valor de la función objetivo de las máquinas que constituyen la población en la arquitectura Pittsburgh, por lo que no se repetirá aquí la explicación del procedimiento.

Paso 4: Proceso de recompensa.

Este paso se ejecuta a partir de la segunda iteración del algoritmo. Es necesario que la máquina se haya evaluado como mínimo dos veces antes de poder aplicarle la recompensa.

La función que se utiliza para implementar el programa de recompensas acepta como entrada:

- El índice de la meta-regla aplicada (*num_meta*).
- La variable que contiene la base de meta-reglas existente en la iteración actual.
- Un factor utilizado en el proceso de recompensa de las meta-reglas (parámetro *factor_recompensa*).
- Un factor que interviene directamente en el proceso de penalización de la meta-regla, disminuyendo su fuerza (parámetro *penalizacion*).
- El registro *registro_fitness*, que contiene los valores de la función objetivo de la máquina en las distintas iteraciones.
- Una matriz en la que se guardan las metareglas que encajan en cada iteración (*M*).

La recompensa puede ser positiva (el factor *factor_recompensa* influirá entonces en el aumento de la fuerza de la meta-regla recompensada positivamente) o negativa (el factor *factor_recompensa* influirá en este caso en la disminución de la fuerza de la meta-regla recompensada negativamente).

Esta función devuelve como salida: la variable que contiene la base de meta-reglas con las fuerzas de estas meta-reglas actualizadas y una bandera que indica si ha habido recompensa positiva (*rec*).

Cuando se llama a esta función, la máquina ha sido evaluada como mínimo dos veces (una vez en la iteración actual y otra vez en la anterior), y se ha aplicado como mínimo una meta-regla (en la iteración anterior).

El objetivo de esta rutina es analizar si los cambios introducidos por la meta-regla aplicada en la máquina han sido beneficiosos. Para ello, se compara el valor de la función objetivo que presenta la máquina después del cambio (iteración actual) con el que presentaba antes del cambio (iteración anterior). En función del resultado de esta comparación, se recompensará (positivamente o negativamente) o se penalizará a la meta-regla que introdujo el cambio. Se calcula la diferencia de valores de la función objetivo entre las dos iteraciones y se realiza una de las siguientes acciones:

Si el valor de la función objetivo de la máquina actual ha mejorado (es decir, si ha disminuido su valor respecto al que presentaba en la anterior iteración), se recompensa positivamente la meta-regla que produjo el cambio aumentando su fuerza según la siguiente expresión:

$$fuerza = fuerza + (factor\_recompensa \times mejora \times num\_aleat) \qquad (6.9)$$

donde *factor_recompensa* es un parámetro introducido por el usuario, *mejora* es un valor positivo con la diferencia existente entre los dos valores de la función objetivo comparados (la cantidad en que mejora) y *num_aleat* es un número aleatorio entre [0,1].

Si el valor de la función objetivo de la máquina actual ha empeorado (es decir, si ha aumentado su valor respecto al valor de función objetivo que presentaba en la anterior iteración), se recompensa negativamente la meta-regla que produjo el cambio disminuyendo su fuerza según la siguiente expresión:

$$fuerza = fuerza - (factor\_recompensa \times empeoramiento \times num\_aleat) \qquad (6.10)$$

donde *factor_recompensa* es un parámetro introducido por el usuario, *empeoramiento* es la diferencia existente entre los dos valores de función objetivo comparados (la cantidad en que empeora) y *num_aleat* es un número aleatorio entre [0,1].

Si el valor de la función objetivo de la máquina actual no ha cambiado (es decir, si su valor de función objetivo en la iteración actual es igual que el valor de función objetivo que tenía en la anterior iteración), se penaliza la meta-regla que se aplicó, disminuyendo su fuerza según la siguiente expresión:

$$fuerza = fuerza - (penalización \times num\_aleat) \tag{6.11}$$

donde *penalización* es un parámetro introducido por el usuario y *num_aleat* es un número aleatorio entre [0,1].

Se puede apreciar que las recompensas y penalizaciones tienen una componente aleatoria. El usuario puede controlar la suavidad o la agresividad de las recompensas (positivas y negativas) y de las penalizaciones, según los valores que introduzca en los parámetros *factor_recompensa* y *penalizacion*.

Además, en el proceso de recompensa se considera el resto de las meta-reglas cuyos índices se guardan en *M*. Estas meta-reglas son todas las meta-reglas que encajaron en la anterior iteración, pero que a diferencia de la meta-regla *num_meta*, no se llegaron a aplicar. Se seleccionan las meta-reglas de *M* que tienen codificadas las mismas acciones sobre los antecedentes y consecuentes de las reglas que la meta-regla *num_meta* aplicada. En el caso de que existan, se les aplica una recompensa positiva o negativa o se les penaliza, del mismo modo que se hace con la meta-regla *num_meta*.

Si el valor de función objetivo de la máquina en la iteración actual ha empeorado, se elimina dicha máquina y el algoritmo sigue trabajando en la iteración actual con la máquina de la iteración anterior que tenía mejor valor de función objetivo. El registro *registro_fitness* se actualiza, tomando como valor de función objetivo en la presente iteración el valor de función objetivo de la máquina de la anterior iteración.

Por último, se inicializa la matriz *M* a una matriz vacía, quedando así preparada para guardar los índices de las meta-reglas que encajan en la presente iteración, y que se buscarán más adelante, en una etapa posterior del algoritmo (paso 7).

Esta etapa es importante, ya que es la que realiza la evaluación de las meta-reglas. El valor de fuerza de las meta-reglas (cuarto campo) actúa como una medida de aptitud de las mismas. Este valor será tomado en cuenta en:

- El proceso de borrado (paso 6), para decidir qué meta-reglas se deben borrar (cuáles son las peores, es decir, cuáles tienen menor valor de fuerza).

- En la selección de la meta-regla a aplicar (paso 9), para decidir qué meta-regla se debe aplicar sobre la máquina (cuál es mejor, es decir, cuál tiene mayor fuerza).

- En el algoritmo genético (paso 12), en el proceso de selección de las mejores meta-reglas para realizar la repoblación (las que tienen mayor fuerza).

Paso 5: Proceso de depurado.

En este proceso se recorre toda la base de meta-reglas y se eliminan de la base aquellas meta-reglas cuya parte de acción sobre las reglas de la máquina esté toda codificada con "0" (*don't care*), por ser las que, en el caso de ser aplicadas sobre la regla de la máquina, no introducirían ningún cambio en su estructura.

La función que se utiliza acepta como entrada la variable que contiene la base de meta-reglas existente en la iteración actual y devuelve como salida la misma variable, pero con la base de meta-reglas depurada.

El proceso de depurado es necesario para evitar estancamientos en el algoritmo. Si las meta-reglas están compuestas solamente por símbolos *don't care*, no introducen ninguna acción beneficiosa sobre la máquina, y el aprendizaje no avanza en ninguna dirección, lo que conlleva un estancamiento de la búsqueda no deseado.

Paso 6: Proceso de borrado.

En este paso, se procederá a borrar N elementos (N es un parámetro introducido por el usuario). Por lo tanto, se llamará N veces a la función que busca una meta-regla de la base de meta-reglas que se debe borrar según los criterios que se expondrán a continuación.

La función que se utiliza para seleccionar una meta-regla a borrar acepta como entrada:

- La base de meta-reglas existente en la iteración actual.
- Un factor que se utiliza en la comparación con la media de fuerzas para realizar el borrado (parámetro *factor_borrado*)
- El límite de experiencia exigido y establecido como constante en la inicialización del algoritmo (parámetro *umbral_experiencia*).

Para elegir la meta-regla que se debe eliminar, se sigue un procedimiento basado en estudiar la fuerza y la experiencia de las meta-reglas que componen la base total de meta-reglas. En primer lugar, se asocia una puntuación o voto a cada meta-regla. En función de este valor, cada meta-regla tendrá una mayor o menor probabilidad de ser elegida para su eliminación. Se recorre toda la base de meta-reglas y se asigna una puntuación a cada una según el siguiente criterio:

Si (*experiencia*(*i*) > *umbral_experiencia*) & (*fuerza* (*i*) ≤*factor_borrado*× *sum_fuerzas*)
voto (*i*) = |(sum_fuerzas/fuerza(i))|
en otro caso,
voto (*i*) = *0* (6.12)

donde *experiencia*(*i*) es la experiencia de la meta-regla *i*, *umbral_experiencia* es un parámetro introducido por el usuario que indica la experiencia mínima que debe tener una meta-regla para ser borrada (número de veces que se debe haber sido aplicada antes), *fuerza*(*i*) es la fuerza de la meta-regla *i*, *sum_fuerzas* es la suma de las fuerzas de todas las meta-reglas que componen la base y *factor_borrado* es un parámetro introducido por el usuario, con valor menor que 1, que establece un porcentaje de fuerza sobre la fuerza total existente en la base.

Por lo tanto, una meta-regla que se haya aplicado varias veces (que tiene suficiente experiencia) y cuya fuerza sea menor que un límite determinado (relacionado con la fuerza total existente en la base de meta-reglas) es una meta-regla que no introduce acciones beneficiosas en la base de meta-reglas, y por tanto, recibirá una puntuación para ser eliminada mayor que una que no cumpla lo mismo.

A continuación, se hace un experimento de ruleta sobre las meta-reglas considerando la puntuación de cada una. Este experimento se implementa con un bucle de la siguiente manera:

punto = *voto_total* × *num_aleat*
acumulacion = *0*

Para cada elemento i de la base,

acumulacion = *acumulacion* + *voto*(*i*)
Si (*acumulacion* > *punto*)
eliminar = *i*
Fin del experimento

donde *voto_total* es la suma de los votos de todas las meta-reglas de la base, *num_aleat* es un número aleatorio entre [0,1], *voto*(*i*) es la puntuación de la meta-regla *i* y *eliminar* es el índice de la meta-regla que ha sido seleccionada para borrarse.

Si no hay ninguna meta-regla que cumpla estos requisitos, y la variable *eliminar* está vacía, se buscan las meta-reglas que tienen el mínimo valor de fuerza, y entre éstas, se selecciona la que tiene mayor valor de experiencia.

El proceso de borrado colabora en la renovación de la base de meta-reglas, eliminando meta-reglas que no contribuyen a desarrollar el aprendizaje correcto. Además, representa un modo de controlar la longitud total de la base de meta-reglas, factor muy importante en este algoritmo, ya que con bases muy largas el costo computacional del algoritmo aumenta haciéndolo más lento.

Paso 7: Búsqueda de las meta-reglas que encajan con los antecedentes de la máquina.

En esta etapa, el algoritmo recorre toda la base de meta-reglas y va comparando la matriz de antecedentes de la máquina finita de estados borrosa con la parte de comparación de cada meta-regla (primer campo), guardando en la variable *M* los índices de las meta-reglas que encajan. En esta comparación se tiene en cuenta que el orden de

las reglas no es relevante, es decir, que las dos matrices pueden encajar si sus filas encajan, aunque estas filas estén colocadas en distinto orden en cada matriz.

La función que se utiliza acepta como entrada la variable que contiene la base de meta-reglas existente en la iteración actual y la matriz que contiene los antecedentes de las reglas de la máquina finita de estados borrosa generada. Esta función devuelve como salida la variable *M*, que contiene los índices de las meta-reglas que encajan con la matriz de antecedentes.

Paso 8: Proceso de recubrimiento.

El usuario introduce un parámetro, *num_encajes*, que es el número mínimo de meta-reglas que deben encajar con la matriz de antecedentes. Si el número de meta-reglas de la base que han encajado en el paso anterior es igual o mayor que el valor de este parámetro, no se realiza recubrimiento. Pero si no hay suficientes meta-reglas que encajen, se llama a la función de recubrimiento, tantas veces como elementos falten en *M*. Esta rutina genera, cada vez que se la llama, una meta-regla que encaja con la matriz de antecedentes existente en la iteración actual.

Si el algoritmo se encuentra en la primera iteración, la base de meta-reglas está vacía, y por lo tanto, al llegar a esta etapa, la matriz *M* está también vacía. En este caso particular, no hay ninguna meta-regla que encaje con la matriz de antecedentes de la máquina porque simplemente las meta-reglas aún no se han creado. Entonces, se generará con este proceso de recubrimiento una base de meta-reglas inicial, con un número *num_encajes* de meta-reglas, que encajan con la matriz de antecedentes de la máquina. Por consiguiente, es esta rutina la que genera la base de meta-reglas inicialmente. Esta base irá cambiando de tamaño y sus meta-reglas irán cambiando por diversos factores: repoblación de la base con algoritmos genéticos (paso 12), etapas de depurado (paso 5), de borrado (paso 6) y nuevos procesos de recubrimiento.

La función que implementa el recubrimiento acepta como entrada:

- La matriz que contiene los antecedentes de las reglas de la máquina finita de estados borrosa.
- El valor de fuerza que se desea que tenga una meta-regla al ser creada por primera vez (parámetro *fuerza_inicial*).

- El valor de experiencia que se desea que tenga una meta-regla al crearse por primera vez (parámetro *experiencia_inicial*).
- Los conjuntos con los posibles valores de los parámetros asociados a las funciones de pertenencia $P_i = \left\{\mathbf{p}_1^i, \mathbf{p}_2^i, ..., \mathbf{p}_{N_p}^i\right\}$.
- La configuración para la generación de consecuentes (parámetro *num_puntos*).
- El número de iteración actual.
- La probabilidad de introducir símbolos *don´t care* en la parte de comparación (primer campo) de la meta-regla que se genera (parámetro *p_cov*).

Como ya se ha explicado previamente en la introducción, una meta-regla consta de 6 campos distintos: el **campo de comparación**, que es lo que se compara con la matriz de antecedentes de la máquina finita de estados borrosa para saber si hay encaje entre la máquina y la meta-regla, el **campo de acción sobre los antecedentes de la máquina**, que define los cambios a realizar sobre los antecedentes de una de las reglas de la máquina, el **campo de acción sobre los consecuentes de la máquina**, que define los cambios a realizar sobre los consecuentes de una de las reglas de la máquina, así como el número de la regla de la máquina sobre la que se aplica la meta-regla, el **campo de fuerza**, que indica el valor de fuerza de la meta-regla, el **campo de experiencia**, que indica el valor de experiencia de la meta-regla y el **campo de antigüedad**, que guarda el número de la iteración en la que la meta-regla se crea (otros pasos del algoritmo usarán el valor guardado en este campo para calcular la antigüedad de la meta-regla).

El primer campo, campo de comparación, se genera copiando la matriz de antecedentes de la máquina. Luego, se recorre elemento a elemento, y sobre cada uno se realiza un experimento con probabilidad *p_cov* (parámetro introducido por el usuario) que decide si se deja el elemento tal y como está o se sustituye por un "0". Este símbolo es un *don't care*.

Cuantos más símbolos *don't care* tenga la parte de comparación de la meta-regla, más posibilidades tiene de encajar no sólo con la matriz de antecedentes a partir de la que ha sido generada, sino con otras distintas, en otras iteraciones del algoritmo. Es decir, el número de símbolos *don't care* influye directamente en la capacidad de generalización de la meta-regla. Si se generan pocos símbolos *don't care* en este campo, las meta-reglas serán menos generales y encajarán con menos máquinas. Esto obligará al proceso de recubrimiento a generar más meta-reglas por iteración capaces de encajar

con una máquina dada. Como ventaja, este hecho hará que la base de meta-reglas contenga meta-reglas más variadas, pero como desventaja, la base de meta-reglas será más grande, lo que ralentiza el algoritmo.

El campo de una meta-regla de acción sobre los antecedentes de la máquina es un vector que se genera eligiendo aleatoriamente los códigos posibles de las funciones de pertenencia asociadas a la señal externa y a cada uno de los estados de la máquina para la parte de antecedentes de la misma. En esta elección se incluye la posibilidad de incluir un símbolo más en la codificación, el "0", que es un *don't care*.

El campo de una meta-regla de acción sobre los consecuentes de la máquina es un vector que se genera eligiendo aleatoriamente los códigos posibles para los consecuentes de la máquina. En esta elección se incluye la posibilidad de incluir un símbolo más en la codificación, el "0", que es un *don't care*.

El número de símbolos *don't care* en estos campos de acción (sobre antecedentes y sobre consecuentes de la máquina) influye directamente en el tipo de cambio que sufre la máquina al aplicarse sobre ella una meta-regla. Cuantos más símbolos *don't care* existan en estos campos, más suave será el cambio que se realiza en la máquina de iteración a iteración, y por lo tanto, la búsqueda de la máquina se realizará de un modo más progresivo. Con pocos símbolos *don't care* en estos campos, el cambio que se realiza en la máquina de iteración a iteración al aplicar una meta-regla sobre ella será más brusco, dando lugar a una búsqueda menos progresiva.

En el campo de fuerza de la meta-regla recién creada se introduce un valor de fuerza inicial, introducido por el usuario en el parámetro *fuerza_inicial*. Este campo será modificado en la evolución del algoritmo según sean beneficiosos los cambios introducidos por la meta-regla sobre la máquina (es el valor que se recompensa positivamente o negativamente o se penaliza en el paso 4).

En el campo de experiencia de la meta-regla recién creada se introduce un valor de experiencia inicial, introducido por el usuario en el parámetro *experiencia_inicial*. Este campo será modificado en la evolución del algoritmo dependiendo del número de veces que se aplique la meta-regla. Normalmente, el parámetro *experiencia_inicial* vale 0 (cuando la meta-regla se crea por primera vez nunca se ha aplicado sobre la máquina finita de estados borrosa).

En el campo de antigüedad de la meta-regla recién creada se guarda la iteración en la que la meta-regla ha sido creada. En etapas posteriores (etapa 11), se calculará la

antigüedad que tiene la meta-regla como la diferencia entre el número de iteración contenido en este campo y la iteración actual en la que se considera.

Todos estos campos se guardan en la variable que guarda la base de meta-reglas existente en cada iteración.

Paso 9: Selección de la meta-regla a aplicar sobre la máquina finita de estados borrosa.

En este paso del algoritmo, elegimos una meta-regla de entre todas las que encajan con la matriz de antecedentes de la máquina (cuyos índices están en *M*) para aplicarla sobre la máquina.

La función que se realiza esta selección acepta como entrada la variable que contiene la base de meta-reglas existente en la iteración actual y la variable *M*, que contiene los índices de las meta-reglas que encajan con la matriz de antecedentes. Esta función devuelve como salida el índice de la meta-regla que se selecciona para aplicarse sobre la máquina finita de estados borrosa (*num_meta*).

Para elegir la meta-regla, hacemos un experimento con una probabilidad del 50% de elegir esta meta-regla al azar entre todas las posibles (método de exploración pura) o de elegirla haciendo un experimento de ruleta sobre las fuerzas de las meta-reglas candidatas (método de explotación pura). En la literatura relacionada con los métodos de aprendizaje con refuerzo, la combinación de estos dos métodos recibe el nombre de selección ε - *greedy* [Sutton y Barto, 1998].

En el segundo caso, este experimento de ruleta se implementa con un bucle del siguiente modo:

punto = *sum_fuerzas* × *num_aleat*
acumulacion = *0*
Para cada elemento i de M,
    acumulacion = *acumulacion* + *fuerza* (*i*)
    Si (*acumulacion* > *punto*)
        num_meta = *i*
        Fin del experimento

donde *sum_fuerzas* es la suma de las fuerzas de todas las meta-reglas de *M*, *num_aleat* es un número aleatorio entre [0,1], *fuerza* (*i*) es el valor de fuerza de la meta-regla *i*, y *num_meta* es la variable donde se guarda el índice de la meta-regla que se selecciona para ser aplicada sobre la máquina.

Paso 10: Aplicación de la meta-regla sobre la máquina de estados borrosa.
En esta etapa del algoritmo se aplica la meta-regla elegida en el paso anterior sobre la regla de la máquina finita de estados borrosa codificada en la misma meta-regla, en su segundo campo. El modo en que la meta-regla cambia la máquina es el descrito en la sección 6.3.3.2.

La función que se utiliza acepta como entrada:

- La base de meta-reglas existente en la iteración actual.
- El índice de la meta-regla que se selecciona para ser aplicada sobre la máquina finita de estados borrosa (*num_meta*).
- La matriz que contiene los antecedentes de las reglas de la máquina finita de estados borrosa.
- La matriz que contiene los consecuentes de las reglas de la máquina finita de estados borrosa.

Esta función devuelve como salida la matriz que contiene los nuevos antecedentes de las reglas y la matriz que contiene los nuevos consecuentes de las reglas de la máquina finita de estados borrosa, tras las modificaciones introducidas por la meta-regla aplicada.

Paso 11: Proceso de arranque del algoritmo genético.
En esta etapa se analizan las meta-reglas que encajan en esta iteración (cuyos índices están guardados en *M*) y se decide si es necesario arrancar un algoritmo genético sobre las meta-reglas de *M* en la iteración actual.

La función que toma esta decisión acepta como entrada:

- La variable que contiene la base de meta-reglas existente en la iteración actual.

- La variable *M*, que contiene los índices de las meta-reglas que encajan con la matriz de antecedentes.
- La antigüedad requerida en las meta-reglas que encajan para disparar el algoritmo genético (parámetro *umbral_antigüedad*).
- El número de la iteración actual.
- El porcentaje de elementos de *M* cuya antigüedad debe superar el umbral de antigüedad *umbral_antiguedad* para que se dispare un algoritmo genético (parámetro *porción*).
- El número mínimo de iteraciones que deben pasar desde el último algoritmo genético que se ha disparado antes de disparar uno nuevo (parámetro *min_iter*).
- El registro *iter*, donde se guardan los números de las iteraciones en las que se ha ejecutado un algoritmo genético.

Esta función devuelve como salida una bandera (*band*) que indica si se debe disparar el algoritmo genético en la iteración actual.

La toma de la decisión sobre cuándo se debe disparar un algoritmo genético se ha implementado de dos modos distintos. En algunas pruebas se ha seguido un método y en otras, otro. A continuación, se exponen las dos opciones.

En el primer método, la función toma la decisión dependiendo de la antigüedad de las meta-reglas que hay en *M*. Se calculan los valores de antigüedad de las reglas, haciendo la diferencia entre la iteración actual *k* y la iteración en que fueron creadas (sexto campo de la meta-regla, ver etapa 8). A continuación, se buscan todas las meta-reglas de *M* cuyos valores de antigüedad superen el umbral *umbral_antiguedad* introducido como parámetro por el usuario y se da la orden de disparo sólo si hay un porcentaje mínimo de elementos cuya antigüedad supere ese umbral (este porcentaje lo da el parámetro *porcion*, también introducido por el usuario). De este modo, se controla que el algoritmo genético no se dispare si las meta-reglas no son lo suficientemente antiguas y no han tenido suficientes oportunidades de actuar sobre la máquina.

En el segundo método se dispara el algoritmo genético con una frecuencia constante, según indique el parámetro *min_iter*.

Paso 12: Ejecución de un algoritmo genético sobre las meta-reglas de *M*.

Si la bandera *band* tiene valor 1, entonces se dispara un algoritmo genético sobre las meta-reglas cuyos índices están guardados en la variable *M*.

La función que se realiza este algoritmo genético acepta como entrada:

- La base de meta-reglas existente en la iteración actual.
- El conjunto *M* de las meta-reglas que encajan con la matriz de antecedentes.
- El porcentaje de los mejores elementos de la base de meta-reglas que se seleccionarán para llevar a cabo la repoblación (parámetro *alfa*), estableciendo así el solape entre poblaciones.
- Las probabilidades para la elección de uno de los operadores genéticos siguientes: reproducción, mutación y cruce (parámetros *p1, p2, p3*).
- La probabilidad de realizar una mutación sobre un elemento de una meta-regla (parámetro *p_mut*).
- La probabilidad de que la mutación de un elemento de la parte de comparación de una meta-regla lleve necesariamente a un comodín (parámetro *p_mut2*).
- Los posibles valores de los parámetros asociados a las funciones de pertenencia.
- El número de puntos deseado para generar los consecuentes de las reglas de la máquina finita de estados borrosa (parámetro *num_puntos*).
- El valor de experiencia que se desea que tenga una meta-regla al crearse por primera vez (parámetro *experiencia_inicial*)
- El número de reglas que tiene la máquina finita de estados borrosa (parámetro *num_reglas*).
- El número de la iteración actual.

Esta función devuelve como salida la variable que contiene la nueva base de meta-reglas generada.

El algoritmo genético comienza con un proceso de **selección**, en el que se eligen las mejores meta-reglas que hay en *M*. Una meta-regla es mejor que otra si su fuerza es mayor. Si la meta-regla se ha aplicado sobre la máquina y la ha mejorado, habrá sido recompensada y su valor de fuerza será mayor que otras meta-reglas que no hayan introducidos cambios beneficiosos en la máquina, y que por tanto, hayan sido recompensadas negativamente o penalizadas disminuyendo su fuerza. Nos interesa

repoblar utilizando como "padres" las mejores meta-reglas. Este proceso de selección analiza el cuarto campo de todas las meta-reglas de M (campo que contiene el valor de la fuerza de la meta-regla), ordena las meta-reglas de mejor a peor (de mayores a menores valores de fuerza) y elige un porcentaje de las mejores meta-reglas existentes, según el parámetro *alfa* introducido por el usuario. Los campos de las meta-reglas seleccionadas se guardan en variables distintas durante el resto del proceso de repoblación.

A continuación, se comienza con la **repoblación**. Se generan nuevas meta-reglas "hijas" de estas meta-reglas "padres" que sustituirán a las meta-reglas de *M* que no han sido seleccionadas para repoblar, es decir, se generan tantas meta-reglas nuevas como meta-reglas antiguas hayan sido rechazadas por ser peores.

Para generar las nuevas meta-reglas, se aplicarán tres **operadores genéticos**: reproducción, mutación y cruce. Se hace un experimento para elegir qué operador se aplica cada vez con probabilidades *p1, p2* y *p3* para la reproducción, la mutación y el cruce respectivamente. Estas probabilidades de selección de cada operador genético son parámetros de algoritmo. Cada operador genético, si es seleccionado, funciona de la siguiente manera:

1. Reproducción: Se elige aleatoriamente una meta-regla de las seleccionadas para repoblar y se genera una meta-regla "hija" que es una copia exacta de la meta-regla "padre". Ambas meta-reglas se introducirán posteriormente en la nueva población.

En el sexto campo de la meta-regla "hija" se guarda el número de iteración actual, por ser la iteración en la que esta meta-regla se genera por primera vez.

2. Mutación: Se elige aleatoriamente una meta-regla de las seleccionadas para repoblar y se genera una meta-regla "hija" que es el resultado de mutar la meta-regla "padre". Ambas meta-reglas se introducirán posteriormente en la nueva población.

La mutación actúa sobre los tres primeros campos de la meta-regla "padre" (parte de comparación, parte de acción sobre los antecedentes de la máquina y parte de acción sobre los consecuentes de la máquina). Se van recorriendo elemento a elemento los campos, y sobre cada uno de ellos se realiza un experimento aleatorio en el que se mutará el correspondiente elemento con una probabilidad *p_mut* (parámetro establecido a priori).

Cuando hay que mutar un elemento de la parte de comparación de la meta-regla, se realiza un nuevo experimento en el que se muta el elemento con una probabilidad *p_mut2* (parámetro establecido en la configuración). Si el experimento aleatorio resulta negativo, se procede de la siguiente manera. Si el elemento es un símbolo *don't care*, "0", se cambia por cualquier otro símbolo distinto de los permitidos. Si el elemento no era un *don`t care*, se cambia por un "0" (*don´t care*). Si el experimento resulta positivo, se cambia el elemento por un *don't care*

Cuando hay que mutar un elemento de la parte de acción sobre los antecedentes o de la parte de acción sobre los consecuentes de la meta-regla, se elige aleatoriamente cualquier valor de los permitidos.

El valor de la fuerza del nuevo individuo (meta-regla "hija") es una copia de la fuerza de la meta-regla "padre". A la experiencia de la meta-regla "hija" se le da el valor del parámetro *experiencia_inicial*, y en el campo de antigüedad se guarda el número de iteración actual, por ser la iteración en la que esta meta-regla se genera por primera vez.

El operador mutación requiere:

- La cardinalidad de los conjuntos de parámetros asociados a las funciones de pertenencia.
- El número de puntos equidistantes en el intervalo [0,1] y que se utilizan para (parámetro *num_puntos*).
- La meta-regla seleccionada para repoblar por mutación que hará de "padre".
- El número de la iteración actual.
- El número de reglas que tiene la máquina finita de estados borrosa (parámetro *num*_reglas).
- Valor de experiencia que se desea que tenga una meta-regla al crearse por primera vez (parámetro *experiencia_inicial*).

3. Cruce: Se eligen aleatoriamente dos meta-reglas de las seleccionadas para repoblar y se generan dos meta-reglas "hijas" que son el resultado de cruzar las meta-reglas "padres". Las cuatro meta-reglas se introducirán posteriormente en la nueva población.

El cruce actúa sobre los tres primeros campos de las meta-reglas "padre" (parte de comparación, parte de acción sobre los antecedentes de la máquina y parte de acción sobre los consecuentes de la máquina). Para ello, la parte de comparación, que originalmente es una matriz, se transforma en un vector fila con todos los elementos de la matriz ordenados por orden de filas. A este vector se le añaden los vectores de elementos de los otros dos campos. Se construye este vector para cada una de las meta-reglas a cruzar.

Para cruzar las meta-reglas "padre", se eligen al azar dos posiciones del vector distintas (dos puntos de corte distintos) y se intercambian los elementos de ambos vectores correspondientes a las posiciones intermedias entre estos dos puntos de corte. Por ejemplo, supongamos que se desea cruzar las meta-reglas que aparecen en la tabla 6.9.

Meta-regla 1					**Meta-regla 2**				
Campo de comparación					Campo de comparación				
entrada externa	estado 1	estado 2	estado 3	estado 4	entrada externa	estado 1	estado 2	estado 3	estado 4
2	4	0	1	3	0	0	0	0	3
0	0	2	2	3	0	3	0	5	4
5	0	4	2	0	5	1	0	2	0
1	3	0	4	5	0	0	5	4	1
0	0	0	3	3	4	2	2	0	4
1	0	3	0	0	3	0	0	0	1
1	5	0	1	0	0	3	0	0	5
0	1	3	0	3	4	0	3	5	0
0	1	0	0	3	1	2	0	0	0
4	0	5	1	3	5	0	4	0	2
Campo de acción sobre los antecedentes					Campo de acción sobre los antecedentes				
3	5	0	1	4	0	0	2	5	6
Campo de acción sobre los consecuentes					Campo de acción sobre los consecuentes				
0	3	0	1	1	1	2	5	0	3

Tabla 6.9. Ejemplo de dos meta-reglas "padres" seleccionadas para realizar una operación de cruce.

Colocamos cada una de las meta-reglas en un vector, tal y como se indica en la tabla 6.10. Se eligen dos puntos de cruce distintos aleatoriamente. Por ejemplo, supongamos que estos puntos corresponden a las posiciones 17 y 41. Las dos meta-reglas "hijas" serán el resultado de cruzar el contenido de los vectores entre estas dos posiciones, tal y como se indica en la tabla 6.11.

Meta-regla 1 en forma de vector															
Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2	4	0	1	3	0	0	2	2	3	5	0	4	2	0
Posición	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	1	3	0	4	5	0	0	0	3	3	1	0	3	0	0
Posición	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	1	5	0	1	0	0	1	3	0	3	0	1	0	0	3
Posición	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
	4	0	5	1	3	3	5	0	1	4	0	3	0	1	1
Meta-regla 2 en forma de vector															
Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	3	0	3	0	5	4	5	1	0	2	0
Posición	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	0	0	5	4	1	4	2	2	0	4	3	0	0	0	1
Posición	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	0	3	0	0	5	4	0	3	5	0	1	2	0	0	0
Posición	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
	5	0	4	0	2	0	0	2	5	6	1	2	5	0	3

Tabla 6.10. Meta-reglas "padres" seleccionadas para realizar el cruce, colocadas como vectores.

Primera meta-regla "hija" en forma de vector															
Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2	4	0	1	3	0	0	2	2	3	5	0	4	2	0
Posición	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	1	0	5	4	1	4	2	2	0	4	3	0	0	0	1
Posición	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	0	3	0	0	5	4	0	3	5	0	1	1	0	0	3
Posición	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
	4	0	5	1	3	3	5	0	1	4	0	3	0	1	1
Segunda meta-regla "hija" en forma de vector															
Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	3	0	3	0	5	4	5	1	0	2	0
Posición	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	0	3	0	4	5	0	0	0	3	3	1	0	3	0	0
Posición	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	1	5	0	1	0	0	1	3	0	3	0	2	0	0	0
Posición	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
	5	0	4	0	2	0	0	2	5	6	1	2	5	0	3

Tabla 6.11. Meta-reglas "hijas" resultantes de la operación de cruce, colocadas como vectores.

El valor de la fuerza de los nuevos individuos (meta-reglas "hijas") es la media de las fuerzas de las meta-reglas "padres". A las experiencias de las meta-reglas "hijas" se les da el valor del parámetro *experiencia_inicial*, y en sus campo de antigüedad se

guarda el número de iteración actual, por ser la iteración en la que estas meta-reglas se generan por primera vez.

La función que implementa el cruce acepta como entrada:

- Las meta-reglas seleccionadas para repoblar por cruce que harán de “padres”.
- El número de la iteración actual.
- Valor de experiencia que se desea que tenga una meta-regla al crearse por primera vez (parámetro *experiencia_inicial*).

Una vez generadas todas las meta-reglas necesarias para completar la longitud total de *M*, se añaden a la base total de meta-reglas.

Lo último que se realiza en la iteración es la actualización de todos los registros utilizados en el algoritmo: el registro en el que se guardan los valores de las fuerzas de las meta-reglas que componen la base de meta-reglas en cada iteración (*registro_fuerza*), el registro en el que se guardan los valores de las experiencias de las meta-reglas que componen la base de meta-reglas en cada iteración (*registro_experiencia*) y el registro en el que se guardan los valores de las antigüedades de las meta-reglas que componen la base de meta-reglas en cada iteración (*registro_antigüedad*).

La siguiente iteración del algoritmo comienza en la etapa 2. El algoritmo se detiene, si encuentra una máquina que cumple las especificaciones del usuario, en el paso 3. En la figura 6.5. se muestra la representación esquemática del algoritmo genético utilizado en la arquitectura Michigan en algunas iteraciones.

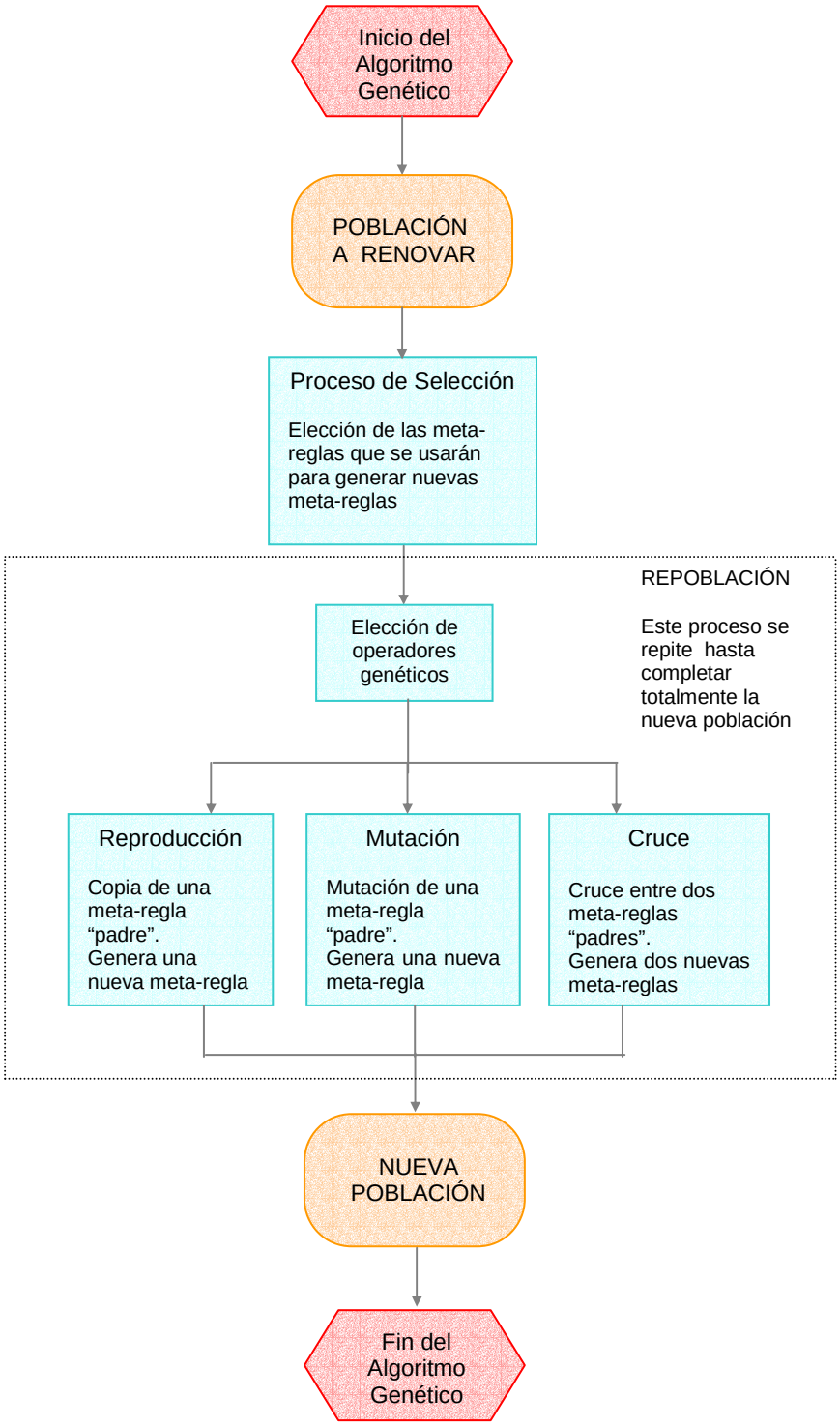


Figura 6.5. Representación esquemática del algoritmo genético

6.3.4 Proceso de validación.

Una vez obtenida la máquina finita de estados borrosa óptima y los centros del clustering, se procede a la validación de este conjunto como clasificador de patrones. El modo de validar el clasificador final es el mismo, independientemente del sistema evolutivo seguido para encontrarlo (sistema tipo Pittsburgh o sistema tipo Michigan).

Realizamos una prueba o test de la eficiencia de este conjunto como clasificador en el problema de clasificación concreto. Para ello, evaluamos la máquina con un conjunto de test formado por trazas diferentes a las del conjunto de entrenamiento, pero pertenecientes también al mismo dominio. Se analiza la matriz de activaciones de los estados para todas las trazas, y se calculan los parámetros de reactividad, contabilizando las veces que el estado de detección supera el umbral de detección y dividiendo este valor por el número total de activaciones del estado. A continuación, se calcula la pertenencia del parámetro de reactividad calculado a cada una de las clases. Para ello, se utiliza un método de interpolación lineal que toma como referencia los pares (valor del parámetro de reactividad, pertenencia a la clase) obtenidos del conjunto de entrenamiento, y como variable independiente el parámetro de reactividad calculado. Una vez estimada la pertenencia, se asigna la traza a la clase cuyo valor de pertenencia es superior.

Con este procedimiento obtenemos la clasificación del conjunto de test y el error cometido en esta clasificación.

Este procedimiento se ha extendido y aplicado en lo que se denomina validación cruzada, que consiste en dividir en distintos conjuntos las trazas disponibles del dominio (más de dos conjuntos), y hacer entrenamientos con uno de los conjuntos y validaciones con los restantes, intercambiando los papeles del “conjunto de entrenamiento” y “conjuntos de test” en cada prueba. El objetivo final es obtener una media del error cometido por el clasificador en cada problema, para analizar la capacidad de generalización del sistema y su dependencia con el conjunto de entrenamiento utilizado.

Capítulo 7
Validación de los algoritmos con datos simulados.

7.1 Introducción.

En este capítulo se detallan los resultados obtenidos con los algoritmos descritos en el capítulo anterior (sistemas Pittsburgh y Michigan) en la clasificación de series temporales obtenidas mediante simulación de modelos ocultos de Markov. El objetivo principal es evaluar la calidad de los algoritmos planteados en cuanto a su capacidad de aprendizaje a partir de los datos de entrenamiento del modelo subyacente a los datos. El modelo oculto de Markov es adecuado como referencia para estos algoritmos, puesto que el comportamiento futuro del sistema sólo depende del último estado alcanzado, al igual que en el modelo borroso recurrente sobre el que se investiga en esta tesis. Además, existen algoritmos bien establecidos desde hace tiempo, que permiten resolver los problemas asociados a la identificación de los modelos subyacentes y la clasificación de la serie temporal en forma supervisada. De esta manera, se dispone de un sistema de referencia con el que realizar valoraciones comparativas.

Se investigarán parámetros importantes de los algoritmos, con el propósito de establecer su grado de influencia en el comportamiento del sistema. En el caso del sistema tipo Pittsburgh, se evaluará el papel del solapamiento de las poblaciones de individuos de generación a generación y la relevancia de los diferentes operadores

genéticos. Por otra parte, en el sistema tipo Michigan se evaluarán dos métodos que permite variar la frecuencia de disparo del algoritmo genético, para determinar cuál es la verdadera importancia de su participación como sistema de descubrimiento en el algoritmo.

Se comenzará el capítulo con una revisión escueta de los modelos ocultos de Markov, ya que serán el modelo de referencia en este estudio. A continuación, se establecerán los dos modelos usados para los experimentos: se trata de dos modelos simples con dos estados cada uno, que difieren únicamente en la matriz de probabilidades de transición. Se proseguirá con la descripción de los experimentos realizados sobre los sistema Pittsburgh y Michigan. Finalmente, se hará un resumen de las conclusiones obtenidas.

7.2 Los modelos ocultos de Markov.

7.2.1 Introducción.

Un modelo de Markov finito (en inglés, *finite Markov model* – MM) se utiliza para modelar cierto tipo de procesos estocásticos.

Supongamos que tenemos un conjunto finito de estados, $Q = \{q_1, ..., q_N\}$, con cardinalidad N, y que el proceso que se trata de modelar está exactamente en un estado de Q en todo momento. El proceso cumple la "propiedad de Markov", según la cual la historia que ha dado lugar al estado actual es irrelevante para el comportamiento futuro del proceso. Lo único que influye en el comportamiento futuro es el estado actual. En cada momento, el proceso pasa al siguiente estado (que podría ser el mismo estado que el estado previo) basándose en una distribución de probabilidad de transición, $a_{i,j}$, que depende sólo del estado previo. La notación $a_{i,j}$ representa la probabilidad de que el proceso que se encuentra en el estado i pase al siguiente estado j.

Un MM cumple la propiedad de que la probabilidad de que ocurra una transición de un estado dado a algún estado siguiente es 1:

$$\forall i \in Q \qquad \sum_{j \in Q} a_{i,j} = 1 \tag{7.1}$$

Un modelo de Markov oculto finito (en inglés, *finite hidden Markov model* – HMM) es similar al MM, pero en los HMM la información de los estados está oculta para el observador. Los HMM también reciben el nombre de "procesos de Markov parcialmente observables". En ellos existe un segundo proceso estocástico el cual produce una observación en cada estado. Podemos encontrar una descripción de estos modelos en la referencia clásica [Rabiner, 1989]. Estos sistemas evolucionan en el tiempo pasando aleatoriamente de estado a estado y emitiendo en cada momento al azar algún símbolo del alfabeto Σ. Solamente los símbolos emitidos en un estado son observables, pero no la ruta o secuencia de estados.

Por lo tanto, en un HMM existe un conjunto de observaciones posibles, $V = \{v_1,...v_M\}$, de cardinalidad M y una distribución de probabilidad $b_i(k)$ de producir una observación dada en cada estado. La notación $b_i(k)$ representa la probabilidad de observar v_k cuando el proceso está en el estado i.

Un HMM cumple la misma propiedad que un MM, y por lo tanto, la probabilidad de producir alguna observación en cada estado es 1:

$$\forall i \in Q \qquad \sum_{k \in V} b_i(k) = 1 \tag{7.2}$$

Un MM es un caso particular de un HMM donde $V = Q$, es decir, donde la observación producida en cada estado es siempre el estado en sí mismo.

$$\forall i \in Q \qquad Q \cdot b_i(i) = 1 \tag{7.3}$$

Para completar el modelo, se necesita la probabilidad inicial de los estados, π. La notación π_i representa la probabilidad de que un proceso comience en el estado i:

$$\sum_{i \in Q} \pi_i = 1 \tag{7.4}$$

Por lo tanto, nos referiremos con el símbolo λ a un modelo HMM particular, es decir a un conjunto $<Q,V,a,b, \pi>$. Denotaremos una secuencia de observaciones específica como $O = <O_1,..., O_T>$, donde T es la longitud de la secuencia de la

observación y cada $O_t \in V$. La secuencia de estados que produce O la denotamos como $I = <I_1,...,I_T>$.

Estos sistemas evolucionan en el tiempo pasando aleatoriamente de estado a estado y emitiendo en cada momento al azar algún símbolo del alfabeto S. Cuando se encuentra en el estado $q_{t-1} = i$, tiene la probabilidad $a_{i,j}$ de moverse al estado $q_t = j$ en el siguiente instante y la probabilidad $b_j(k)$ de emitir el símbolo $o_t = v_k$ en el tiempo t. Solamente los símbolos emitidos por el proceso son observables, pero no la ruta o secuencia de estados, de ahí el calificativo de "oculto" de Markov, ya que el proceso de Markov es no observado.

En resumen, un MM es una máquina de estado finita que cambia de estado una vez en cada unidad de tiempo y que genera un dato o_t observable cada vez que está en un estado según una función de densidad de probabilidad $b_j(o_t)$. La manera en que ocurren las transiciones entre estados en el modelo es también probabilística y viene gobernada por una matriz de transición entre estados. Si en un MM la secuencia de estados que produce la secuencia de observaciones no se produce de forma determinista, entonces, el MM se llama HMM. Por lo tanto, un HMM es un proceso estocástico doble, ya que en él existe un proceso estocástico subyacente que no puede ser directamente observado, pero que puede ser estudiado a través de otros conjuntos de procesos estocásticos responsables de las secuencias de observaciones.

7.2.2 Problemas en los HMM.

Debido a la forma general y a los elementos de un HMM descritos en la sección anterior, se puede apreciar que para poder aplicar este modelo de un modo útil es necesario solucionar previamente tres problemas. Estos problemas son los siguientes:

- Dada una secuencia de observaciones $O = (o_1,...,o_T)$ y un modelo $\lambda = (a,b,\pi)$, es importante estudiar cómo se puede calcular eficientemente $P(O|\ \lambda)$, donde este término representa la probabilidad de que se genere una secuencia de observaciones determinada dado un modelo. Este es el denominado *problema de evaluación*, que consiste en estimar la probabilidad de que dado un modelo, éste haya producido un vector de datos particular.

- Dada una secuencia de observaciones $O = (o_1,...,o_T)$ y un modelo λ, se puede tratar de elegir una secuencia de estados correspondiente $I = (I_1,...,I_T)$ que sea óptima en algún sentido. Es decir, dados un vector de datos y un modelo, se debe determinar la secuencia de estados ocultos que maximiza la probabilidad de que el modelo haya producido los datos. Este problema es el *problema del descubrimiento*.

- El tercer problema que hay que abordar es el ajuste de los parámetros del modelo λ para maximizar $P(O|\lambda)$. Este es el *problema del entrenamiento* y es el más complicado de los tres. Dicho de otro modo, dada una secuencia de datos observados, se debe buscar el modelo que maximice la probabilidad de haber originado esos datos. Solucionar este problema es muy importante para la mayoría de las aplicaciones ya que permite adaptar los parámetros del modelo de un modo óptimo a los datos observados y crear los mejores modelos para abordar el problema del reconocimiento de patrones en datos reales.

  A continuación, los estudiamos detalladamente.

7.2.2.1 El problema de la evaluación.

Como se ha descrito, consiste en estimar la probabilidad de que dado un modelo determinado éste haya producido unos datos determinados. Un método para solucionar este problema puede basarse en la aplicación del siguiente razonamiento sobre todas las posibles secuencias de estados I de longitud T:

$$P(O \mid \lambda) = \sum_I P(O \mid I, \lambda) \times P(I \mid \lambda) \tag{7.5}$$

La probabilidad de una secuencia de estados determinada es :

$$P(I \mid \lambda) = \pi_{I1} \times a_{I1,I2} \times ... \times a_{IT-1,IT} \tag{7.6}$$

La probabilidad de que ocurra una secuencia de observaciones dada una secuencia de estados es:

$$P(O \mid I, \lambda) = b_{I1}(O_1) \times ... \times b_{IT}(O_T) \tag{7.7}$$

Pero en este planteamiento, el número de secuencias de estados es exponencial, por lo tanto el método es pobre. Por ejemplo, para un modelo con 5 estados y 100 observaciones, serían necesarias 10^{72} operaciones para encontrar la probabilidad de la secuencia observada [Rabiner y Juang, 1986]. Por lo tanto, se suele utilizar un método más eficiente, que se detalla a continuación.

La aproximación estándar es el denominado procedimiento hacia delante – hacia atrás (en inglés, *forward-backward procedure*), que aprovecha la propiedad de Markov de que la ruta seguida para llegar al estado actual es irrelevante en el futuro comportamiento del proceso. Primero se define:

$$\alpha_t(j) = P(O_1,...,O_t, I_t = j \mid \lambda) \tag{7.8}$$

que es la probabilidad de que el proceso produzca las primeras *t* observaciones y termine en el estado *j* en el instante *t*. La probabilidad $P(O|\lambda)$ se calcula fácilmente de α_T.

Se aplica la regla de la cadena para obtener la siguiente relación:

$$\alpha_t(j) = P(I_t = j \mid \lambda) \times P(O_1,...,O_t \mid I_t = j, \lambda) \tag{7.9}$$

Según la ley de la probabilidad total, se puede expresar:

$$P(O_1,...,O_t \mid \lambda) = \sum_{j \in Q} \left[P(O_1,...,O_t \mid I_t = j, \lambda) \times P(I_t = j \mid \lambda) \right] = \sum_{j \in Q} \alpha_t(j) \tag{7.10}$$

De lo que se deduce, cuando la observación entera es considerada, que:

$$P(O \mid \lambda) = \sum_{j \in Q} \alpha_T(j) \tag{7.11}$$

La probabilidad α_T se calcula iterativamente del siguiente modo:

$$1.\quad \alpha_1(i) = \pi_i \times b_i(O_1) \tag{7.12}$$

$$2.\quad \alpha_{t+1}(i) = \left[\sum_{j\in Q} \alpha_t(j) \times a_{j,i}\right] \times b_i(O_{t+1}) \tag{7.13}$$

En este planteamiento, el cálculo de $P(O|\lambda)$ usando α_T requiere $N \times T$ operaciones en vez del número exponencial que se requería en el primer método descrito. En la figura 7.1 se representa la secuencia de operaciones necesarias para el cómputo de la variable $\alpha_{t+1}(j)$.

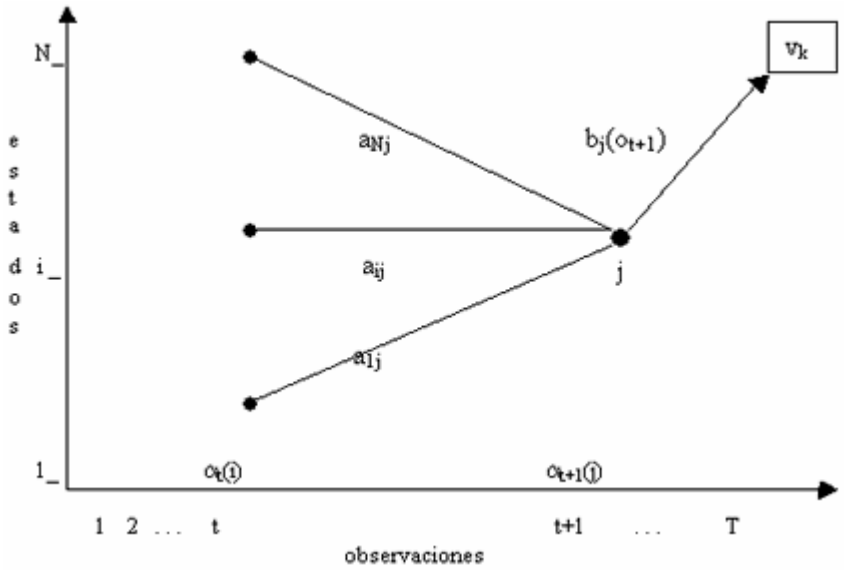


Figura 7.1. Secuencia de operaciones necesarias para el cómputo de la variable $\alpha_{t+1}(j)$.

La probabilidad α_t representa la parte "hacia delante", mientras que β_t representa la parte "hacia detrás". Definimos β_t como:

$$\beta_t(j) = P(O_{t+1}, O_{t+2}, ..., O_T \mid I_t = j, \lambda) \tag{7.14}$$

que es la probabilidad de que el proceso comience en el estado *j* en el instante *t* y que produzca las observaciones desde el instante *t+1* hasta *T*.

También podemos usar β para determinar $P(O|\lambda)$:

$$P(O_1, ..., O_T \mid \lambda) = P(O_1 \mid \lambda) \times P(O_2, ..., O_T \mid O_1, \lambda) \tag{7.15}$$

Primero se calcula $P(O_1|\lambda)$:

$$P(O_1 \mid \lambda) = \sum_{j \in Q} \left[P(O_1 \mid I_1 = j, \lambda) \times P(I_1 = j \mid \lambda) \right] = \sum_{j \in Q} \left[b_j(O_1) \times \pi_j \right] \qquad (7.16)$$

Luego se calcula $P(O_2,...,O_T \mid O_1, \lambda)$:

$$P(O_2,...,O_T \mid O_1, \lambda) = \sum_{j \in Q} \left[P(O_2,...,O_T \mid O_1, I_1 = j, \lambda) \times P(I_1 = j \mid \lambda) \right] \qquad (7.17)$$

Pero una vez fijada la condición $I_1 = j$, se pueden ignorar las condiciones de O_1 porque del conocimiento de la observación producida en el instante 1 no podemos obtener más información una vez que sabemos el estado en el instante 1. Por lo tanto, podemos simplificar:

$$\begin{aligned} P(O_2,...,O_T \mid O_1, \lambda) &= \sum_{j \in Q} \left[P(O_2,...,O_T \mid I_1 = j, \lambda) \times P(I_1 = j \mid \lambda) \right] = \\ &= \sum_{j \in Q} \left[\beta_1(j) \times \pi_j \right] \end{aligned} \qquad (7.18)$$

Agrupando las expresiones 7.16 y 7.18 se obtiene:

$$P(O_1,...,O_T \mid \lambda) = \sum_{j \in Q} \left[b_j(O_1) \times \pi_j \right] \times \sum_{j \in Q} \left[\beta_1(j) \times \pi_j \right] \qquad (7.19)$$

La probabilidad β_t se puede calcular fácilmente como:

1. $\beta_T(i) = 1$ (7.20)

2. $\beta_t(i) = \sum_{j \in Q} a_{i,j} \times b_j(O_{t+1}) \times \beta_{t+1}(j)$ (7.21)

En la figura 7.2 se representa la secuencia de operaciones requeridas para el cálculo de β_t.

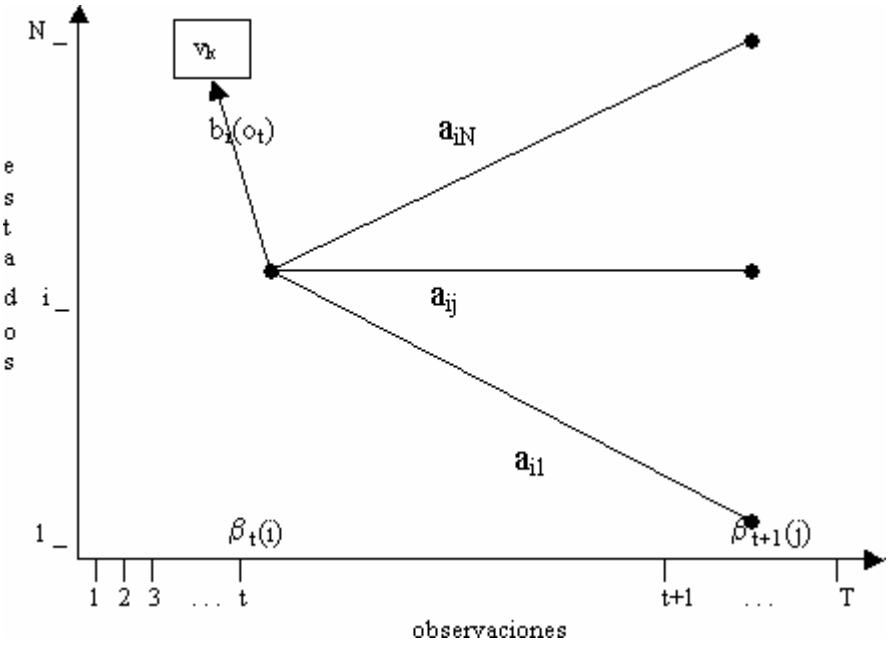


Figura 7.2. Secuencia de operaciones requeridas para el cálculo de β_t.

A continuación, se exponen dos cantidades muy útiles que se calculan a partir de α y β: $\gamma_t(j)$ y $\xi_t(i,j)$.

Cálculo de $\gamma_t(j)$.

Se define:

$$\gamma_t(j) = P(I_t = j \mid O, \lambda) \tag{7.22}$$

como la probabilidad de que un proceso esté en el estado *j* en el instante *t*, dados unos determinados *O* y λ. Usando la regla de la cadena, se obtiene:

$$P(O, I_t = j \mid \lambda) = P(O \mid \lambda) \times P(I_t = j \mid O, \lambda) \tag{7.23}$$

Lo que nos lleva a:

$$P(I_t = j \mid O, \lambda) = \frac{P(O, I_t = j \mid \lambda)}{P(O \mid \lambda)} \tag{7.24}$$

Usando de nuevo la regla de la cadena, se obtiene:

$$P(O, I_t = j \mid \lambda) = P(O_1,...,O_t \mid \lambda) \times P(O_{t+1},...,O_T \mid O_1,...,O_t, I_t = j, \lambda) \quad (7.25)$$

Se puede simplificar el término $P(O_{t+1},...,O_T \mid O_1,...,O_t, I_t = j, \lambda)$ a $P(O_{t+1},...,O_T \mid I_t = j, \lambda)$ gracias a la propiedad de Markov, según la cual una vez que se sabe que $I_t = j$, las observaciones $O_1,...,O_t$ son irrelevantes. Por lo tanto, $P(O, I_t = j \mid \lambda) = \alpha_t(j) \times \beta_t(j)$ y:

$$\gamma_t(j) = P(I_t = j \mid O, \lambda) = \frac{\alpha_t(j) \times \beta_t(j)}{P(O \mid \lambda)} \quad (7.26)$$

La probabilidad $P(O \mid \lambda)$ se ha calculado en la resolución del problema de evaluación anteriormente expuesto.

Cálculo de $\xi_t(i, j)$.

Se define:

$$\xi_t(i, j) = P(I_t = i, I_{t+1} = j \mid O, \lambda) \quad (7.27)$$

como la probabilidad de que un proceso esté en el estado *i* en el instante *t* y pase al estado *j* en el instante *t*+1, dados unos determinados *O* y λ. Usando la regla de la cadena se llega a:

$$P(I_t = i, I_{t+1} = j \mid O, \lambda) = \frac{P(I_t = i, I_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)} \quad (7.28)$$

Aplicando de nuevo la regla de la cadena al numerador se obtiene:

$$\begin{aligned} &P(I_t = i, I_{t+1} = j, O \mid \lambda) = \\ &P(I_t = i, O_1,...,O_t \mid \lambda) \times P(I_{t+1} = j, O_{t+1},...,O_T \mid I_t = i, O_1,...,O_t, \lambda) \end{aligned} \quad (7.29)$$

El último término de la expresión se puede simplificar porque las observaciones $O_1,...,O_t$ son irrelevantes si se sabe que $I_t = i$. La versión simplificada de este término se puede expandir con la regla de la cadena del siguiente modo:

$$\begin{aligned} &P(I_{t+1} = j, O_{t+1},...,O_T \mid I_t = i, \lambda) = \\ &P(I_{t+1} = j \mid I_t = i, \lambda) \times P(O_{t+1},...,O_T \mid I_t = i, I_{t+1} = j, \lambda) \end{aligned} \tag{7.30}$$

Una vez más, se puede simplificar el último término imponiendo la condición $I_t = 1$, por la propiedad de Markov. Y expandiendo este término simplificado se obtiene:

$$\begin{aligned} &P(O_{t+1},...,O_T \mid I_{t+1} = j, \lambda) = \\ &P(O_{t+1} \mid I_{t+1} = j, \lambda) \times P(O_{t+2},...,O_T \mid I_{t+1} = j, O_{t+1}, \lambda) \end{aligned} \tag{7.31}$$

Finalmente, podemos omitir O_{t+1} en el último término gracias a la propiedad de Markov y obtener la siguiente expresión:

$$\begin{aligned} &P(I_t = i, I_{t+1} = j \mid O, \lambda) = \\ &\frac{P(I_t = i, O_1..O_t \mid \lambda) P(I_{t+1} = j \mid I_t = i, \lambda) P(O_{t+1} \mid I_{t+1} = j, \lambda) P(O_{t+2}..O_T \mid I_{t+1} = j, \lambda)}{P(O \mid \lambda)} \end{aligned} \tag{7.32}$$

Toda esta expresión da lugar a:

$$\xi_t(i,j) = P(I_t = i, I_{t+1} = j \mid O, \lambda) = \frac{\alpha_t(i) \times a_{i,j} \times b_j(O_{t+1}) \times \beta_{t+1}(j)}{P(O \mid \lambda)} \tag{7.33}$$

Por lo tanto, para calcular $\xi_t(i,j)$ sólo es necesario calcular $\alpha_t(i)$ y $\beta_{t+1}(j)$.

7.2.2.2 El problema del descubrimiento. El algoritmo de Viterbi.

En el problema del descubrimiento se trata de determinar la secuencia de estados ocultos *I* que maximiza la probabilidad de que el modelo haya producido los datos, dados un vector de datos *O* y un modelo λ.

Se debe identificar un criterio según el cual maximizar nuestra selección de *I*. Luego se recorren todos los posibles *I* y se elige el que maximice el criterio. Es evidente que este método es poco eficiente dado el número de secuencias de estados posibles, por

lo tanto, es necesario un método más eficiente que aproveche las ventajas que introduce la propiedad de Markov.

El *algoritmo de Viterbi* es la solución más común a este problema. En él se define $\delta_t(j)$ como la probabilidad máxima de que una secuencia de estados de longitud *t* produzca las primeras *t* observaciones de *O* y que termine en el estado *j*. No mantiene un seguimiento de la secuencia de estados que llega a ese máximo, sólo sigue la probabilidad máxima correspondiente. Por este motivo, se define $\psi_t(j)$ para recuperar la secuencia de estados actual. Definimos $\psi_t(j)$ como el estado justo antes de *j* que permite llegar a la secuencia al valor máximo de probabilidad.

El algoritmo se desarrolla en los siguientes pasos:

1. Inicialización:

$$\delta_1(j) = \pi_j \times b_j(O_1) \tag{7.34}$$

$$\psi_1(j) = 0 \tag{7.35}$$

2. Recursión:

$$\delta_t(j) = \max_{i \in Q} \left[\delta_{t-1}(i) \times a_{i,j}\right] \times b_j(O_t) \tag{7.36}$$

$$\psi_1(j) = \arg\max_{i \in Q} \left[\delta_{t-1}(i) \times a_{i,j}\right] \tag{7.37}$$

3. Terminación:

$$P^* = \max_{i \in Q} \left[\delta_T(i)\right] \tag{7.38}$$

$$I_T^* = \arg\max_{i \in Q} \left[\delta_T(i)\right] \tag{7.39}$$

4. Descubrimiento de la secuencia:

$$I_T^* = \psi_{t+1}(I_{t+1}^*) \tag{7.40}$$

Computar P^* y I_T^* requiere de un orden de $N \times T$ operaciones, y el descubrimiento de la ruta más probable requiere T pasos adicionales.

7.2.2.3 El problema del entrenamiento. El algoritmo de Baum-Welch.

El problema del entrenamiento es el más complicado, ya que trata de buscar el modelo que maximice la probabilidad de haber originado unos datos, conociendo únicamente esa secuencia de datos observados.

No existe ningún método analítico para determinar el ajuste del modelo más probable. Por lo tanto, se deben utilizar técnicas iterativas, como el *algoritmo de Baum-Welch*, o métodos de gradiente descendientes. A continuación, se hace una descripción del algoritmo de Baum-Welch.

El entrenamiento es especialmente difícil, ya que sólo se dispone de la secuencia de datos observados que el proceso produce. No se dispone de las transiciones asociadas a los estados que han ocurrido. Si esta información estuviera disponible, el entrenamiento sería mucho más sencillo, pero sin estas variables ocultas, se debe desarrollar suposiciones sobre las transiciones entre estados que han ocurrido.

Si se suma $\gamma_t(i)$ sobre ***t*,** se obtiene el número esperado de veces que el estado *i* es visitado, o lo que es lo mismo, el número de transiciones hechas desde el estado *i*, si se excluye el último instante de tiempo. Por lo tanto, se obtiene lo siguiente:

- $\sum_{t=1}^{T-1}\gamma_t(i)$ es el número esperado de transiciones hechas desde el estado *i*.

- $\sum_{t=1}^{T-1}\xi_t(i,j)$ es el número esperado de transiciones hechas desde el estado *i* al estado *j*.

Con estas herramientas, se pueden contar las transiciones de los estados para ajustar el modelo.

Las fórmulas de reestimación del algoritmo de Baum-Welch son las siguientes:

1. $\hat{\pi}_i = \gamma_1(i), \quad para \quad j \in Q$ (7.41)

2. $$\hat{a}_{i,j} = \frac{\sum_{i=1}^{T-1}\xi_t(i,j)}{\sum_{i=1}^{T-1}\gamma_t(i)}$$ (7.42)

3. $$\hat{b}_i(k) = \frac{\sum_{i=1}^{T}\left[\gamma_t(i)\times\begin{cases}1 & si \quad O_t = k\\ 0 & en \quad otro \quad caso\end{cases}\right]}{\sum_{i=1}^{T}\gamma_t(i)}$$ (7.43)

En el numerador de la última expresión sólo se incluyen aquellos *t* que cumplen $O_t = k$, donde *k* es la observación que está siendo examinada.

Cuando se tiene un conjunto de *L* secuencias de observaciones, $O^1,...,O^L$, se realiza una estimación similar pero sobre todas las secuencias de una vez. Por lo tanto, las fórmulas anteriores se pueden escribir como:

$$1.\ \hat{\pi}_i = \frac{\sum_{l=1}^{L} \gamma_1^{\,l}(i)}{L} \tag{7.44}$$

$$2.\ \hat{a}_{i,j} = \frac{\sum_{l=1}^{L}\sum_{i=1}^{T-1} \xi_t^{\,l}(i,j)}{\sum_{l=1}^{L}\sum_{i=1}^{T-1} \gamma_t^{\,l}(i)} \tag{7.45}$$

$$3.\ \hat{b}_i(k) = \frac{\sum_{l=1}^{L}\sum_{i=1}^{T}\left[\gamma_t^{\,l}(i) \times \begin{cases} 1 & si \quad O_t^{\,l} = k \\ 0 & en \quad otro \quad caso \end{cases}\right]}{\sum_{l=1}^{L}\sum_{i=1}^{T} \gamma'^{\,l}{}_t(i)} \tag{7.46}$$

El algoritmo entrena un modelo del siguiente modo:

1. Crea un modelo inicial λ_0. Se puede hacer aleatoriamente o mezclando conocimiento que se tenga del proceso con la aleatoriedad.

2. Usando las fórmulas anteriormente descritas, crea λ_1 de λ_0. Luego crea λ_2 de λ_1.

3. Repite este proceso hasta que se llega a la convergencia, o hasta extinguir recursos, dando lugar a un modelo final estimado λ'.

Se puede encontrar una descripción detallada del algoritmo de Baum-Welch en [Sundaram, 2000]. El método de entrenamiento de Baum-Welch difiere del entrenamiento de Viterbi en que el algoritmo de Baum-Welch asume que cualquier

estado puede ocurrir en cualquier instante con alguna probabilidad y actualiza los parámetros del modelo basándose en estas probabilidades en vez de elegir una única secuencia de estados como mejor y actualizar los parámetros del modelo según esta secuencia de estados.

7.2.3 Arquitecturas de HMMs.

Un HMM puede ser representado como un grafo dirigido de transiciones /emisiones. La arquitectura específica que permita modelar de la mejor forma posible las propiedades observadas depende en gran medida de las características del problema. Las arquitecturas mas usadas son:

1. Ergódicas o completamente conectadas, en las cuales cada estado del modelo puede ser alcanzado desde cualquier otro estado en un número finito de pasos (figura 7.3).

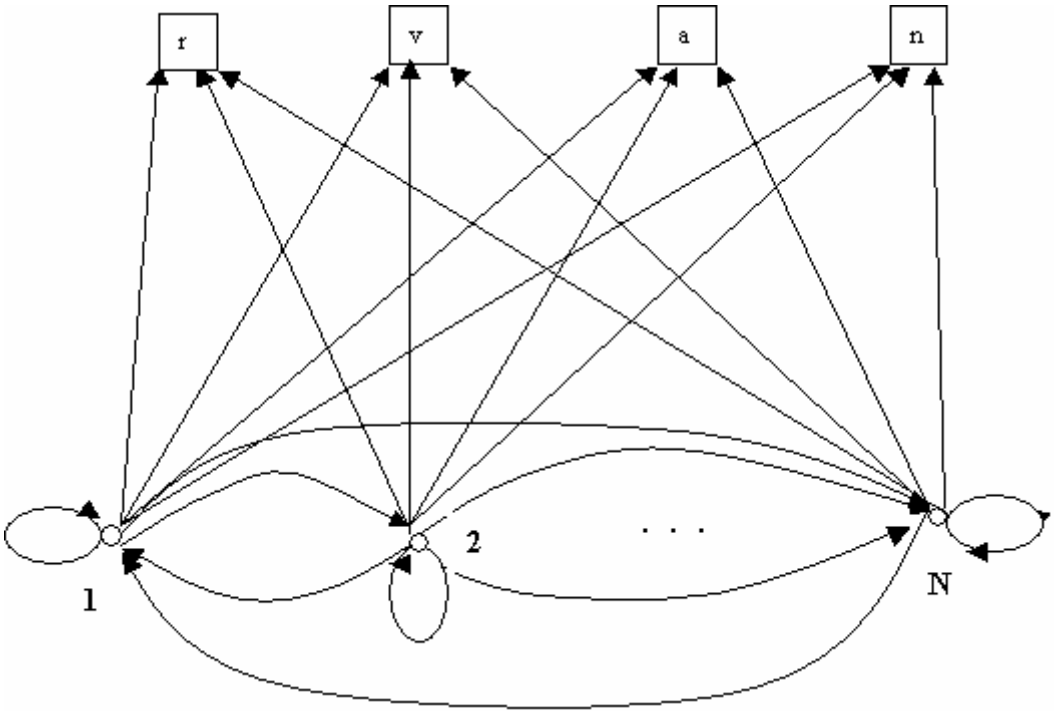


Figura 7.3. Ejemplo de arquitectura de HMM ergódica.

2. Izquierda-derecha, hacia adelante o Bakis, las cuales tienen la propiedad de que en la medida que el tiempo crece se avanza en la secuencia de observación asociada O, y en esa misma medida el índice que señala el estado del modelo permanece o crece, es decir, los estados del sistema van de izquierda a derecha (figura 7.4). En

secuencias biológicas y en reconocimiento de la voz, estas arquitecturas modelan bien los aspectos lineales de las secuencias.

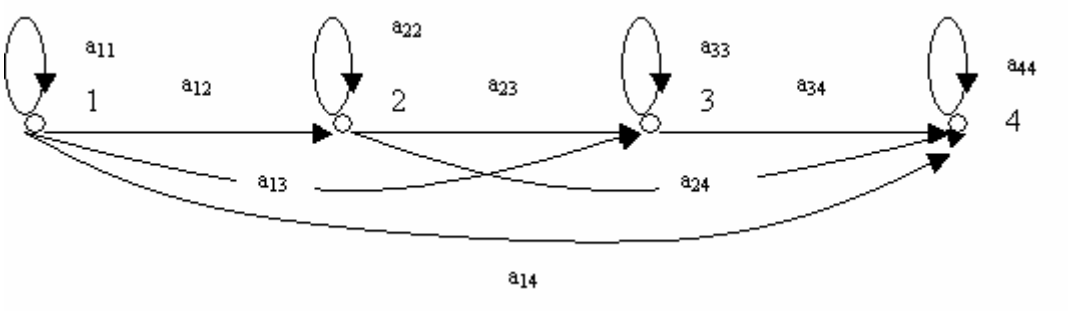


Figura 7.4. Modelo izquierda-derecha con 4 estados.

3. Izquierda-derecha paralelas, son dos arquitecturas izquierda-derecha conectadas entre sí (figura 7.5).

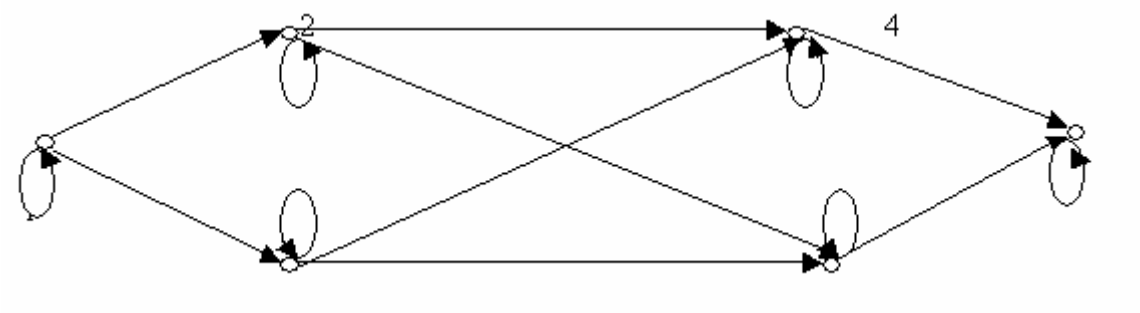


Figura 7.5. Modelo izquierda-derecha paralelo con 6 estados.

7.2.4 Aplicaciones de los HMMs.

Hasta ahora se han considerado secuencias de observación caracterizadas por símbolos discretos que pertenecen a un alfabeto finito y que usan probabilidades discretas en cada estado del modelo. No obstante, en algunos problemas las secuencias de observación son señales continuas, y por lo tanto, es conveniente usar HMMs con densidades de observación continuas y funciones de densidad de probabilidades que aseguren la reestimación consistente de los parámetros del modelo. En este trabajo se usarán funciones de densidad de probabilidad gaussianas para las observaciones.

Por otra parte, se ha revisado el entrenamiento de los parámetros del modelo con una sola secuencia de observación, pero en la práctica existen muchas aplicaciones, tales como reconocimiento de la voz y alineamiento de secuencias biológicas, en las que se debe trabajar con múltiples secuencias de observación para hacer mas fiable el

modelo, esto es, $O = [O^{(1)}, O^{(2)}, ..., O^{(k)}]$, donde $O^{(k)} = (O_1^{(k)} O_2^{(k)} ... O_T^{(k)})$ es la k-ésima secuencia de observación.

Una de las principales aplicaciones de los HMMs se encuentra en el reconocimiento del habla. En un sistema típico de reconocimiento del habla se asume que la secuencia de los vectores de habla correspondientes a una secuencia de palabras observada está generada por un modelo paramétrico. Si se usa un modelo de Markov, el problema de encontrar la probabilidad condicional de evidencia acústica se reemplaza por la estimación de los parámetros del modelo de Markov. Dado un conjunto de ejemplos de entrenamiento, los parámetros del modelo pueden ser estimados por medio de un procedimiento de reestimación robusto y eficiente. Este proceso se llama *entrenamiento acústico* y un tipo de procedimiento usado para reestimar los parámetros del modelo es el algoritmo de reestimación de Baum-Welch.

Un reconocedor de habla relaciona los vectores de habla de entrada con la secuencia de la palabra que necesita ser reconocida [Rabiner y Juang, 1993]. Existe un problema inherente en esto, porque la relación puede no ser uno-a-uno, debido a que distintas secuencias de palabras puedan tener vectores de habla similares. Este problema se aborda desde un punto de vista estadístico mediante el uso de probabilidades. Dado un vector de entrada acústico, el reconocedor elige la secuencia de palabras más probable. Si los vectores de habla u observaciones se representan por $O = o_1,...,o_t$, donde o_t es el vector de habla observado en el instante t, la salida del reconocedor de habla será:

$$\hat{W} = \arg\max P(w_i \mid O) \tag{7.47}$$

donde w_i es la *i-ésima* palabra del vocabulario.

Usando la regla de Bayes, esta probabilidad se puede calcular como:

$$P(w_i \mid O) = \frac{P(O \mid w_i)P(w_i)}{P(O)} \tag{7.48}$$

Si la probabilidad $P(w_i)$ es conocida, entonces la palabra hablada más probable depende de la probabilidad $P(O|w_i)$. Si la dimensionalidad de la observación es grande, la computación de $P(O|w_i)$ es impracticable. Sin embargo, si se asume que la producción de la palabra se corresponde a un modelo paramétrico como son los

modelos de Markov, entonces la estimación de $P(O|w_i)$ se reemplaza por un problema más simple que es el de la estimación de los parámetros del modelo.

En un sistema típico de reconocimiento del habla, las palabras de una secuencia de palabras a reconocer, son modeladas usando un modelo paramétrico. Este modelo es un HMM. Se asume que la secuencia de los vectores de habla observados correspondientes a cada palabra se genera por el modelo correspondiente a esa palabra [Rabiner y Juang, 1986].

7.3 Objetivo y metodología general de los experimentos.

7.3.1 Metodología general.

El objetivo principal de este estudio es evaluar los algoritmos de aprendizaje basados en algoritmos genéticos descritos en el capítulo anterior.

Como ya se comentó en la introducción de este capítulo, se utilizará un problema de referencia, como es el de la clasificación de series temporales basadas en modelos ocultos de Markov. Este problema de referencia tiene una solución estándar como ya se comentó que resulta de la combinación del algoritmo de identificación de Baum-Welch junto con la estimación de la probabilidad $P(O \mid \lambda)$.

En primer lugar, estudiaremos la eficiencia del algoritmo basado en el método de Baum-Welch, específicamente en lo que a su capacidad de generalización se refiere y las características de la muestra de entrenamiento. Esto es necesario para poder analizar los resultados que se obtengan después con la máquina de estados borrosa.

Uno de los parámetros importantes es la longitud de la serie temporal, ya que determinará el número medio de ocurrencias de situaciones en las que difieran el modelo 1 y el modelo 2 propuestos. Como veremos, esto tiene una influencia notable, ya que al aumentar el número de datos por serie temporal, se logrará una mejora sustancial en la exactitud en el proceso de evaluación de los HMM.

Una vez que se ha estudiado el problema de referencia junto a su solución estándar, se pasará a realizar el estudio sobre los algoritmos propuestos. Se comenzará con el sistema tipo Pittsburgh, donde se investigará la calidad de su aprendizaje en

función de algunos parámetros críticos como el nivel de solapamiento entre poblaciones sucesivas (parámetro alfa) y las probabilidades de los operadores genéticos cruce, mutación y reproducción. Finalmente, se comprobará que la longitud de las series de datos utilizadas tiene una influencia también en el caso del sistema tipo Pittsburgh, evaluándose la importancia de esta influencia.

Tras estudiar el sistema tipo Pittsburgh, se pasará al sistema tipo Michigan. En este caso, se ha centrado el estudio en el efecto de la frecuencia de disparo del algoritmo genético, la comparación con un algoritmo de búsqueda aleatoria simple y el efecto de la longitud de las series temporales en la eficiencia del clasificador encontrado.

7.3.2 Modelo utilizado en el estudio.

En las pruebas que se presentan a continuación se utilizarán dos procesos estocásticos de características ligeramente diferentes, basados en el modelo oculto de Markov, con funciones de densidad de probabilidad para las observaciones de tipo gaussiano. Mediante estos modelos se generarán series temporales que deberán ser clasificadas. En ese proceso se utilizará el algoritmo de Baum-Welch para comparar los resultados obtenidos mediante la utilización de la máquina de estados borrosa como clasificador en la arquitectura Pittsburgh y en la arquitectura Michigan.

Comencemos por describir los modelos utilizados. Se trata de dos modelos simples con sólo dos estados, S_1 y S_2. Las características específicas de ambos modelos se recogen en las tablas 7.1 y 7.2 y se representan en las figuras 7.6 y 7.7.

Características del modelo 1					
Características de las funciones de densidad de probabilidad para las observaciones	S_1	S_2	Probabilidades de transición entre estados	Desde S_1	Desde S_2
Centro	0.2	0.8	Hasta S_1	0.5	0.8
Desviación	0.1	0.1	Hasta S_2	0.5	0.2

Tabla 7.1. Características del primer HMM.

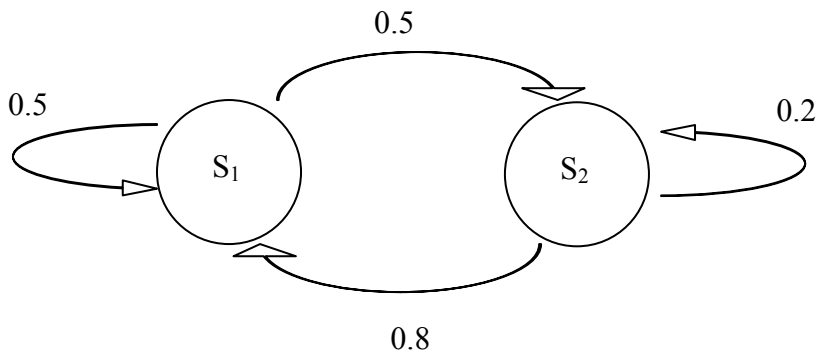


Figura 7.6. Diagrama de transiciones en el modelo 1.

Características del modelo 2					
Características de las funciones de densidad de probabilidad para las observaciones	S_1	S_2	Probabilidades de transición entre estados	Desde S_1	Desde S_2
Centro	0.2	0.8	Hasta S_1	0.5	0.6
Desviación	0.1	0.1	Hasta S_2	0.5	0.4

Tabla 7.2. Características del segundo HMM.

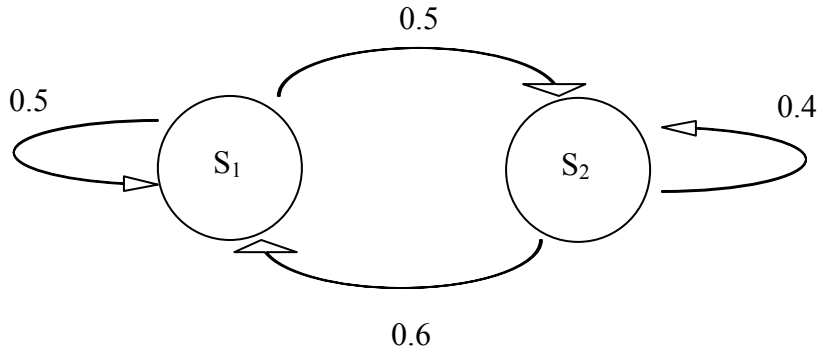


Figura 7.7. Diagrama de transiciones en el modelo 2.

Como se ve fácilmente, la diferencia entre los modelos no está en los centros y desviaciones de los estados sino en las probabilidades de transición entre los estados. Por consiguiente, en las pruebas realizadas lo que se pretende es que la máquina finita de estados borrosa "aprenda" la dependencia existente entre muestras consecutivas y sea capaz de distinguir las trazas pertenecientes a cada modelo.

Este problema de clasificación es complicado, de resolución no inmediata, debido a que los dos modelos difieren únicamente en dos probabilidades de transición, y las series temporales que generan pueden tener características muy similares.

En algunas de las pruebas presentadas se utiliza el algoritmo de Baum-Welch para la identificación del modelo junto con la evaluación, es decir, el cálculo de $P(O \mid \lambda)$, siendo O la serie temporal y λ el modelo. Como se describió al principio de este capítulo, el algoritmo de Baum-Welch está específicamente diseñado para obtener los parámetros de un modelo oculto de Markov a partir de series temporales. Con los modelos identificados con este algoritmo se intentará clasificar también las trazas de ambos modelos, mediante el cálculo de las probabilidades $P(O \mid \lambda_1)$ y $P(O \mid \lambda_2)$, siendo

O la serie temporal y λ_1 y λ_2 los modelos a los que se quiere asignar la serie temporal: si $P(O \mid \lambda_1) > P(O \mid \lambda_2)$ asignaremos O a λ_1 y en caso contrario la asignaremos a λ_2.

Obviamente, este algoritmo tiene ventajas adicionales para este problema de clasificación sobre los clasificadores que diseñamos basados en las máquinas finitas de estados borrosas, y en ningún momento se pretende sustituir este algoritmo por los clasificadores borrosos propuestos. Por el contrario, lo que se pretende con los resultados de la clasificación obtenidos con el algoritmo de Baum-Welch es tener una medida de referencia con la que comparar los resultados obtenidos en la clasificación con las máquinas de estados borrosas.

7.4 Estudio del error de clasificación en el método basado en la identificación del HMM en función de la longitud de cada serie temporal.

7.4.1 Introducción.

En el primer experimento nos centraremos en el algoritmo de referencia, que permite clasificar las series de datos simuladas a partir de los modelos HMM. El objetivo es estudiar la eficiencia del algoritmo Baum-Welch en la reconstrucción de los modelos y la clasificación mediante la evaluación de $P(O \mid \lambda)$ de las series de datos en función del número de muestras por serie temporal de las series que componen los conjuntos de entrenamiento usados por el algoritmo en la reconstrucción. Estudiaremos aquí la influencia del número de datos por serie temporal en la capacidad del algoritmo para realizar una clasificación.

7.4.2 Descripción del experimento.

Las series de datos generadas por cada uno de los HMM se agruparán en distintos conjuntos: 60 para el conjunto de entrenamiento y 60 para el conjunto de test. Se realiza la identificación de los modelos asociados a las series temporales del conjunto de entrenamiento mediante el algoritmo de Baum-Welch. A continuación, se clasificarán las trazas pertenecientes a los conjuntos de entrenamiento y de test asociándolas a uno u otro de los modelos reconstruidos.

Se hacen distintos experimentos repitiendo esta metodología, para distintas longitudes de series temporales: 30 datos, 45 datos, 70 datos, 100 datos, 150 datos, 200 datos y 500 datos por serie. Los resultados de estos experimentos se muestran en la tabla 7.3. En la figura 7.8 se representan el error de entrenamiento y el error del test para las diferentes longitudes de las series temporales.

	Longitud de la serie temporal	Error de entrenamiento	Error de test
Experimento 1	30	20%	23.33%
Experimento 2	45	18%	19%
Experimento 3	70	15%	18%
Experimento 4	100	11.67%	8.33%
Experimento 5	150	4%	1%
Experimento 6	200	2%	3%
Experimento 7	500	0%	0%

Tabla 7.3. Error mediante identificación por Baum-Welch en función de la longitud de la serie temporal.

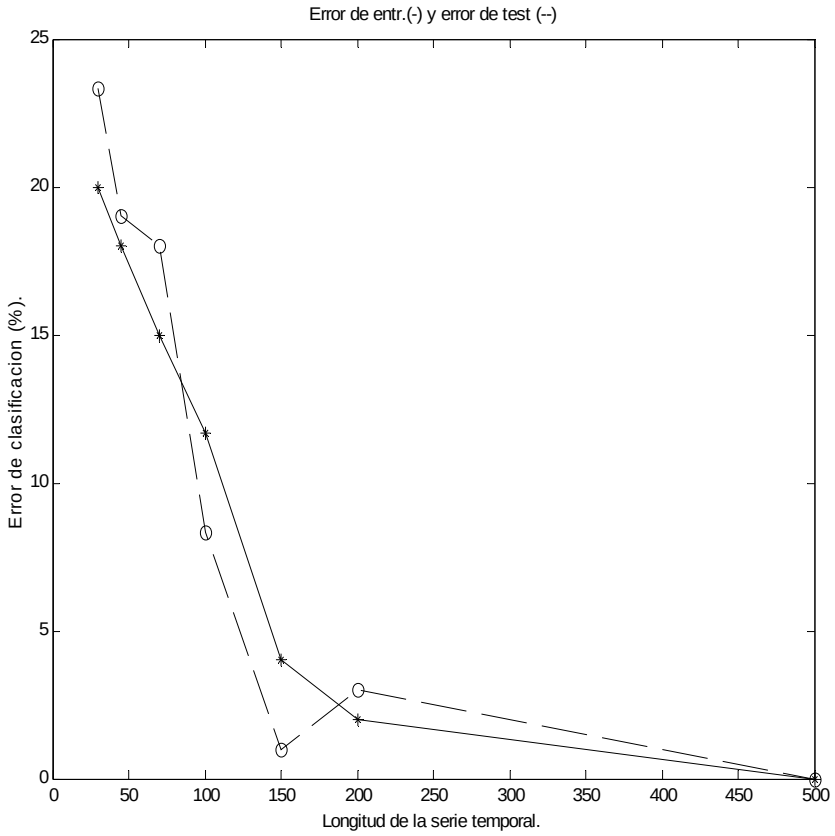


Figura 7.8. Representación del error de entrenamiento y el error de test para diferentes longitudes de las series temporales (Baum-Welch y evaluación de $P(O \mid \lambda)$).

7.4.3 Conclusión.

Se puede apreciar que el error disminuye a medida que aumenta el número de muestras por serie temporal de las trazas que componen los conjuntos de entrenamiento. Por lo tanto, el número de muestras por serie temporal de las trazas que componen los conjuntos de entrenamiento es un factor decisivo en el rendimiento del algoritmo.

Se ha comprobado experimentalmente que la mayor dependencia del número de muestras por serie temporal de las trazas está en el proceso de estimación de $P(O|\lambda)$, es decir, en el proceso de evaluación. Mientras que el algoritmo de Baum-Welch identifica los modelos con una eficiencia adecuada a partir de un número mínimo de muestras determinado, el proceso de evaluación mejora su eficiencia a medida que se aumenta el número de datos por serie temporal. La dependencia proviene de la propia naturaleza estocástica de los datos producidos, que implica que la probabilidad de que se den suficientes situaciones en las que las series temporales ofrezcan particularidades que puedan ser asignables a uno u otro modelo se incremente con el número de datos de la serie temporal.

7.5 Estudio de un sistema Pittsburgh en la clasificación de series de datos producidas por un proceso de Markov.

7.5.1 Introducción.

En esta parte del capítulo se estudiará la aplicación de un sistema tipo Pittsburgh para el diseño de máquinas de estado borrosas utilizables en la clasificación de series temporales, tomando como referencia los conjuntos de series de datos generados, con 15 datos por serie. Los conjuntos de entrenamiento y de test se componen de 60 series con estas características cada uno.

El objetivo de este estudio, igual que en el caso de los sistemas tipo Michigan, será analizar la influencia de diversos parámetros del algoritmo. Las primera pruebas realizadas tendrán como objetivo observar la influencia del grado de solapamiento entre poblaciones de generaciones sucesivas (parámetro *alfa*), sobre el resultado. El segundo

estudio complementa al anterior, analizando como la modificación en la preponderancia de unos operadores genéticos sobre otros influye en el proceso de búsqueda. Finalmente, se mostrará cómo el aumento en el número de datos por serie temporal permite mejorar la calidad del aprendizaje.

7.5.2 Análisis de la influencia del parámetro *alfa*.

7.5.2.1 Descripción del experimento.

Se han realizado una serie de pruebas para la clasificación de series temporales simuladas con diferentes valores del parámetro *alfa*. Recordemos que este parámetro establece la fracción de la población que se selecciona para la siguiente generación de entre los mejores individuos y no sufre modificación (solape). Los valores del parámetro *alfa* que se han investigado son: 0.1, 0.3, 0.5, 0.7 y 0.9. Se han realizado un total de 5 experimentos por valor del parámetro *alfa*.

Los parámetros del entrenamiento son los mismos para todas las pruebas realizadas. Se muestran en la tabla 7.4.

Parámetro	Abreviatura	Valor
Tamaño de la población	*num_maquinas*	200
Número de reglas de cada máquina	*num_reglas*	10
Número de estados de cada máquina	*num_estados*	4
Número del estado de detección	*num_detec*	4
Umbral para considerar que el estado de detección se activa a alta	*param_alta*	0.7
Probabilidad del operador de reproducción	*p1*	5%
Probabilidad del operador de mutación	*p2*	30%
Probabilidad del operador de cruce	*p3*	65%
Número de reglas a mutar en caso de mutación	*mutar_reglas*	5
Número de elementos dentro de una regla a mutar en caso de mutación	*mutar_elementos*	5

Tabla 7.4. Valores de los parámetros del entrenamiento de los sistemas Pittsburgh.

7.5.2.2 Resumen de resultados.

Los resultados obtenidos se describen en la tabla 7.5. Las dos primeras columnas dan cuenta del valor medio y la desviación del menor error sobre el conjunto de test alcanzado con las máquinas obtenidas durante el entrenamiento. Las siguientes dos

columnas dan el error medio y la desviación del menor error de entrenamiento alcanzado por máquinas obtenidas durante el entrenamiento y cuyo error de test es el menor. Las dos últimas columnas dan el error medio y la desviación del mejor error de entrenamiento.

alfa	Media del mejor error en test	Desviación del mejor error en test	Media del mejor error en el entr. (con menor error en test)	Desv. del mejor error en el entr. (con menor error en test)	Media del mejor error de entr.	Desv. del mejor error de entr.
0.1	34.17%	0.0441	28.33%	0.0933	17.92%	0.0285
0.3	36.67%	0.0441	26.67%	0.0441	17.22%	0.0255
0.5	37%	0.0139	19.33%	0.0365	17%	0.0139
0.7	35.33%	0.0321	21%	0.0325	19.33%	0.0190
0.9	35.67%	0.0522	30.67%	0.0508	27%	0.0415

Tabla 7.5. Resumen de los resultados del entrenamiento en el sistema tipo Pittsburgh para diferentes valores del parámetro *alfa*.

Los resultados de la tabla se han representado gráficamente en la figura 7.9. En ella se puede apreciar claramente como un valor excesivo de *alfa* empeora los resultados, lo que es debido al problema de la convergencia prematura en este tipo de sistemas. Esto ocurre en los tres tipos de errores analizados. Se observa que el error de entrenamiento para el menor error de test es mejor para el parámetro *alfa* = 0.5. El error de test no sufre grandes variaciones y no da buenos resultados, probablemente debido a la escasa longitud de las trazas, que impide un aprendizaje correcto del modelo subyacente.

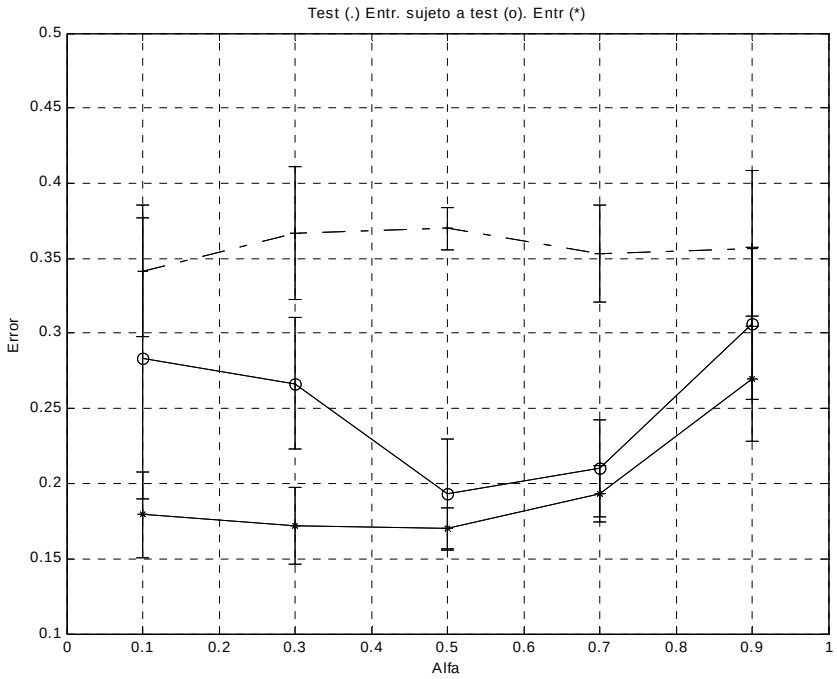
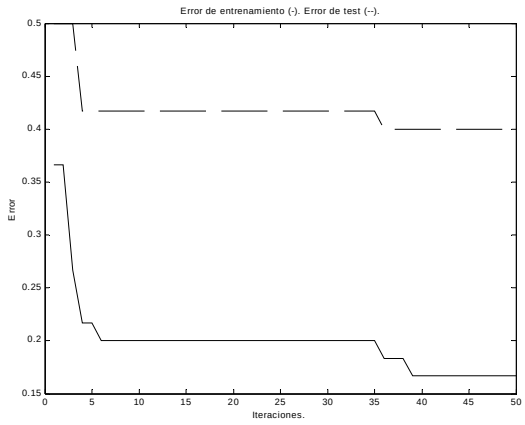


Figura 7.9. Errores de entrenamiento (*), test (--) y mejor error de entrenamiento para el mejor de test (o).

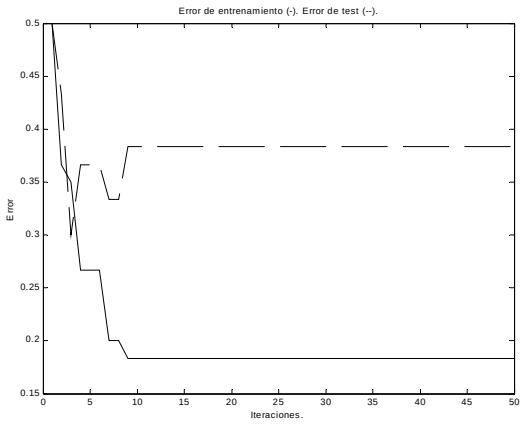
7.5.2.3 Curvas de entrenamiento y test.

A continuación, en las figuras 7.10, 7.11, 7.12, 7.13 y 7.14 se muestran las curvas de entrenamiento y test de las pruebas realizadas, cuyos resultados se expusieron anteriormente de forma resumida.

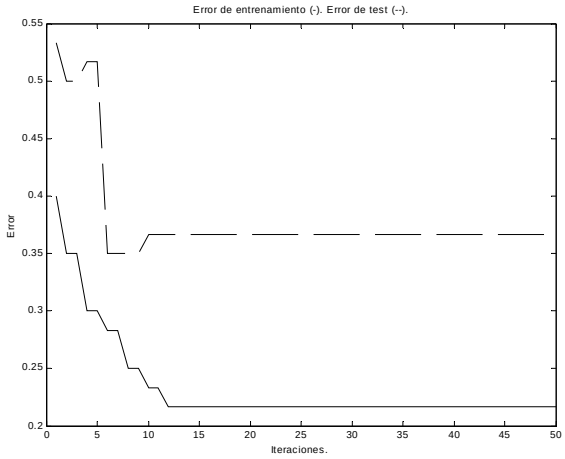
Primera prueba: 10% de solape (*alfa* = 0.1).



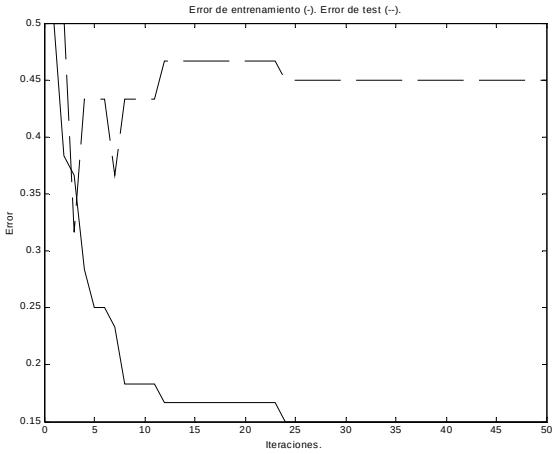
Mínimo error de entrenamiento: 16.67%
Mínimo error de test: 40%
Error de entrenamiento (mínimo del test): 16.67%



Mínimo error de entrenamiento: 18.33%
Mínimo error de test: 30%
Error de entrenamiento (mínimo del test): 30%



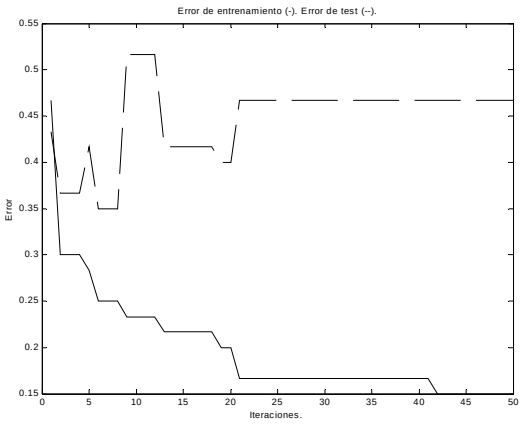
Mínimo error de entrenamiento: 21.67%
Mínimo error de test: 35%
Error de entrenamiento (mínimo del test): 25%



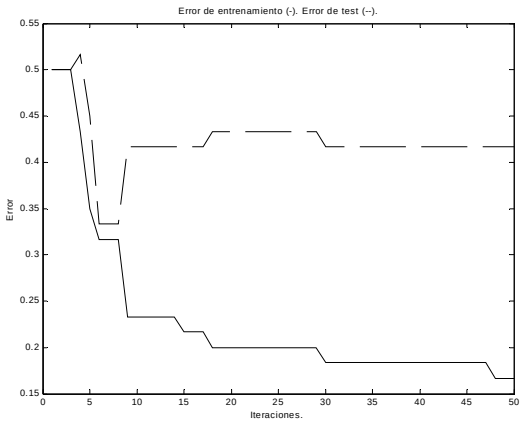
Mínimo error de entrenamiento: 15%
Mínimo error de test: 31.67%
Error de entrenamiento (mínimo del test): 36.67%

Figura 7.10. Curvas de entrenamiento y test para *alfa* = 0.1 (un 10% de la población es solapada con la siguiente generación).

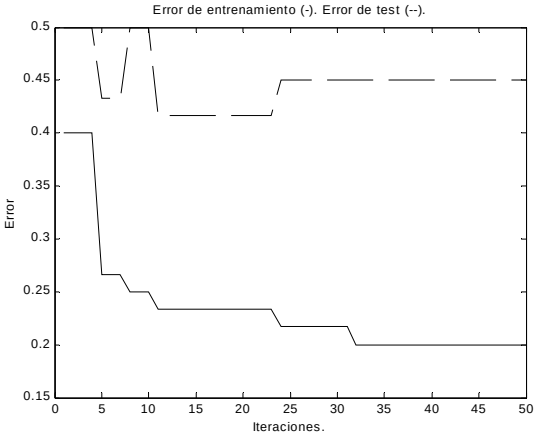
Segunda prueba: 30% de solape (*alfa* = 0.3).



Mínimo error de entrenamiento: 15%
Mínimo error de test: 35%
Error de entrenamiento (mínimo del test): 25%



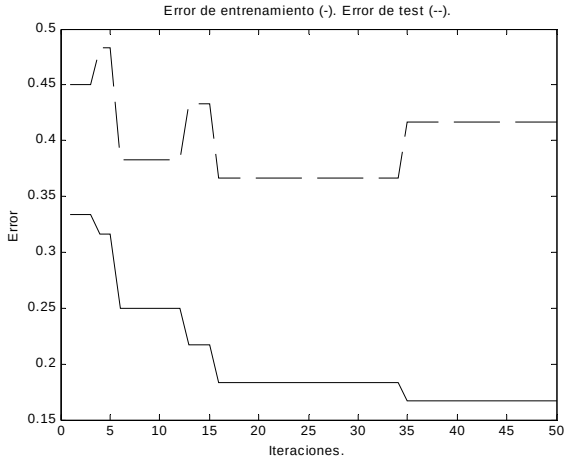
Mínimo error de entrenamiento: 16.67%
Mínimo error de test: 33.33%
Error de entrenamiento (mínimo del test): 31.67%



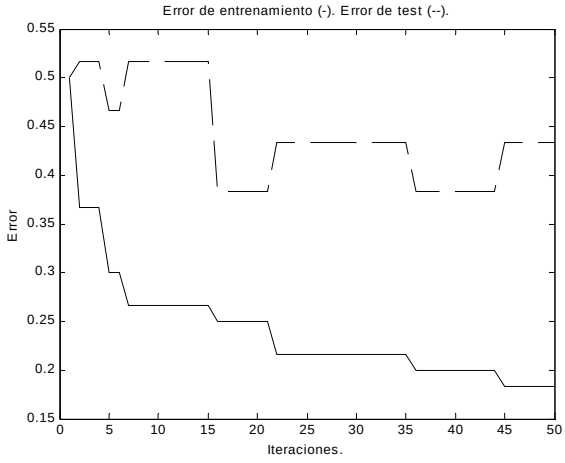
Mínimo error de entrenamiento: 20%
Mínimo error de test: 41.67%
Error de entrenamiento (mínimo del test): 23.33%

Figura 7.11. Curvas de entrenamiento y test para *alfa* = 0.3 (un 30% de la población es solapada con la siguiente generación).

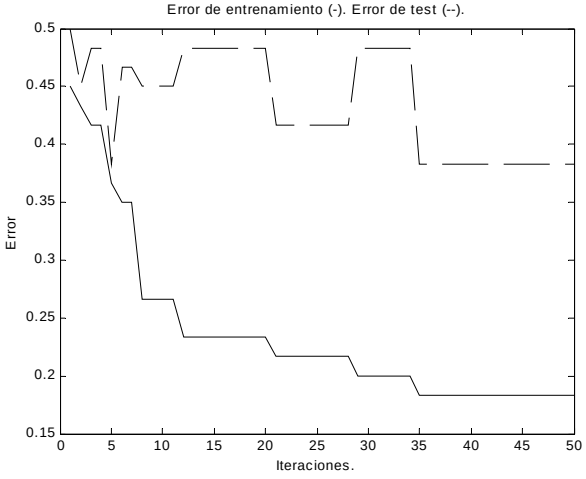
Tercera prueba: 50% de solape (*alfa* = 0.5).



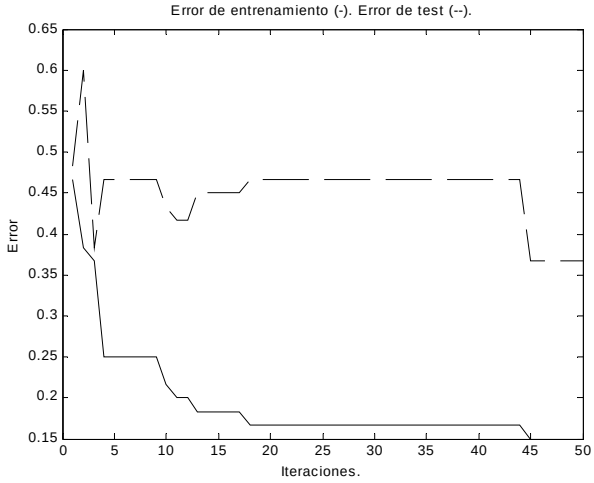
Mínimo error de entrenamiento: 16.67%
Mínimo error de test: 36.67%
Error de entrenamiento (mínimo del test): 18.33%



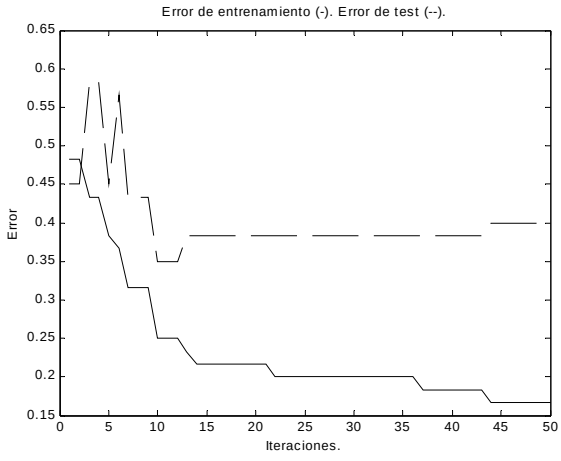
Mínimo error de entrenamiento: 18.33%
Mínimo error de test: 38.33%
Error de entrenamiento (mínimo del test): 31.67%



Mínimo error de entrenamiento: 18.33%
Mínimo error de test: 38.33%
Error de entrenamiento (mínimo del test): 18.33%



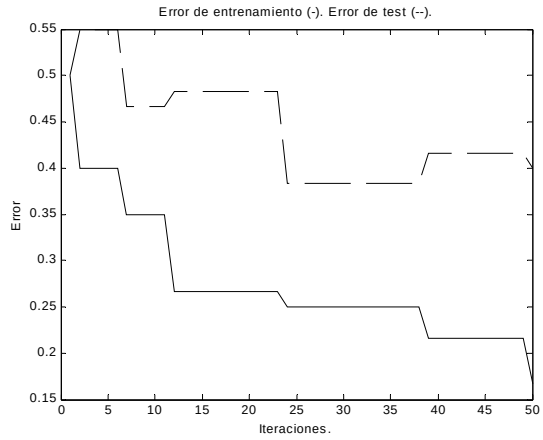
Mínimo error de entrenamiento: 15%
Mínimo error de test: 36.67%
Error de entrenamiento (mínimo del test): 15%



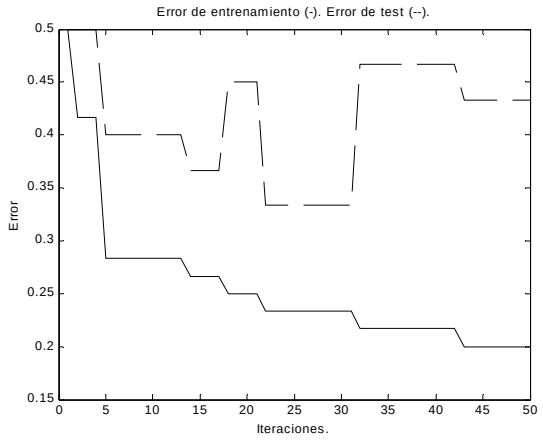
Mínimo error de entrenamiento: 16.67%
Mínimo error de test: 35%
Error de entrenamiento (mínimo del test): 25%

Figura 7.12. Curvas de entrenamiento y test para *alfa* = 0.5 (50% de solape).

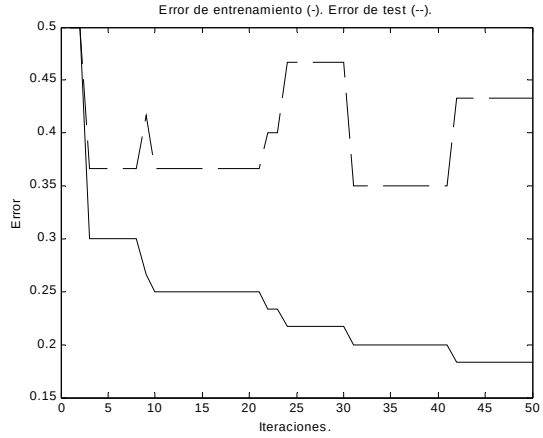
Cuarta prueba: 70% de solape (*alfa* = 0.7).



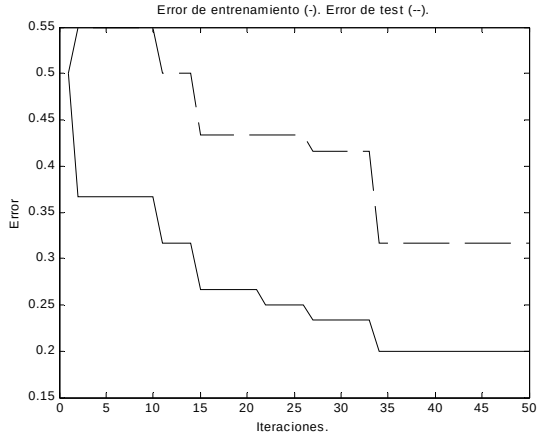
Mínimo error de entrenamiento: 16.67%
Mínimo error de test: 40%
Error de entrenamiento (mínimo del test): 16.67%



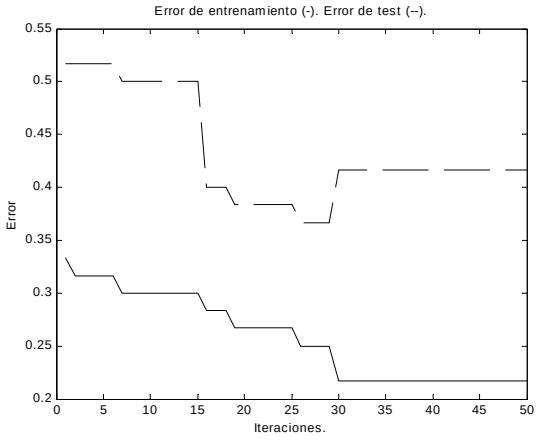
Mínimo error de entrenamiento: 20%
Mínimo error de test: 33.33%
Error de entrenamiento (mínimo del test): 23.33%



Mínimo error de entrenamiento: 18.33%
Mínimo error de test: 35%
Error de entrenamiento (mínimo del test): 20%



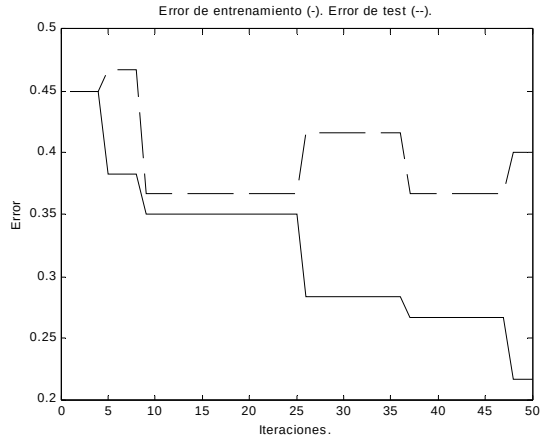
Mínimo error de entrenamiento: 20%
Mínimo error de test: 31.67%
Error de entrenamiento (mínimo del test): 20%



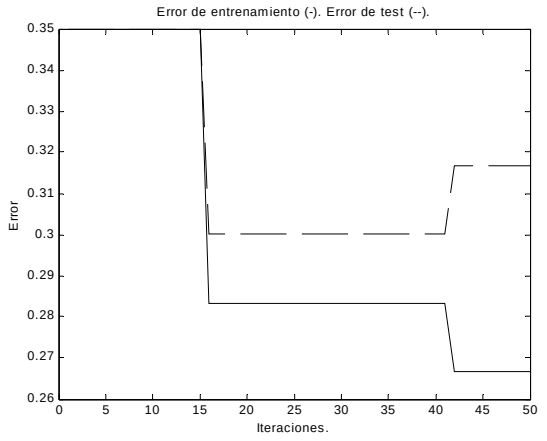
Mínimo error de entrenamiento: 21.67%
Mínimo error de test: 36.67%
Error de entrenamiento (mínimo del test): 25%

Figura 7.13. Curvas de entrenamiento y test para *alfa* = 0.7 (un 70% de la población es solapada con la siguiente generación).

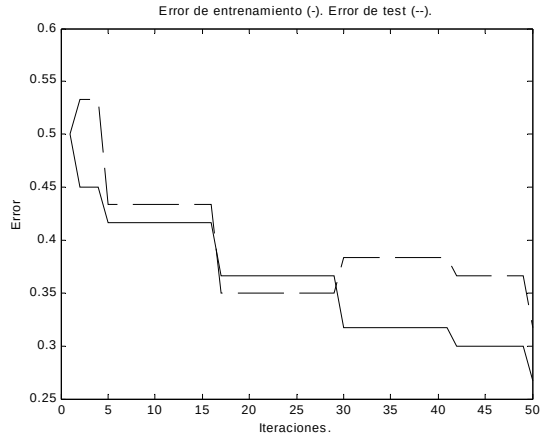
Quinta prueba: 90% de solape (*alfa* = 0.9).



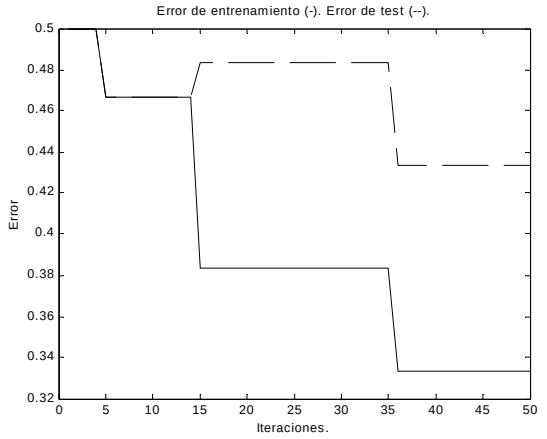
Mínimo error de entrenamiento: 21.67%
Mínimo error de test: 36.67%
Error de entrenamiento (mínimo del test): 26.67%



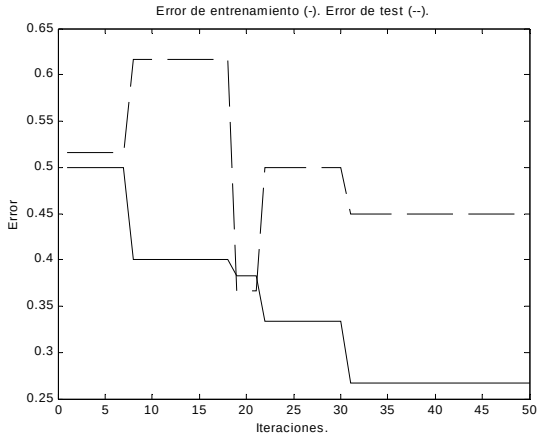
Mínimo error de entrenamiento: 26.67%
Mínimo error de test: 30%
Error de entrenamiento (mínimo del test): 28.33%



Mínimo error de entrenamiento: 26.67%
Mínimo error de test: 31.67%
Error de entrenamiento (mínimo del test): 26.7%



Mínimo error de entrenamiento: 33.3%
Mínimo error de test: 43.33%
Error de entrenamiento (mínimo del test): 33.3%



Mínimo error de entrenamiento: 26.67%
Mínimo error de test: 36.67%
Error de entrenamiento (mínimo del test): 38.33%

Figura 7.14. Curvas de entrenamiento y test para *alfa* = 0.9 (un 90% de la población es solapada con la siguiente generación).

7.5.2.4 Discusión de los resultados.

En líneas generales, y en lo que al error de entrenamiento se refiere, se observa un empeoramiento a medida que aumenta el solape en las poblaciones. Esto es consecuencia de la pérdida de diversidad en las poblaciones sucesivas, que hace más costoso el proceso de búsqueda y lleva al problema de la convergencia prematura.

Las ventajas computacionales derivadas de mantener un número de individuos de la población en la siguiente generación hay que sopesarlas teniendo en cuenta este problema. Sin embargo, se observa que con un solape considerable (por ejemplo, *alfa* = 0.5) se consiguen resultados muy similares y en algunos casos mejores respecto a valores menores del solapamiento. En ese sentido, es interesante resaltar el comportamiento del menor error en el entrenamiento para el mejor error en el test, que en el experimento con *alfa* = 0.5, aventaja al resto significativamente y muestra una tendencia al empeoramiento tanto para el incremento como para el decremento de *alfa* (ver figura 7.9).

Un análisis de las curvas de aprendizaje, para el entrenamiento y el test, muestra como la curva de test no sigue el mismo comportamiento que la curva de entrenamiento, revelando que no se está aprendiendo realmente el modelo subyacente, sino características intrínsecas al conjunto de entrenamiento. En muchas de las gráficas se observa además un empeoramiento progresivo de la curva de test, con la mejoría del error en el entrenamiento, lo que es un síntoma de sobre-ajuste y corrobora la impresión de que el sistema está "aprendiendo" solamente características muy particulares del conjunto de entrenamiento. En el siguiente experimento, veremos que esto se vuelve a reproducir para diferentes probabilidades de los operadores genéticos reproducción, mutación y cruce.

Posteriormente, tras aumentar el número de datos por serie temporal y aplicar el algoritmo, veremos que las curvas de test comienzan a seguir mejor las curvas de entrenamiento.

7.5.3 Análisis de la contribución de los operadores genéticos.

7.5.3.1 Descripción del experimento.

Recordemos que el algoritmo utilizado permite aplicar con diferente probabilidad operadores de reproducción, mutación y cruce. Para estudiar como es la contribución de

cada uno de ellos en el proceso de aprendizaje se realizará un conjunto de tres experimentos compuesto cada uno de ellos por 10 pruebas. En cada experimento se configurará el sistema con diferentes probabilidades para los operadores genéticos.

En el primer experimento, se dará una mayor probabilidad al cruce (experimento C), en el segundo, el operador con ventaja es la mutación (experimento M) y en el tercero, la reproducción (experimento R). En la tabla 7.6 se muestran las probabilidades de cada uno de estos operadores en los experimentos.

Los conjuntos de entrenamiento y test son los mismos que en el experimento anterior (cada uno está compuesto por 60 series de 15 datos cada una). El parámetro *alfa* se fija en 0.5 (por ser un valor que daba buenos resultados en las pruebas anteriormente presentadas) y el resto de los parámetros tienen en mismo valor que en las anteriores pruebas (ver tabla 7.4).

Todos los procesos de entrenamiento constarán de 50 iteraciones del algoritmo.

	Cruce	Mutación	Reproducción
Experimento C	0.7	0.2	0.1
Experimento M	0.2	0.7	0.1
Experimento R	0.2	0.2	0.6

Tabla 7.6. Probabilidades de los operadores en cada uno de los experimentos.

7.5.3.2 Resumen de los resultados.

Los resultados de las 10 pruebas de los diferentes experimentos se muestran de manera resumida en las tablas 7.7 (error mínimo de entrenamiento), 7.8 (error mínimo de test) y 7.9 (mínimo error de entrenamiento para el mejor error de test).

Error mínimo de entrenamiento												
	Número de prueba											
	1	2	3	4	5	6	7	8	9	10	**Media**	**Desv.**
Exp. C	18%	18%	20%	18%	25%	23%		17%	20%	17%	20%	0.03
Exp. M	18%	18%	18%	17%	17%	18%	18%	15%	18%	18%	18%	0.01
Exp. R	16%	23%	20%	18%	28%	20%	18%	22%	18%	15%	20%	0.04

Tabla 7.7. Error mínimo de entrenamiento.

Error mínimo de test												
	Número de prueba											
	1	2	3	4	5	6	7	8	9	10	**Media**	**Desv.**
Exp. C	32%	35%	43%	4%	37%	37%		38%	37%	35%	37%	0.03
Exp. M	40%	35%	35%	27%	30%	28%	35%	35%	37%	37%	34%	0.04
Exp. R	45%	37%	38%	43%	43%	43.3%	33%	33%	42%	35%	39%	0.04

Tabla 7.8. Error mínimo de test.

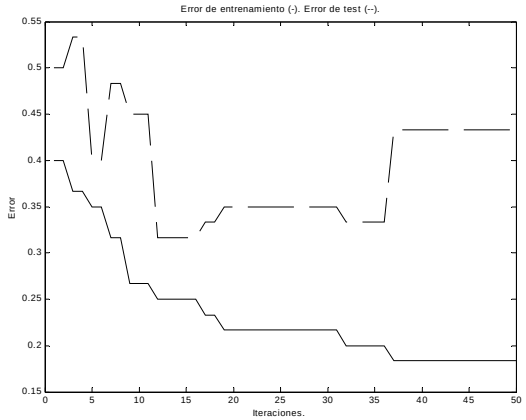
Mínimo error de entrenamiento para el mínimo error de test												
	Número de prueba											
	1	2	3	4	5	6	7	8	9	10	**Media**	**Desv.**
Exp. C	25%	18%	27%	25%	48%	33%		23%	22%	23%	27%	0.09
Exp. M	20%	25%	18%	17%	27%	28%	20%	43%	22%	20%	24%	0.08
Exp. R	17%	23%	27%	40%	43%	20%	37%	30%	20%	20%	28%	0.09

Tabla 7.9. Mínimo error de entrenamiento para el mejor error de test.

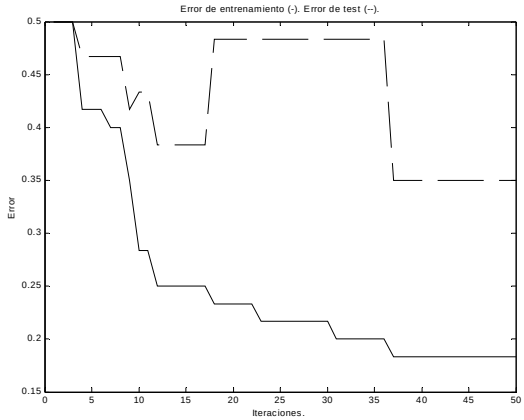
7.5.3.3 Curvas de entrenamiento y test.

Las curvas de entrenamiento y test de las distintas pruebas realizadas en estos experimentos se muestran en las figuras: 7.15 y 7.16 (experimento C), 7.17 y 7.18 (experimento M) y 7.19 y 7.20 (experimento R).

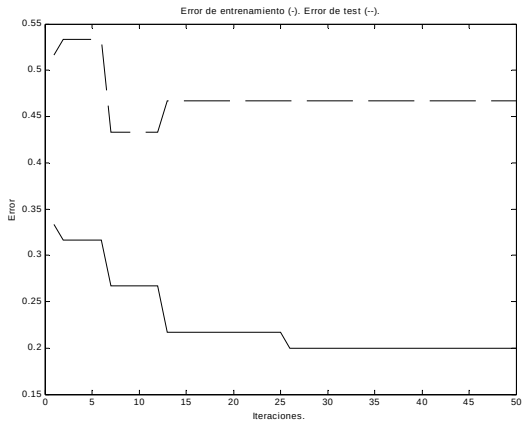
Experimento C.



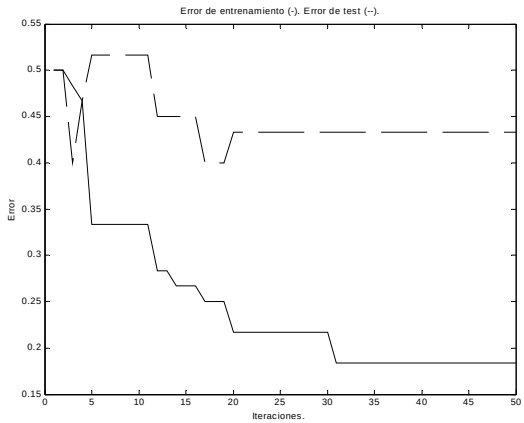
Mínimo error entrenamiento: 18.3%
Mínimo error test: 31.67%
Error en el entr. para el mejor test: 25%



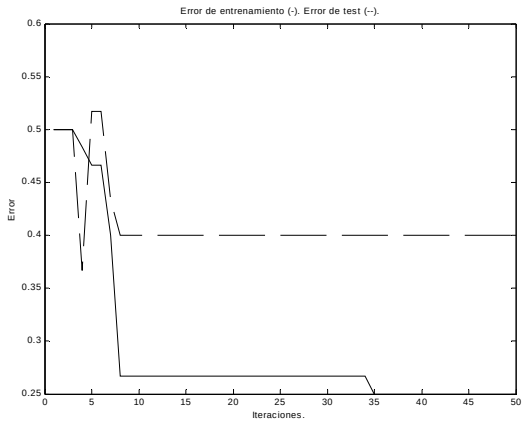
Mínimo error entrenamiento: 18.3%
Mínimo error test: 35%
Error en el entr. para el mejor test: 18.33%



Mínimo error entrenamiento: 20%
Mínimo error test: 43.3%
Error en el entr. para el mejor test: 26.67%

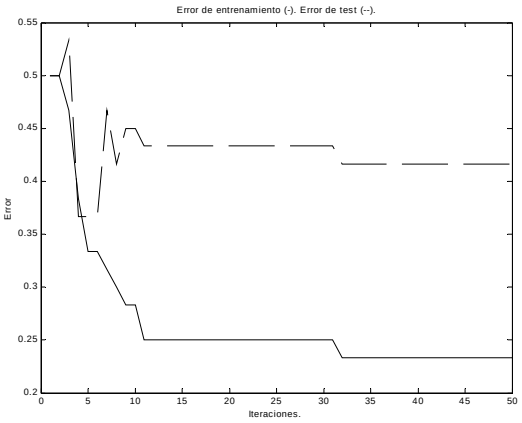


Mínimo error entrenamiento: 18.3%
Mínimo error test: 40%
Error en el entr. para el mejor test: 25%

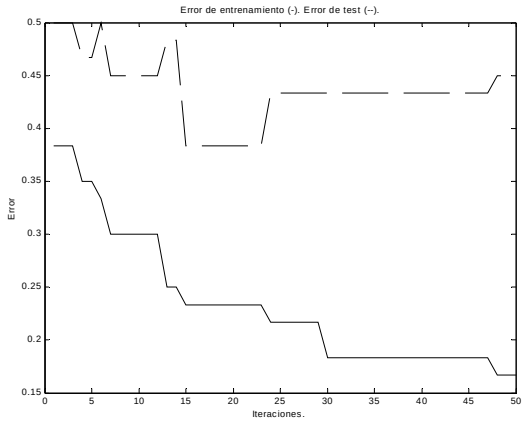


Mínimo error entrenamiento: 25%
Mínimo error test: 36.67%
Error en el entr. para el mejor test: 48.33%

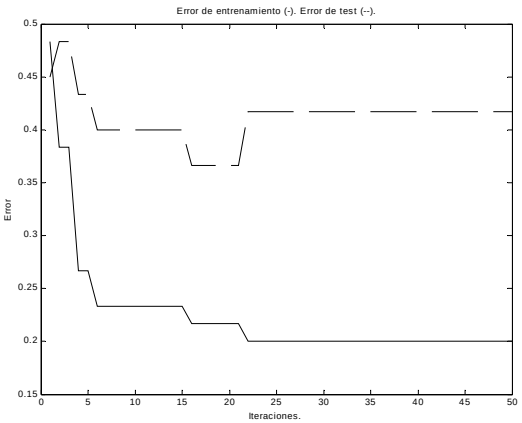
Figura 7.15. Curvas de aprendizaje para el experimento C (1).



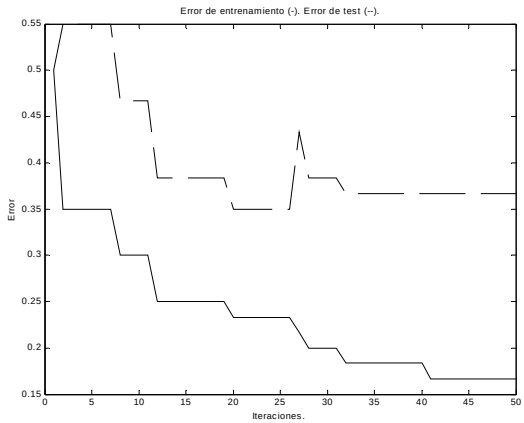
Mínimo error entrenamiento: 23%
Mínimo error test: 36.67%
Error en el entr. para el mejor test: 33.33%



Mínimo error entrenamiento: 16.67%
Mínimo error test: 38.33%
Error en el entr. para el mejor test: 23.33%



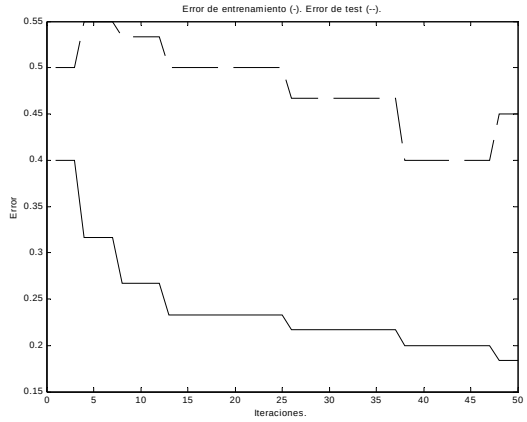
Mínimo error entrenamiento: 20%
Mínimo error test: 36.67%
Error en el entr. para el mejor test: 21.67%



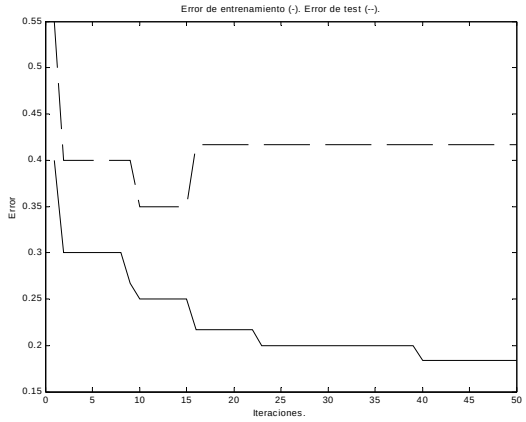
Mínimo error entrenamiento: 16.7%
Mínimo error test: 35%
Error en el entr. para el mejor test: 23.33%

Figura 7.16. Curvas de aprendizaje para el experimento C (2).

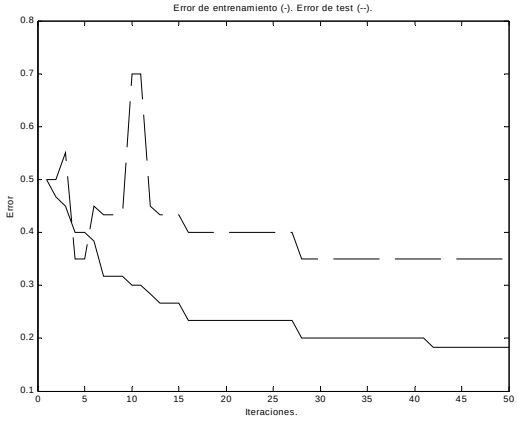
Experimento M.



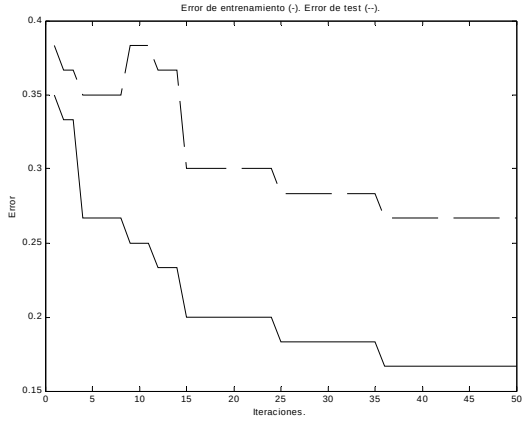
Mínimo error entrenamiento: 18.3%
Mínimo error test: 40%
Error en el entr. para el mejor test: 20%



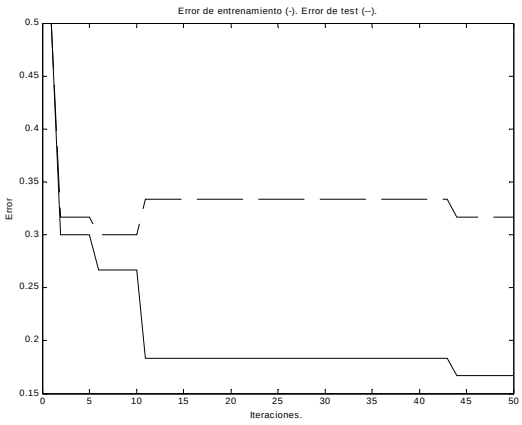
Mínimo error entrenamiento: 18.3%
Mínimo error test: 35%
Error en el entr. para el mejor test: 25%



Mínimo error entrenamiento: 18.3%
Mínimo error test: 35%
Error en el entr. para el mejor test: 18.33%

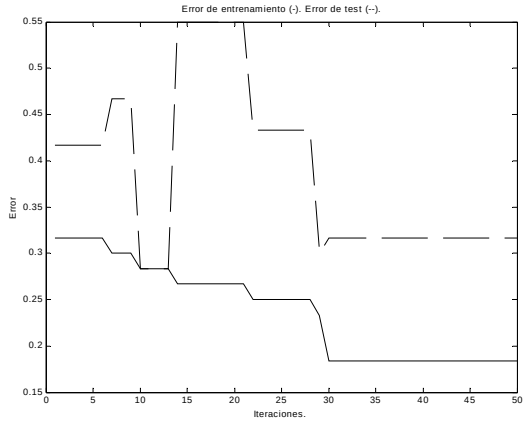


Mínimo error entrenamiento: 16.7%
Mínimo error test: 26.67%
Error en el entr. para el mejor test: 16.67%

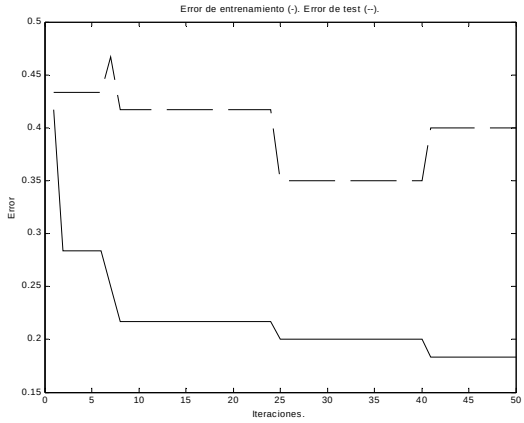


Mínimo error entrenamiento: 16.7%
Mínimo error test: 30%
Error en el entr. para el mejor test: 26.67%

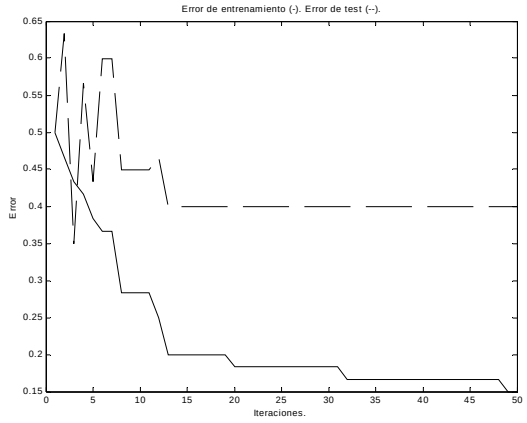
Figura 7.17. Curvas de aprendizaje del experimento M (1).



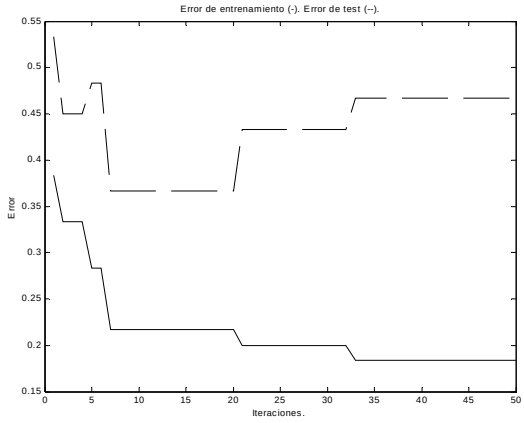
Mínimo error entrenamiento: 18.3%
Mínimo error test: 28.3%
Error en el entr. para el mejor test: 28.3%



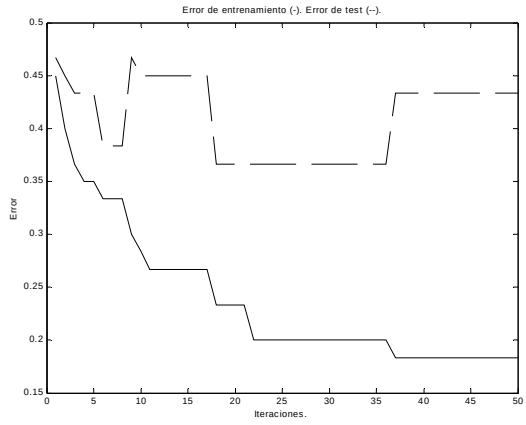
Mínimo error entrenamiento: 18.3%
Mínimo error test: 35%
Error en el entr. para el mejor test: 20%



Mínimo error entrenamiento: 15%
Mínimo error test: 35%
Error en el entr. para el mejor test: 43.33%



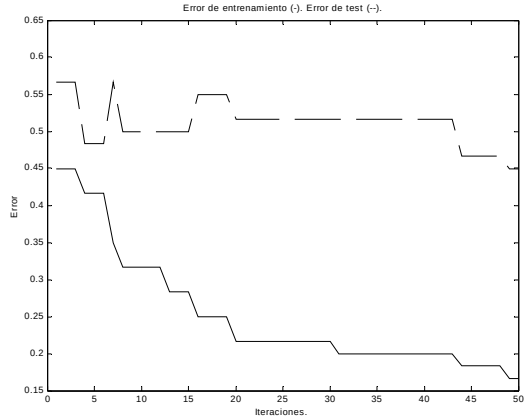
Mínimo error entrenamiento: 20%
Mínimo error test: 36.67%
Error en el entr. para el mejor test: 21.67%



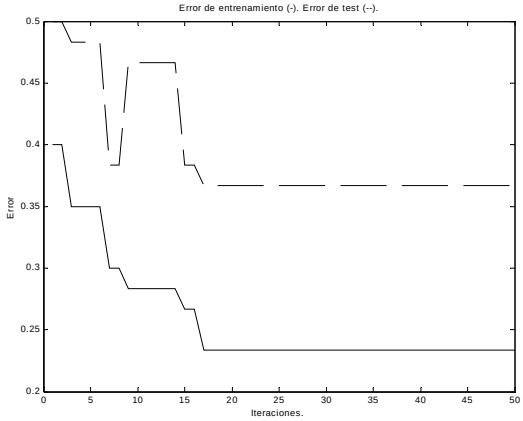
Mínimo error entrenamiento: 18.3%
Mínimo error test: 36.67%
Error en el entr. Para el mejor test: 20%

Figura 7.18. Curvas de aprendizaje del experimento M (2).

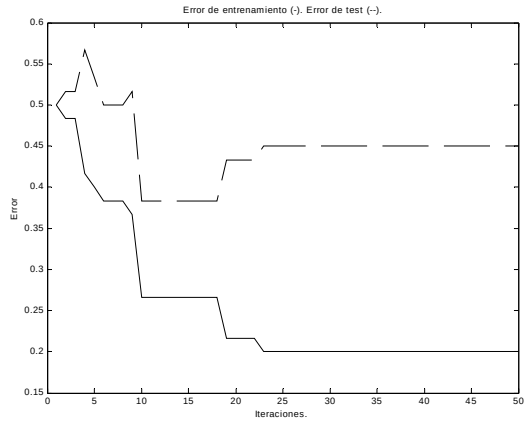
Experimento R.



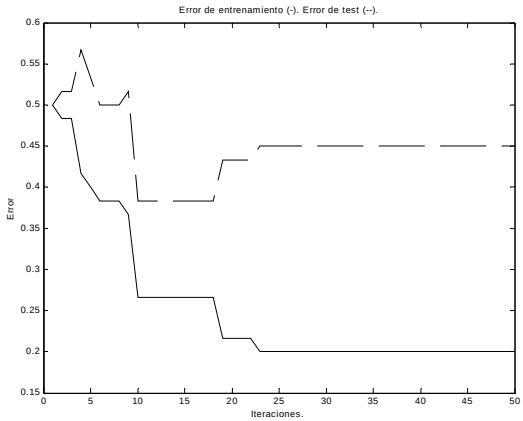
Mínimo error entrenamiento: 16.6%
Mínimo error test: 45%
Error en el entr. para el mejor test: 16.67%



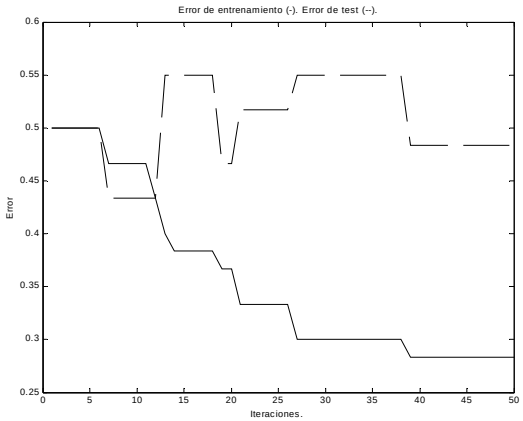
Mínimo error entrenamiento: 23.3%
Mínimo error test: 36.67%
Error en el entr. para el mejor test: 23.3%



Mínimo error entrenamiento: 20%
Mínimo error test: 38.3%
Error en el entr. para el mejor test: 0.2667

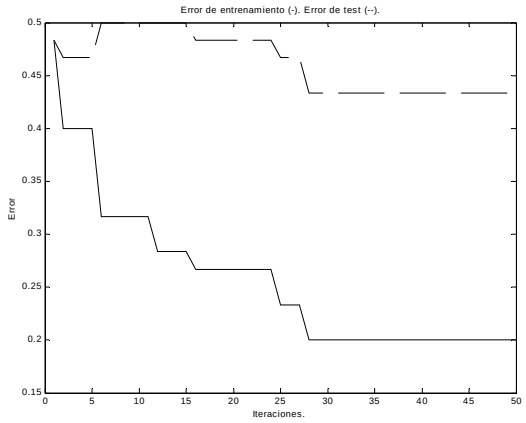


Mínimo error entrenamiento: 18.3%
Mínimo error test: 43.33%
Error en el entr. para el mejor test: 0.4000

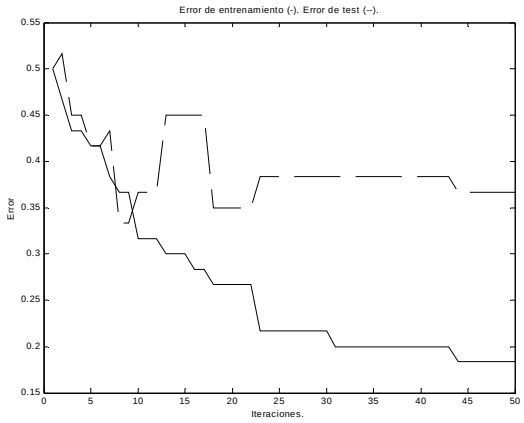


Mínimo error entrenamiento: 28%
Mínimo error test: 43.3%
Error en el entr. para el mejor test: 43.33%

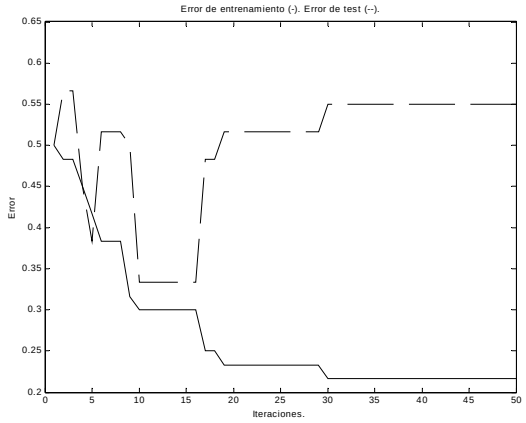
Figura 7.19. Curvas de aprendizaje para el experimento R (1).



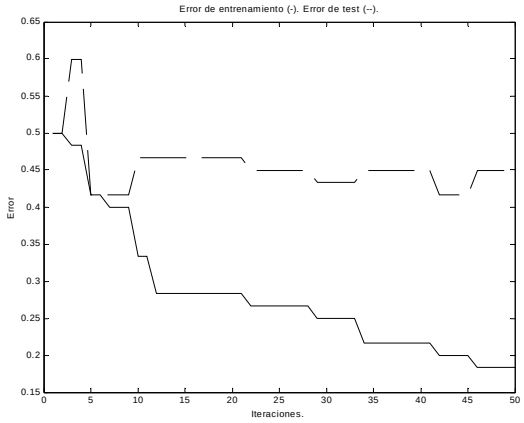
Mínimo error entrenamiento: 20%
Mínimo error test: 43.3%
Error en el entr. para el mejor test: 20%



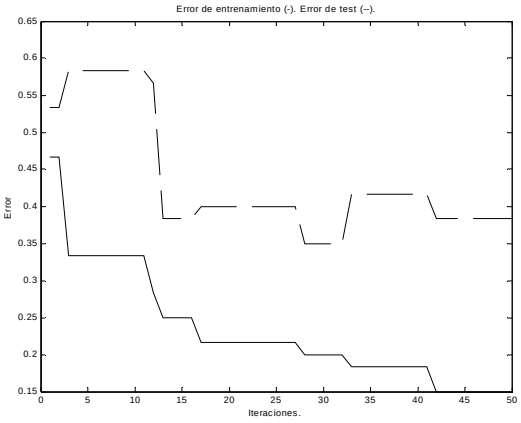
Mínimo error entrenamiento: 18%
Mínimo error test: 33.3%
Error en el entr. para el mejor test: 36.67%



Mínimo error entrenamiento: 21.7%
Mínimo error test: 33.3%
Error en el entr. para el mejor test: 30%



Mínimo error entrenamiento: 18.3%
Mínimo error test: 41.67%
Error en el entr. para el mejor test: 20%



Mínimo error entrenamiento: 15%
Mínimo error test: 35%
Error en el entr. para el mejor test: 20%

Figura 7.20. Curvas de aprendizaje para el experimento R (2).

7.5.3.4 Discusión de los resultados.

En las tablas resumen 7.7, 7.8 y 7.9 se observa que existe un mejor comportamiento en el entrenamiento y en la generalización para los experimentos, cuando la probabilidad del operador mutación es mayor. Esto tiene como efecto que durante el entrenamiento se introduce una mayor diversidad en la población, explorándose así un mayor volumen en el espacio de búsqueda.

Se aprecia en las curvas de aprendizaje el mismo efecto comentado para los experimentos con el parámetro *alfa*. La curva de test no sigue a la curva de entrenamiento. En algunos casos el seguimiento comienza a realizarse, pero en seguida la curva de test comienza a separarse de la de entrenamiento. Esto puede deberse, como ya se comentó en la discusión de los resultados del experimento anterior, al número pequeño de datos en cada serie (15 datos por secuencia), que puede resultar insuficiente para que el sistema aprenda el modelo subyacente a los datos.

7.5.4 Estudio del sistema Pittsburgh en relación al número de muestras en la serie temporal.

7.5.4.1 Descripción del experimento.

En estos experimentos se realizaron tres entrenamientos del sistema Pittsburgh, para series de datos simuladas con los modelos ocultos de Markov descritos al principio. Esta vez se realizan los experimentos para 30 muestras, 45 muestras, 75 muestras y 100 muestras por serie. El objetivo es contrastar si hay una mejoría en la capacidad de generalización del sistema, como ocurrió de hecho con el sistema tipo Michigan (experimento que se expondrá en secciones futuras).

7.5.4.2 Resumen de resultados.

En la tabla 7.10 se muestra un resumen de resultados obtenidos en las distintas pruebas con sistemas Pittsburgh y una comparativa con los obtenidos mediante la aplicación del algoritmo de Baum-Welch. Un análisis similar a éste, se presentará después para el sistema tipo Michigan. Los modelos mencionados en la tabla se corresponden a:

- HMM 1: 30 muestras /secuencia.

- HMM 2: 45 muestras /secuencia.
- HMM 3: 70 muestras /secuencia.
- HMM 4: 100 muestras /secuencia.

En la figura 7.21 se representa el error de entrenamiento y de test de las FFSMs encontradas con los sistemas Pittsburgh y del algoritmo de Baum-Welch.

	Error en el conjunto de entr. con Baum-Welch	Error en el conjunto de test Baum-Welch	Media del error en el entr. de las máquinas Pittsburgh	Desviación del error en el entr.	Media del error en el test de las máquinas Pittsburgh	Desviación del error en el test
HMM 1	20 %	23.33 %	17.78%	2.55%	30%	12.02%
HMM 2	23.33 %	26.67 %	13.89%	3.47%	19.44%	1.92%
HMM 3	23.33 %	25 %	12.22%	4.19%	21.67%	8.82%
HMM 4	11.67 %	8.33 %	8.33%	0%	14.44%	4.81%

Tabla 7.10. Resultados comparativos entre el algoritmo basado en la identificación de Baum-Welch y Pittsburgh para diferentes longitudes de las series temporales.

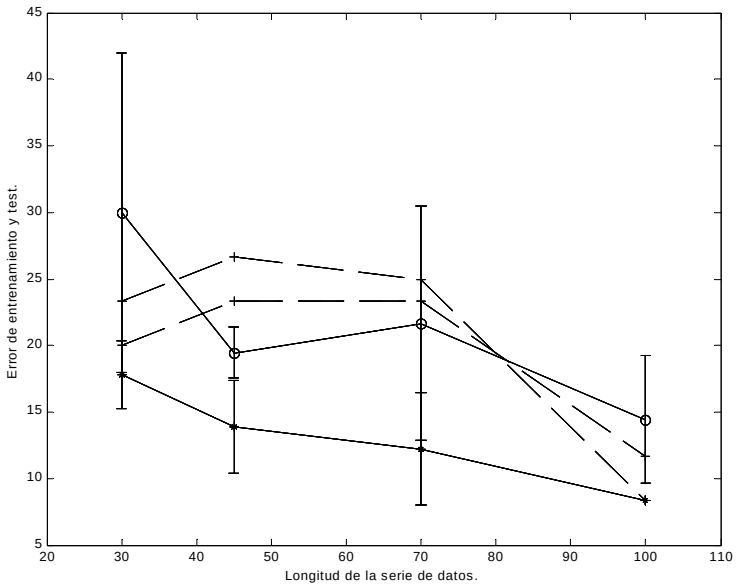
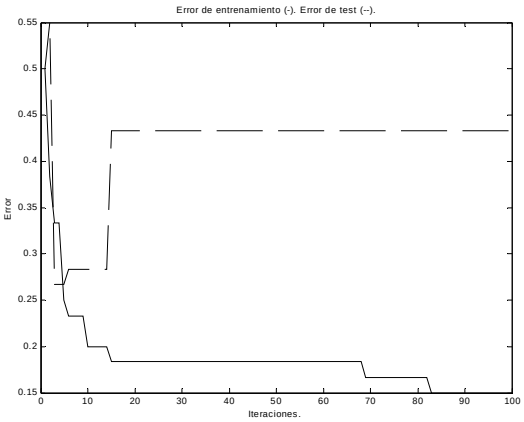


Figura 7.21. Error de entrenamiento (*) y test (o) para diferentes longitudes de las series de datos. En trazo discontinuo, curvas de entrenamiento y test obtenidas mediante identificación de Baum-Welch y evaluación de $P(O \mid \lambda)$.

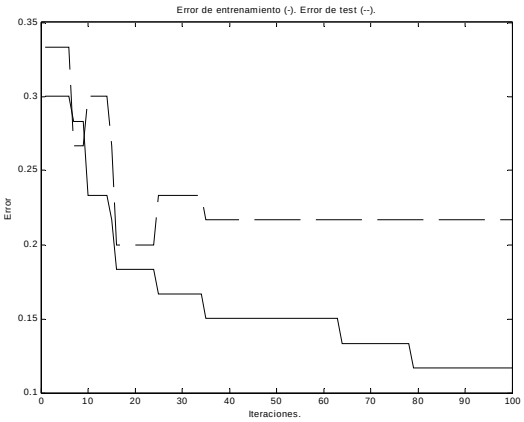
7.5.4.3 Curvas de entrenamiento y test.

Las curvas de entrenamiento y test de las distintas pruebas realizadas en estos experimentos se muestran en las figuras: 7.22 (30 muestras por serie), 7.23 (45 muestrasm por serie), 7.24 (70 muestras por serie) y 7.25 (100 muestras por serie).

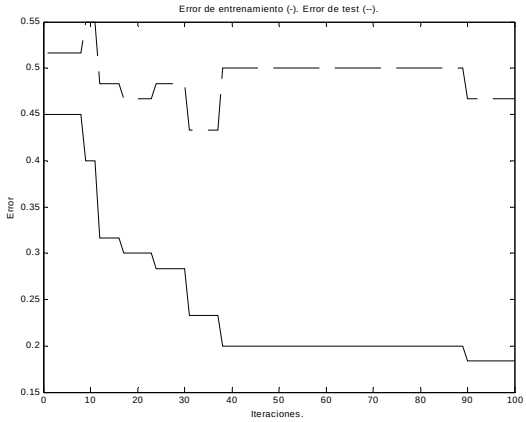
Pruebas con 30 muestras por serie.



Mínimo error de test: 26.67%
Error de entrenamiento asociado al test: 25%
Mínimo error de entrenamiento: 18.33%



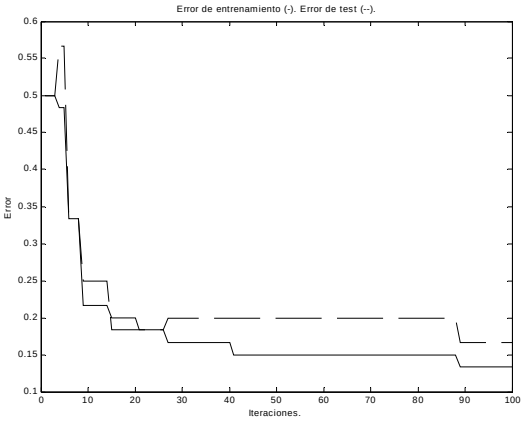
Mínimo error de test: 20%
Error de entrenamiento asociado al test: 18.33%
Mínimo error de entrenamiento: 15%



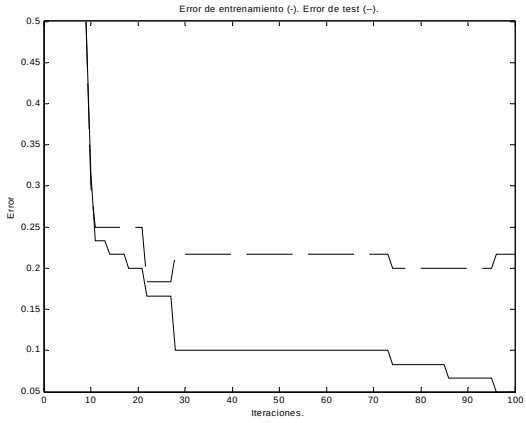
Mínimo error de test: 43.33%
Error de entrenamiento asociado al test: 23.33%
Mínimo error de entrenamiento: 20%

Figura 7.22. Curvas de aprendizaje con series de datos de 30 muestras.

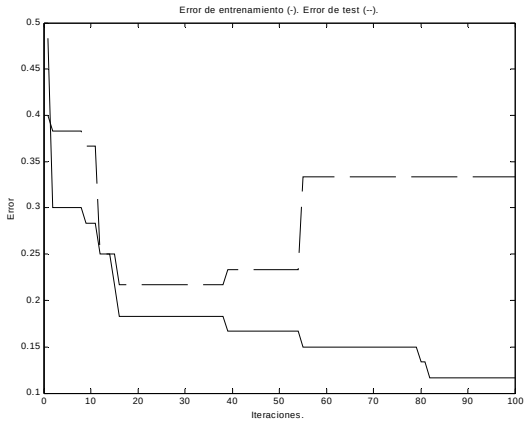
Pruebas con 45 muestras por serie.



Mínimo error de test: 18.33%
Error de entrenamiento asociado al test: 18.33%
Mínimo error de entrenamiento: 15%



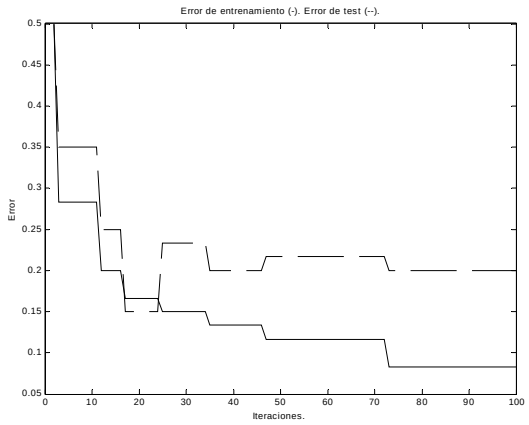
Mínimo error de test: 18.33%
Error de entrenamiento asociado al test: 16.67%
Mínimo error de entrenamiento: 10%



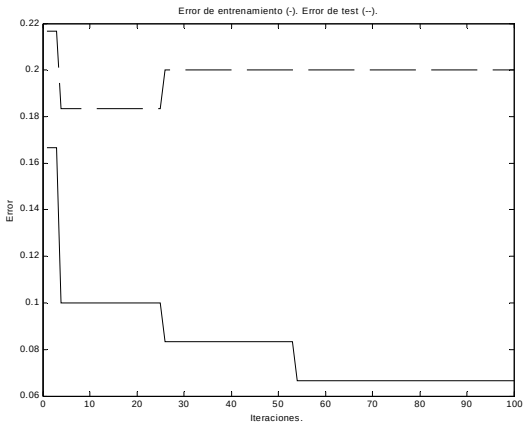
Mínimo error de test: 21.67%
Error de entrenamiento asociado al test: 18.33%
Mínimo error de entrenamiento: 16.67%

Figura 7.23. Curvas de aprendizaje con series de datos de 45 muestras.

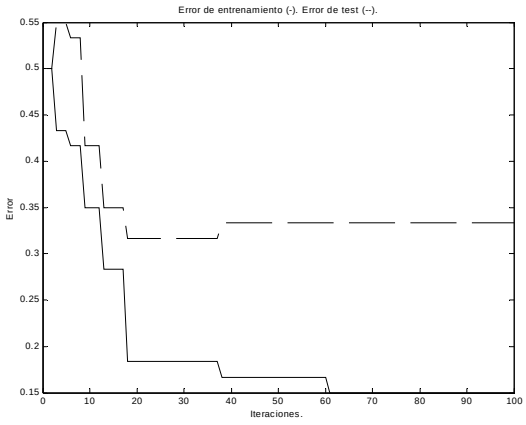
Pruebas con 70 muestras por serie.



Mínimo error de test: 15%
Error de entrenamiento asociado al test: 16.67%
Mínimo error de entrenamiento: 11.67%



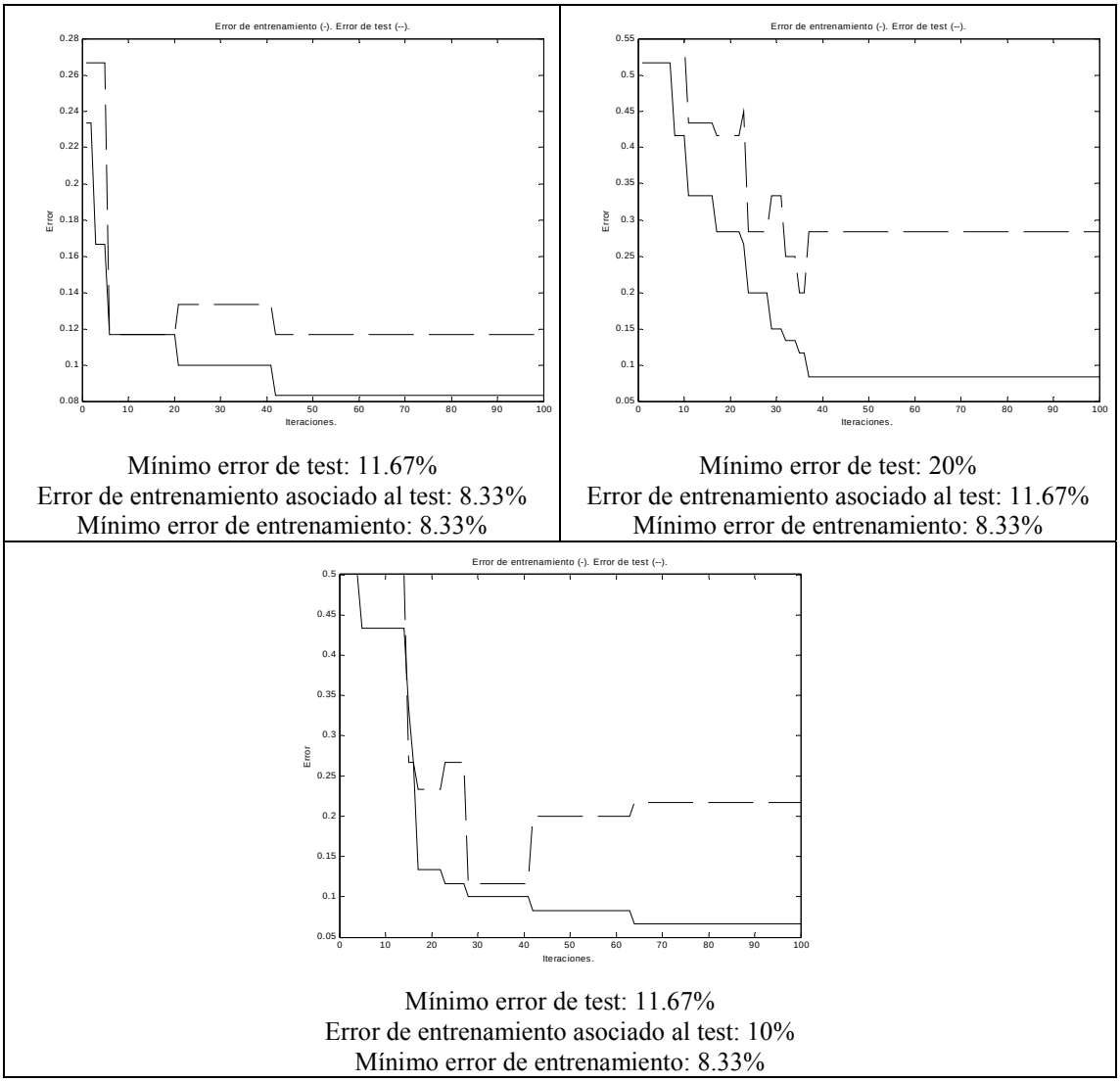
Mínimo error de test: 18.33%
Error de entrenamiento asociado al test: 10%
Mínimo error de entrenamiento: 8.33%



Mínimo error de test: 31.67%
Error de entrenamiento asociado al test: 18.33%
Mínimo error de entrenamiento: 16.67%

Figura 7.24. Curvas de aprendizaje con series de datos de 70 muestras.

Pruebas con 100 muestras por serie.



Mínimo error de test: 11.67%
Error de entrenamiento asociado al test: 8.33%
Mínimo error de entrenamiento: 8.33%

Mínimo error de test: 20%
Error de entrenamiento asociado al test: 11.67%
Mínimo error de entrenamiento: 8.33%

Mínimo error de test: 11.67%
Error de entrenamiento asociado al test: 10%
Mínimo error de entrenamiento: 8.33%

Figura 7.25. Curvas de aprendizaje con series de datos de 100 muestras.

7.5.4.4 Discusión de los resultados.

Se observa en primer lugar que el aumento del número de muestras por serie temporal tiene un efecto en la reducción de la media del error tanto en el entrenamiento como en el test de la clasificación basada en la máquina de estados borrosa diseñada mediante el enfoque de Pittsburgh. La disminución del error en el entrenamiento con el sistema tipo Pittsburgh al pasar de 30 a 100 muestras por serie temporal es de un 46.8% mientras que en el caso del algoritmo basado en identificación por Baum-Welch, esta reducción fue del 58.35%. Sin embargo, también se observa que en los experimentos realizados el error de entrenamiento de las máquinas obtenidas mediante el sistema Pittsburgh, es

menor que en el caso de Baum-Welch. Si bien, también es cierto que la tendencia en Baum-Welch es a disminuir este error de forma más rápida en relación al número de muestras por serie temporal que en el caso de Pittsburgh, con lo que para un número de muestras por serie suficientemente grande, el algoritmo basado en la identificación del modelo supera a la máquina de estados borrosa.

Por otra parte, las medias del error de test en el caso de las máquinas de estado borrosas en dos de los experimentos (45 y 70 muestras por serie temporal) son mejores que en la clasificación realizada usando la identificación del modelo, especialmente en el caso de 45 muestras donde la desviación del error de test es relativamente pequeña. Sin embargo, no se puede afirmar que las máquinas de estado borrosas obtenidas mediante el sistema Pittsburgh sean una opción en general ventajosa respecto a la técnica basada en la identificación del modelo oculto de Markov. Los datos obtenidos apuntan en la dirección de que los sistemas borrosos recurrentes así diseñados tendrían ventajas respecto de la técnica basada en la identificación para un número medio de muestras por serie temporal. Para un número excesivamente pequeño ambas técnicas darían pobres resultados de generalización y para un número elevado de muestras por serie temporal la técnica basada en la identificación del modelo daría los mejores resultados.

Hay que tener en cuenta a la hora de analizar estos resultados, que los datos se han obtenido para 50 iteraciones del sistema evolutivo de Pittsburgh y unos parámetros concretos en el algoritmo que han tenido que ser ajustados previamente. También juega en contra del sistema Pittsburgh su baja eficiencia computacional en relación a la técnica basada en la identificación del modelo basada en Baum-Welch.

Sin embargo, se debe recordar que el objetivo de este experimento es el de demostrar que una máquina de estados borrosa entrenada mediante una técnica tipo Pittsburgh es capaz de aprender el modelo subyacente en un proceso estocástico de tipo Markov, como queda patente del hecho observado de la reducción del error de test al incrementar el número de muestras por serie temporal (una media de un 30% para 30 muestras por serie temporal y una media del 14.44% para 100 muestras por serie temporal).

7.6 Estudio de un sistema Michigan en la clasificación de series de datos producidas por un proceso de Markov.

7.6.1 Introducción.

Estos son los aspectos que deseamos investigar acerca de las máquinas de estado borrosas diseñadas bajo la arquitectura de tipo Michigan:

- Se desea establecer si realmente con esta arquitectura, la máquina de estados borrosa logra mejorar el proceso de aprendizaje respecto a una búsqueda aleatoria simple.

- Se pretende investigar la influencia del parámetro específico relativo a la frecuencia de disparo del algoritmo genético. Este es un parámetro como veremos de suma importancia para la eficiencia del aprendizaje.

- Se desea demostrar la capacidad de generalización de la máquina de estados borrosa obtenida en la arquitectura de tipo Michigan. Para ello se examinará de forma comparativa con el algoritmo basado en la identificación de Baum-Welch y la evaluación $P(O \mid \lambda)$, que es específico para el problema propuesto y cuya capacidad de generalización en este contexto está demostrada.

Se comenzará realizando una prueba preliminar para analizar la capacidad del algoritmo de búsqueda para encontrar una máquina de estados borrosa que al menos se ajuste a los datos de entrenamiento.

Para estudiar el efecto de los parámetros en el entrenamiento del modelo se usarán series de datos de longitud no muy elevada, ya que el objetivo es realizar un primer ajuste del parámetro relativo a la frecuencia de disparo del algoritmo genético en el sistema tipo Michigan. En relación a este parámetro se estudiarán dos estrategias alternativas que regulan el arranque del algoritmo genético, la primera, de tipo indirecto, basada en la antigüedad de las meta-reglas del conjunto de encaje (en analogía al

algoritmo XCS) y la segunda, más directa, estableciendo una frecuencia fija para el arranque del algoritmo genético.

Una vez estudiada la influencia de este parámetro se compararán los resultados con búsquedas aleatorias simples sobre el espacio de las máquinas de estado, para tener una segunda medida comparativa de la situación relativa del algoritmo propuesto.

Finalmente, se obtendrán datos con la máquina de estados borrosa entrenada con series de datos de mayor longitud, para ver así como mejora la capacidad de generalización de la máquina de estados borrosa

7.6.2 Experimento preliminar con la máquina de estados borrosa en la arquitectura de tipo Michigan.

7.6.2.1 Descripción del experimento.

En este caso se pretende obtener una primera evaluación de la capacidad del sistema de tipo Michigan para clasificar las series de datos producidas con modelos de Markov. Además, en esta primera fase se realizó una sintonización inicial de los parámetros del algoritmo para lograr un ajuste al conjunto de entrenamiento similar al obtenido mediante el método de Baum-Welch.

Se generan 60 secuencias con 15 muestras por secuencia para el modelo 1 y 60 secuencias con 15 muestras por secuencia para el modelo 2.

Se construye el conjunto de entrenamiento con 30 secuencias del modelo 1 y 30 secuencias del modelo 2 (en total, 60 secuencias, 30 de cada modelo) y el conjunto de test con las otras 30 secuencias no utilizadas del modelo 1 y las otras 30 secuencias no utilizadas del modelo 2 (en total, 60 secuencias, 30 de cada modelo).

Se realizarán tres entrenamientos con el sistema tipo Michigan para obtener tres máquinas de estado borrosas. En la tabla 7.11 se muestran los valores utilizados para los parámetros del sistema tipo Michigan.

Parámetro	Abreviatura	Valor
Número de reglas de la máquina	*num_reglas*	10
Número de estados de la máquina	*num_estados*	4
Número de subintervalos en el rango [0,1] para la asignación del consecuente	*num_puntos*	6
Número del estado de detección	*num_detec*	4
Umbral para la determinación del parámetro de reactividad.	*param_alta*	0.6
Valor de fuerza inicial de las meta-reglas.	*fuerza_inicial*	5
Valor de experiencia inicial de las meta-reglas.	*experiencia_inicial*	0
Probabilidad de usar un comodín en la generación de meta-reglas en el recubrimiento.	*p_cov*	0.4
Número de encajes mínimo en el conjunto de encajes. Si la población no alcanza este mínimo en la formación del conjunto de encaje se recurrirá al recubrimiento.	*num_encajes*	50
Factor de recompensa en el proceso de asignación de créditos.	*factor_recompensa*	50
Penalización en el proceso de asignación de créditos	*penalización*	0.01
Umbral de antigüedad que debe superar una porción de las meta-reglas del conjunto de encaje para que se active el algoritmo genético	*umbral_antigüedad*	20
Factor de solape en las poblaciones de las meta-reglas	*alfa*	0.5
Probabilidad del operador de reproducción.	*p1*	0.05
Probabilidad del operador de mutación.	*p2*	0.65
Probabilidad del operador de cruce.	*p3*	0.35
Probabilidad de mutar cada elemento de la meta-regla.	*P_mut*	0.6
Probabilidad utilizada para determinar si el resultado de la mutación va a ser un comodín o no.	*P_mut2*	0.5
Número máximo de meta-reglas a borrar.	*N*	50
Factor multiplicativo sobre la fuerza media del conjunto de encaje para establecer el umbral de fuerza mínima en las meta-regla con suficiente antigüedad para sobrevivir al proceso de borrado.	*factor_borrado*	0.001
Umbral de experiencia (utilización de la meta-regla) que debe superar una meta-regla para poder ser borrada.	*umbral_experiencia*	3

Tabla 7.11. Parámetros del sistema tipo Michigan.

7.6.2.2 Resumen de resultados.

A continuación se muestran los resultados de tres entrenamientos con estas características: error de la máquina final en la clasificación del conjunto de entrenamiento y en la clasificación del conjunto de test (tabla 7.12).

	Error de entrenamiento	Error de test
Experimento 1	10%	43.44%
Experimento 2	15%	25%
Experimento 3	8.33%	46.67%

Tabla 7.12. Resultados preliminares con el sistema Michigan.

7.6.2.3 Discusión de los resultados.

En los tres casos se consiguen errores de entrenamiento similares a los obtenidos con Baum-Welch. Sin embargo, los errores de test, son bastante peores, aunque el segundo de los entrenamientos presenta una curva de test que sigue a la curva de entrenamiento y llega a un error de test del 25%, lo cual es positivo. Como veremos más adelante la capacidad de generalización de las máquinas de estado borrosas mejorará notablemente al aumentar el número de muestras en cada serie temporal. De hecho, podemos concluir que con 15 muestras por serie temporal, el sistema tipo Michigan tiende a aprender características particulares de los datos de entrenamiento en lugar del modelo subyacente.

Se ha observado además que el sistema tipo Michigan tiene una gran ventaja frente al método Pittsburgh: las máquinas Michigan se entrenan de una forma más eficiente, es decir, aunque los procesos de entrenamiento Michigan necesiten de más iteraciones del algoritmo que los de Pittsburgh, el tiempo de cómputo es menor. Esto se debe a que en el método Pittsburgh se evalúa una población de 200 máquinas en cada iteración, mientras que en el método Michigan sólo se evalúa una máquina por iteración.

La eficiencia de las máquinas obtenidas por el método Michigan es mejorable, por lo que se debe investigar mejor la influencia de los parámetros en este algoritmo. Dado que se observan períodos largos de estancamiento en el algoritmo, y esto significa que hay poca renovación en la población de reglas, se intuye que un parámetro importante es la frecuencia con la que se disparan los algoritmos genéticos en el algoritmo Michigan. En las siguientes pruebas se realiza un estudio detallado de la influencia de este parámetro sobre el comportamiento del algoritmo global.

7.6.3 Primer estudio de la influencia en el sistema tipo Michigan de la frecuencia de disparo del algoritmo genético.

7.6.3.1 Descripción del experimento.

El algoritmo genético es el método de descubrimiento en el proceso de búsqueda implementado en el sistema tipo Michigan. La frecuencia con la que se activa este proceso sobre la base de meta-reglas es un parámetro importante, ya que una frecuencia demasiado baja provocará un retraso en la exploración de nuevas soluciones, mientras que una frecuencia excesivamente alta impide la adecuada renovación de las recompensas sobre las meta-reglas por parte del subsistema de asignación de créditos.

Se han estudiado dos mecanismos con los que se realiza el disparo del algoritmo genético. En el primero de ellos se introduce un nuevo parámetro (*porción*) que indica el porcentaje de elementos (reglas) que encajan en una determinada iteración que debe sobrepasar un umbral de antigüedad determinado para que se arranque un algoritmo genético. En el segundo se utiliza una frecuencia de disparo prefijada y constante.

Se analizarán 6 valores diferentes del parámetro *porción*, realizándose 4 procesos de entrenamiento para cada uno de ellos. En todas estas pruebas se ha partido de la misma máquina inicial y se ha ejecutado el algoritmo durante el mismo número de iteraciones (600 iteraciones). Los entrenamientos utilizan los parámetros de la tabla 7.11.

Se generan 60 secuencias con 15 muestras por secuencia para el modelo 1 y 60 secuencias con 15 muestras por secuencia para el modelo 2. Se construye el conjunto de entrenamiento con 30 secuencias del modelo 1 y 30 secuencias del modelo 2 (en total, 60 secuencias, 30 de cada modelo) y el conjunto de test con las otras 30 secuencias no utilizadas del modelo 1 y las otras 30 secuencias no utilizadas del modelo 2 (en total, 60 secuencias, 30 de cada modelo).

7.6.3.2 Resumen de resultados.

En la tabla 7.13 se muestran de forma compacta los resultados obtenidos en estos experimentos.

Valor de *porción*	Entrenamientos	Error entrenamiento	Error test	Número de GAs disparados
0.5	Entrenamiento 1	23.33%	41.67 %	52
	Entrenamiento 2	23.33%	33 %	53
	Entrenamiento 3	20%	30%	51
	Entrenamiento 4	16.67 %	40 %	49
	RESUMEN	Media: 20.83 % Desviación: 3.18	Media: 36.25% Desviación: 5.51	Media: 51.25 Desviación: 1.71
0.6	Entrenamiento 1	16.67 %	50 %	30
	Entrenamiento 2*	23.33 %	41.67 %	154
	Entrenamiento 3	21.67 %	36.67 %	28
	Entrenamiento 4	25 %	46.67 %	28
	RESUMEN (sin *)	Media: 21.11% Desviación: 4.19	Media: 44.45% Desviación: 6.94	Media: 28.67 Desviación: 1.15
0.7	Entrenamiento 1	23.33 %	38.33 %	27
	Entrenamiento 2	16.67 %	45 %	26
	Entrenamiento 3*	25 %	41.67 %	82
	Entrenamiento 4*	21.67 %	35 %	99
	RESUMEN (sin *)	Media: 20% Desviación: 4.71	Media: 41.66 % Desviación: 4.71	Media: 26.5 Desviación: 0.71
0.8	Entrenamiento 1*	23.33 %	41.67 %	24
	Entrenamiento 2**	20 %	43.33 %	0
	Entrenamiento 3*	28.33 %	40 %	27
	Entrenamiento 4**	20 %	43.33 %	0
	RESUMEN (de *)	Media: 28.33 % Desviación: 0 %	Media: 40.83 % Desviación: 1.18	Media: 25.5 Desviación: 2.12
	RESUMEN (de **)	Media: 20 % Desviación: 0 %	Media: 43.33 % Desviación: 0 %	Media: 0 Desviación: 0
0.9	Entrenamiento 1**	18.33 %	36.67 %	0
	Entrenamiento 2	15 %	43.33 %	12
	Entrenamiento 3**	25 %	58.33 %	0
	Entrenamiento 4	25 %	55 %	33
	RESUMEN (de **)	Media: 21.66% Desviación: 4.71	Media: 47.5 % Desviación: 15.31	Media: 0 Desviación: 0
1	Entrenamiento 1	30 %	30%	0
	Entrenamiento 2	20 %	33.33%	0
	Entrenamiento 3	21.67 %	48.33 %	0
	Entrenamiento 4	25 %	40 %	0
	RESUMEN	Media: 24.16 % Desviación: 4.4	Media: 37.91 % Desviación: 8.09	Media: 0 Desviación: 0

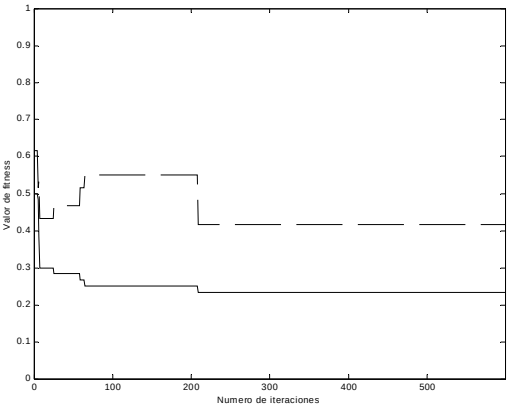
Tabla 7.13. Resultados en las primeras 600 iteraciones con diferentes valores del parámetro *porción*.

7.6.3.3 Curvas de entrenamiento y test.

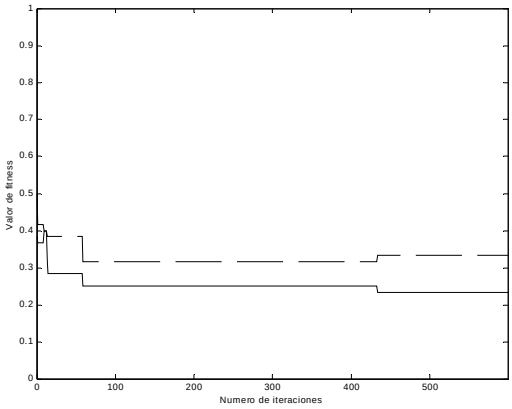
En las figuras 7.26, 7.27, 7.28, 7.29, 7.30 y 7.31 se muestran las curvas de entrenamiento y de test de las distintas pruebas realizadas con los siguientes valores de *porción*: 0.5, 0.6, 0.7, 0.8, 0.9 y 1.

Pruebas con *porción* = 0.5.

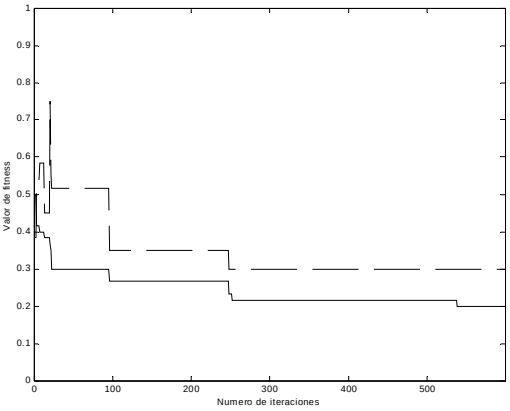
El 50% de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.



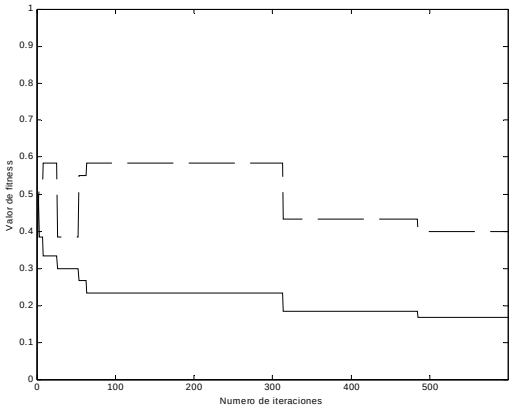
Error en entrenamiento: 23.33 %
Error en test: 41.67 %
Número de algoritmos genéticos: 52



Error en entrenamiento: 23.33 %
Error en test: 33.33 %
Número de algoritmos genéticos: 53



Error en entrenamiento: 20%
Error en test: 30%
Número de algoritmos genéticos: 51

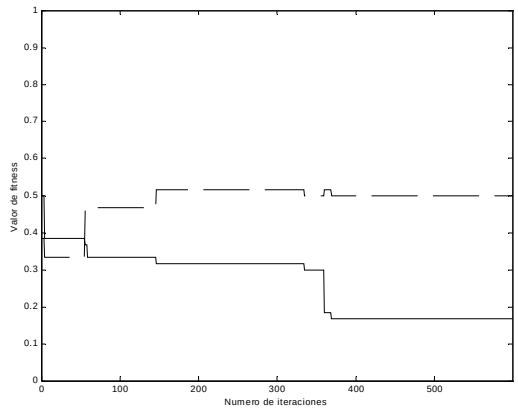


Error de entrenamiento: 16.67%
Error en test: 40%
Número de algoritmos genéticos: 49

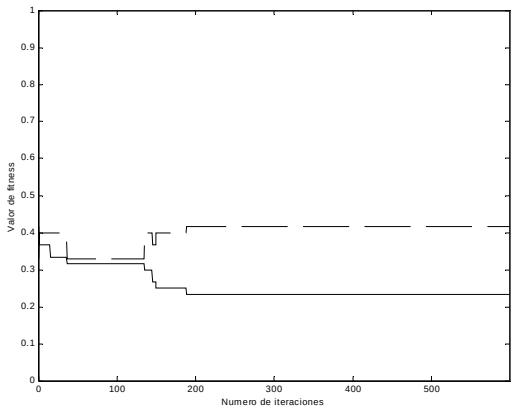
Figura 7.26. Curvas de entrenamiento y test para el experimento con el parámetro *porción* = 0.5.

Pruebas con *porción* = 0.6.

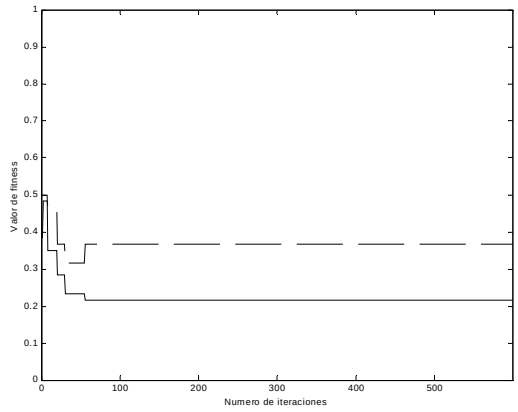
El 60 % de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.



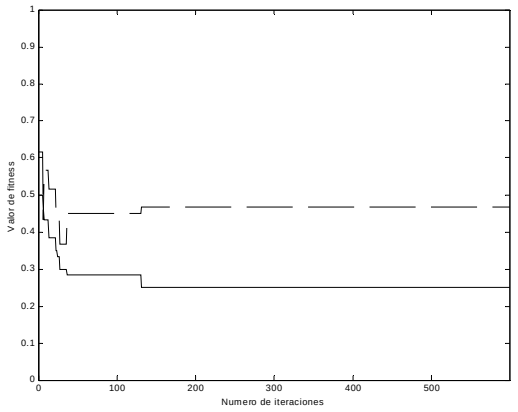
Error en entrenamiento: 16.67 %
Error en test: 50 %
Número de algoritmos genéticos: 30



Error de entrenamiento: 23.33 %
Error en test: 41.67 %
Número de algoritmos genéticos: 154



Error en entrenamiento: 21.67 %
Error en test: 36.67 %
Número de algoritmos genéticos: 28

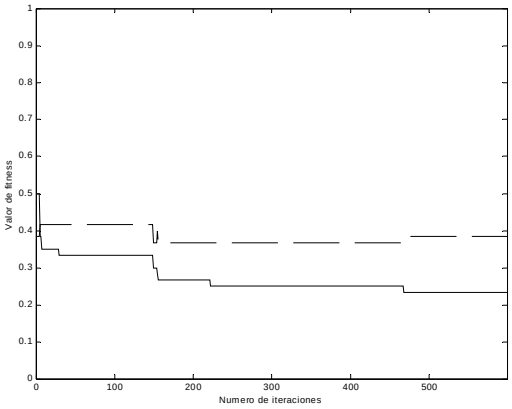


Error en entrenamiento: 25 %
Error en test: 46.67 %
Número de algoritmos genéticos: 28

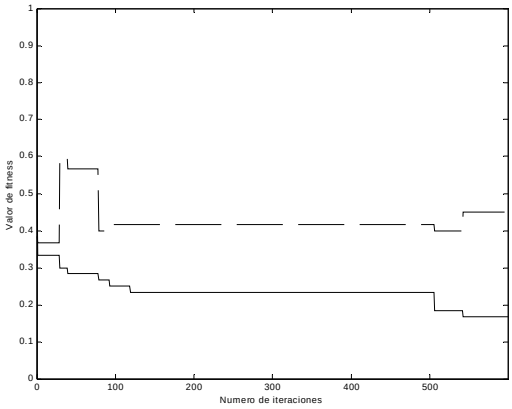
Figura 7.27. Curvas de entrenamiento y test para el parámetro *porción* = 0.6.

Pruebas con *porción* = 0.7.

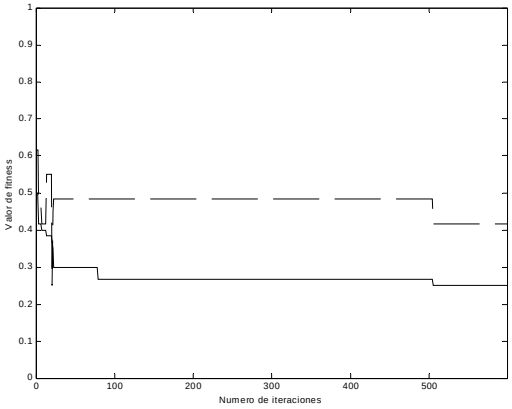
El 70 % de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.



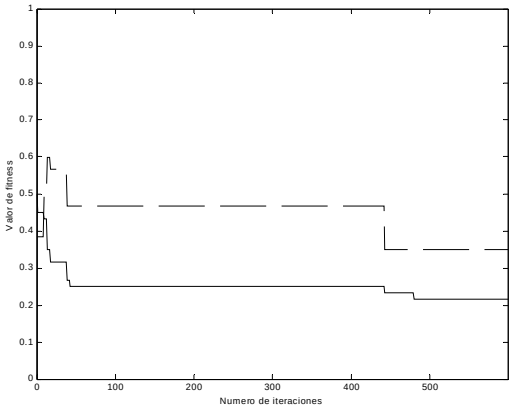
Error en entrenamiento: 23.33 %
Error en test: 38.33 %
Número de algoritmos genéticos: 27



Error en entrenamiento: 16.67 %
Error en test: 45 %
Número de algoritmos genéticos: 26



Error en entrenamiento: 25 %
Error en test: 41.67 %
Número de algoritmos genéticos: 82

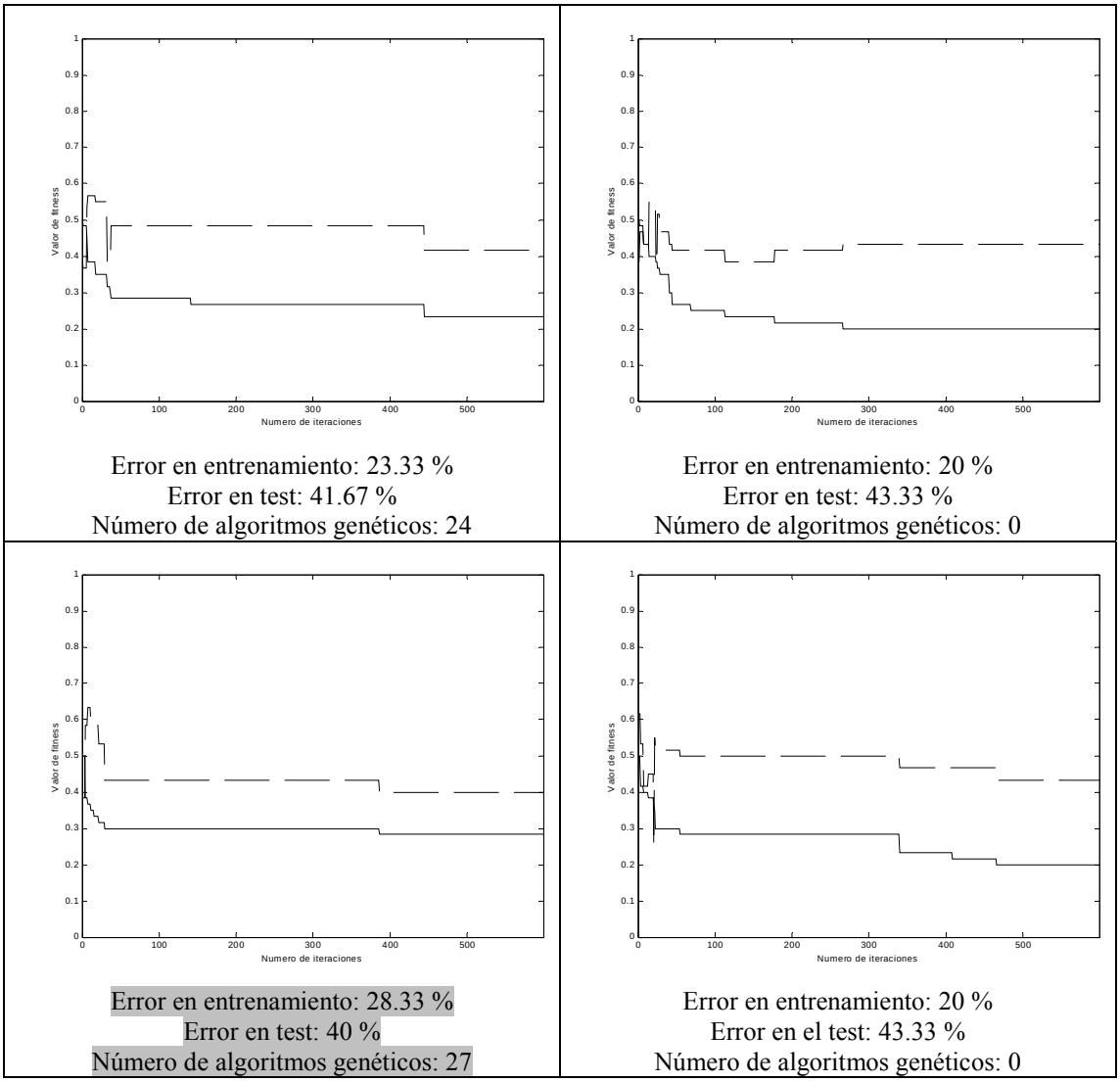


Error en entrenamiento: 21.67 %
Error en test: 35 %
Número de algoritmos genéticos: 99

Figura 7.28. Curvas de entrenamiento y test para el parámetro *porción* = 0.7.

Pruebas con *porción* = 0.8.

El 80 % de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.



Error en entrenamiento: 23.33 %
Error en test: 41.67 %
Número de algoritmos genéticos: 24

Error en entrenamiento: 20 %
Error en test: 43.33 %
Número de algoritmos genéticos: 0

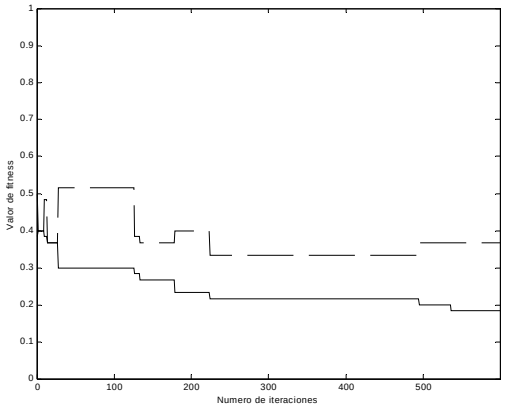
Error en entrenamiento: 28.33 %
Error en test: 40 %
Número de algoritmos genéticos: 27

Error en entrenamiento: 20 %
Error en el test: 43.33 %
Número de algoritmos genéticos: 0

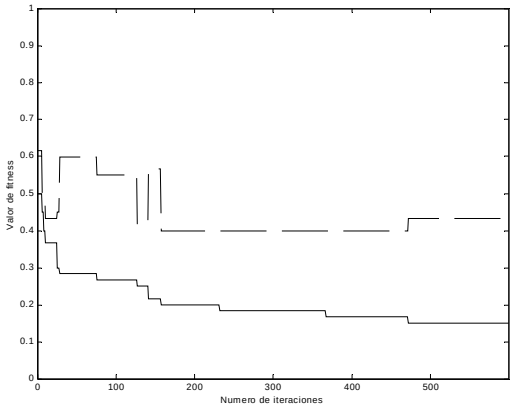
Figura 7.29. Curvas de entrenamiento y test para el parámetro *porción* = 0.8.

Pruebas con *porción* = 0.9.

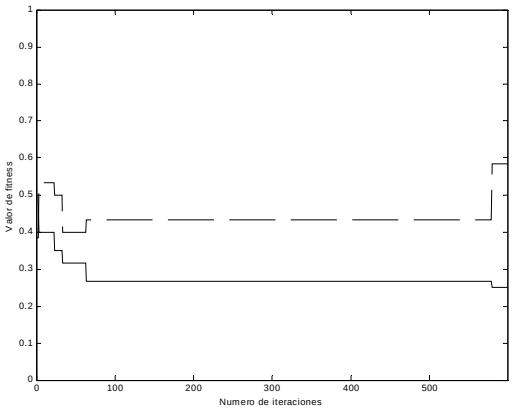
El 90 % de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.



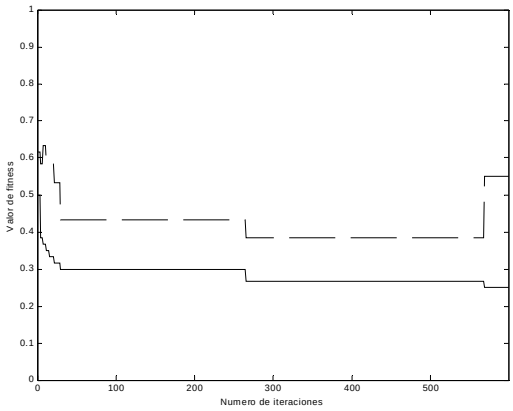
Error en entrenamiento: 18.33 %
Error en test: 36.67 %
Número de algoritmos genéticos: 0



Error en entrenamiento: 15 %
Error en test: 43.33 %
Número de algoritmos genéticos: 12



Error en entrenamiento: 25 %
Error en test: 58.33 %
Número de algoritmos genéticos: 0



Error en entrenamiento: 25 %
Error en test: 55 %
Número de algoritmos genéticos: 33

Figura 7.30. Curvas de entrenamiento y test para el parámetro *porción* = 0.9.

Pruebas con *porción* = 1.

El 100 % de los elementos que encajan deben superar el umbral de antigüedad para que se arranque el algoritmo genético.

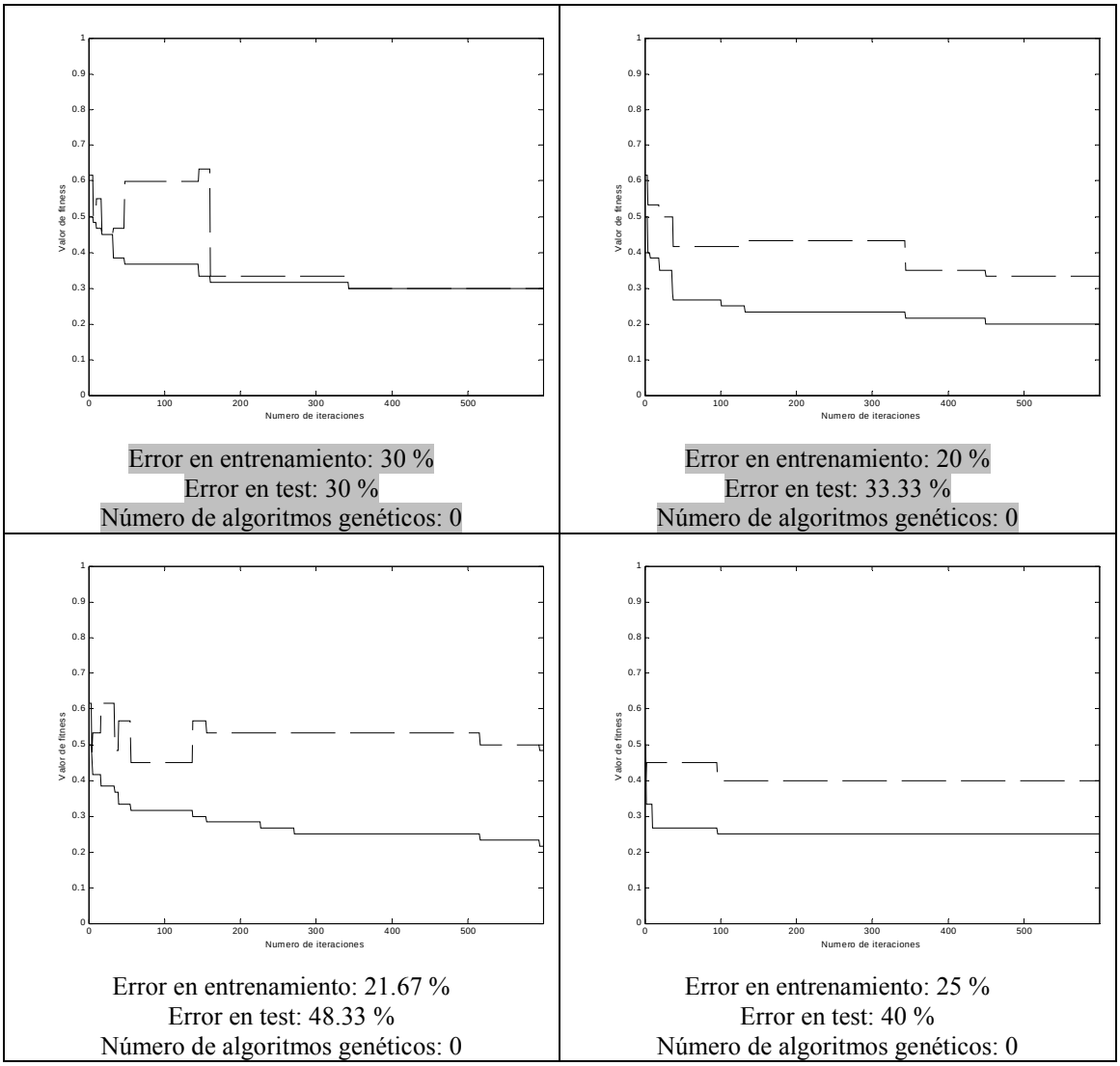


Figura 7.31. Curvas de entrenamiento y test para el parámetro *porción* = 1.0.

7.6.3.4 Discusión de los resultados.

La principal característica observada es cómo el incremento del parámetro *porción* provoca la aparición de un comportamiento irregular en la frecuencia de disparo del algoritmo genético, véase por ejemplo los resultados para *porción* igual a 0.6, donde uno de los experimentos provoca el arranque del algoritmo genético 154 veces frente a una media de 28.67 en los otros 3 experimentos. A medida que se aumenta el número de meta-reglas del conjunto de encaje que deben superar el parámetro antigüedad aparecen

entrenamientos donde el algoritmo genético no se dispara ninguna vez en las primeras 600 iteraciones.

La razón de esta disparidad es que el criterio de arranque depende de la estructura del conjunto de encaje en cada momento del algoritmo, es decir, que hay una dependencia del estado del entorno. Existen estados (situaciones de la máquina de estados borrosa) que tienden a favorecer el arranque del algoritmo genético en contraposición a otros.

Por ejemplo, la llegada de una máquina a situaciones no contempladas por meta-reglas de la población obliga a la renovación del conjunto de encaje debido a que no se alcanza el mínimo número de meta-reglas requerido. En ese caso, las nuevas meta-reglas se introducen mediante el proceso de recubrimiento y tendrán una antigüedad inferior al umbral establecido. En esta situación el algoritmo genético no se disparará. Sin embargo, también es posible el caso opuesto, en el que la máquina es modificada continuamente hacia estados cubiertos por la población de meta-reglas. En esa situación las meta-reglas del conjunto de encaje obtienen fácilmente la antigüedad necesaria para que la fracción de meta-reglas suficientemente antiguas requerida para el arranque del algoritmo genético se alcance fácilmente, originando el arranque continuo del algoritmo genético, como previsiblemente ocurrió en la situación ya comentada con el valor 0.6 del parámetro.

Estas situaciones pueden ser favorecidas por la utilización de un algoritmo genético de estado estacionario donde parte de la población (los mejores individuos) es copiada a la siguiente generación.

Por otra parte, analizando las curvas de entrenamiento y test, se observa que las curvas de test presentan en todos los casos un mal comportamiento, debido, como se ha explicado anteriormente, a la longitud de las series temporales usadas en estas pruebas.

El objetivo de esta prueba es analizar la influencia que tiene la frecuencia de los algoritmos genéticos sobre la calidad del entrenamiento. En las pruebas realizadas se observa que considerar el parámetro relacionado con la antigüedad de los elementos que encajan (*porción*) para controlar el ritmo con que se arrancan los algoritmos genéticos no es una buena estrategia, ya que existe una gran varianza en la frecuencia de arranque de los algoritmos genéticos para pruebas similares. Por lo tanto, en las siguientes pruebas intentaremos estudiar lo mismo, pero controlando directamente la frecuencia de

los algoritmos genéticos por medio de un nuevo parámetro que dispare el algoritmo genético a un ritmo fijo de iteraciones.

Tras estudiar los resultados de estas pruebas, parece que no hay gran diferencia en la eficiencia del algoritmo en función de la frecuencia de algoritmos genéticos. Esto puede deberse a que en solamente 600 iteraciones no se puede apreciar la evolución del aprendizaje. Una fracción muy pequeña de las pruebas realizadas se comporta de un modo compatible con el hecho de que en la máquina esté aprendiendo el modelo subyacente en los datos. Se observa en estas pruebas que la curva de test sigue a la curva de entrenamiento, por ejemplo en las gráficas destacadas en las figuras 7.26, 7.28, 7.29 y 7.31, aunque en cualquier caso los errores en el conjunto de test siguen siendo considerables.

7.6.4 Segundo estudio de la influencia en el sistema tipo Michigan de la frecuencia de disparo del algoritmo genético.

7.6.4.1 Descripción del experimento.

En esta prueba se pretende estudiar cómo influye la frecuencia con la que se ejecutan los algoritmos genéticos sobre la calidad del aprendizaje. El estudio anterior permitió comprobar que la estrategia basada en la antigüedad de una fracción de la población de meta-reglas no permite controlar la frecuencia de disparo de los algoritmos genéticos de un modo regular en esta implementación del sistema tipo Michigan. Para solucionar este problema, pasaremos a controlar directamente la frecuencia de los algoritmos genéticos por medio de un nuevo parámetro que dispara el algoritmo genético a un ritmo fijo de iteraciones, el parámetro *min_iter*.

Se realizan distintos procesos de entrenamiento, para 11 valores distintos del parámetro *min_iter*, se obtendrán así un total de 88 máquinas tipo Michigan. En todas estas pruebas se ha partido de la misma máquina inicial y se ha ejecutado el algoritmo durante el mismo número de iteraciones (600 iteraciones).

Se generan 60 secuencias con 15 muestras por secuencia para el modelo 1 y 60 secuencias con 15 muestras por secuencia para el modelo 2.

Se construye el conjunto de entrenamiento con 30 secuencias del modelo 1 y 30 secuencias del modelo 2 (en total, 60 secuencias, 30 de cada modelo) y el conjunto de

test con las otras 30 secuencias no utilizadas del modelo 1 y las otras 30 secuencias no utilizadas del modelo 2 (en total, 60 secuencias, 30 de cada modelo).

Los parámetros del sistema de tipo Michigan serán los utilizados hasta ahora y que han sido especificados en la tabla 7.11.

7.6.4.2 Resumen de resultados.

Vamos ahora a mostrar un resumen de los resultados obtenidos en este experimento realizado sobre 600 iteraciones del algoritmo.

En la tabla 7.14 se muestra la fracción del total de máquinas entrenadas para valor de *min_iter* con eficiencia aceptable. En la tabla 7.15 se presentan resultados comparativos para las distintas frecuencias de disparo de los algoritmos genéticos estudiadas. En las figura 7.32 se muestra la gráfica correspondiente al porcentaje de entrenamientos considerados eficientes para diferentes valores del parámetro *min_iter*. En la figura 7.33 se muestra la media del error de entrenamiento y la desviación en relación al parámetro *min_iter*.

Prueba	Número total de máquinas entrenadas	Número de máquinas con eficiencia aceptable	Porcentaje de máquinas con eficiencia aceptable
300 alg. genéticos (*min_iter* = 1)	8	1	12.5 %
200 alg. genéticos (*min_iter* = 2)	8	2	25 %
150 alg. genéticos (*min_iter* = 3)	8	4	50 %
100 alg. genéticos (*min_iter* = 5)	4	1	25 %
75 alg. genéticos (*min_iter* = 7)	8	2	25 %
60 alg. genéticos (*min_iter* = 9)	8	1	12.5 %
55 alg. genéticos (*min_iter* = 10)	8	5	62.5 %
29 alg. genéticos (*min_iter* = 20)	8	4	50 %
12 alg. genéticos (*min_iter* = 50)	8	3	37.5 %
9 alg. genéticos (*min_iter* = 70)	12	4	33.33 %
6 alg. genéticos (*min_iter* = 100)	8	4	50 %

Tabla 7.14. Resultados de entrenamientos con diferentes valores del parámetro *min_iter*. Fracción de máquinas encontradas con eficiencia aceptable.

Prueba	Media del error final en el entr.	Desv. del error final en el entr.	Media del error final en el test	Desv. del error final en el test	Media de iteraciones	Desv. de estas iteraciones
300 GA (*min_iter* = 1)	20.83%	3.19	42.5%	5.5084	266.6250	158.5316
200 GA (*min_iter* = 2)	18.33%	3.21	39.5838%	3.3045	394.3750	183.8586
150 GA (*min_iter* = 3)	21.46%	2.88	39.3750%	4.9556	414	124.2440
100 GA (*min_iter* = 5)	19.16%	0.96	39.5825%	3.6955	390.5	160.5106
75 GA (*min_iter* = 7)	22.92%	1.72	40.4175%	4.8586	271.1250	213.5258
60 GA (*min_iter* = 9)	25.00%	3.98	43.3338%	4.8805	325.5	218.3968
55 GA (*min_iter* = 10)	21.46%	1.87	35.4162%	3.5347	353.5	203.3034
29 GA (*min_iter* = 20)	19.79%	1.39	38.3337%	6.3632	296.3750	196.7645
12 GA (*min_iter* = 50)	21.87%	4.58	40.4150%	6.7109	353.25	203.4290
9 GA (*min_iter* = 70)	20.83%	3.51	39.9983%	6.2369	272.0833	133.5695
6 GA (*min_iter* = 100)	22.08%	3.5357	38.5400%	7.7377	356.25	186.8030

Tabla 7.15. Resultados comparativos entre los entrenamientos con diferentes valores del parámetro *min_iter*.

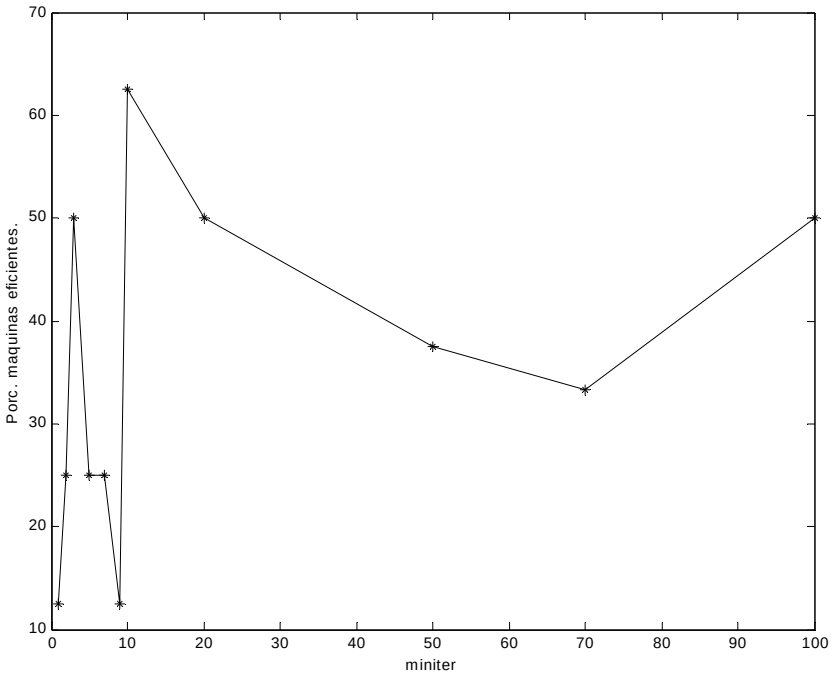


Figura 7.32. Evaluación del parámetro de control de la frecuencia de disparo *min_iter*, con el porcentaje de entrenamientos eficientes.

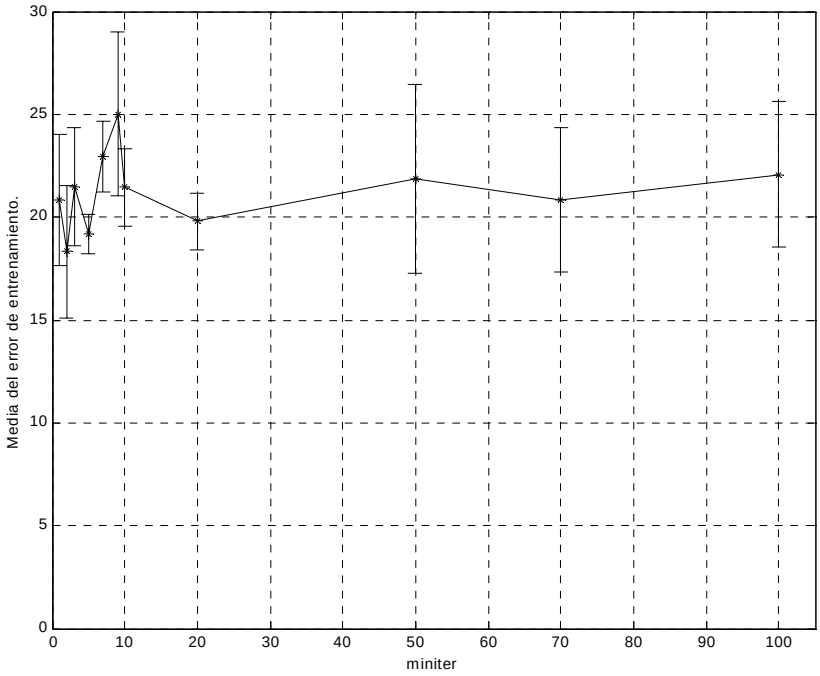


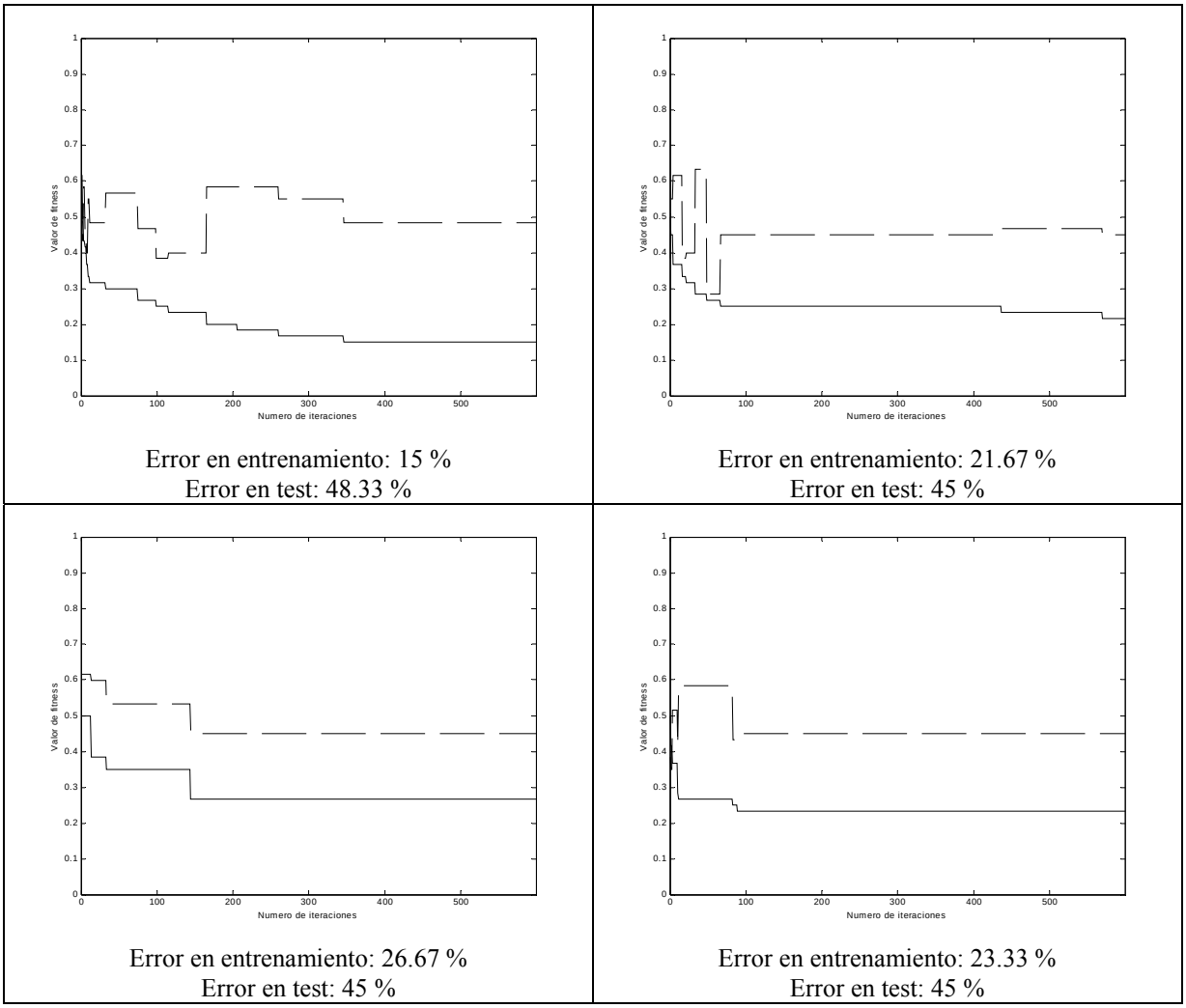
Figura 7.33. Error de entrenamiento medio y desviación en relación al parámetro *min_iter*.

7.6.4.3 Curvas de entrenamiento y test.

Las curvas de entrenamiento y test de estos experimentos se muestran en las figuras comprendidas entre la figura 7.34 y la figura 7.55. Estos experimentos se realizan para los valores de *min_iter*: 1, 2, 3, 5, 7, 9, 10, 20, 50, 70 y 100.

Pruebas con *min_iter* = 1.

Pasada una iteración desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 300 veces.



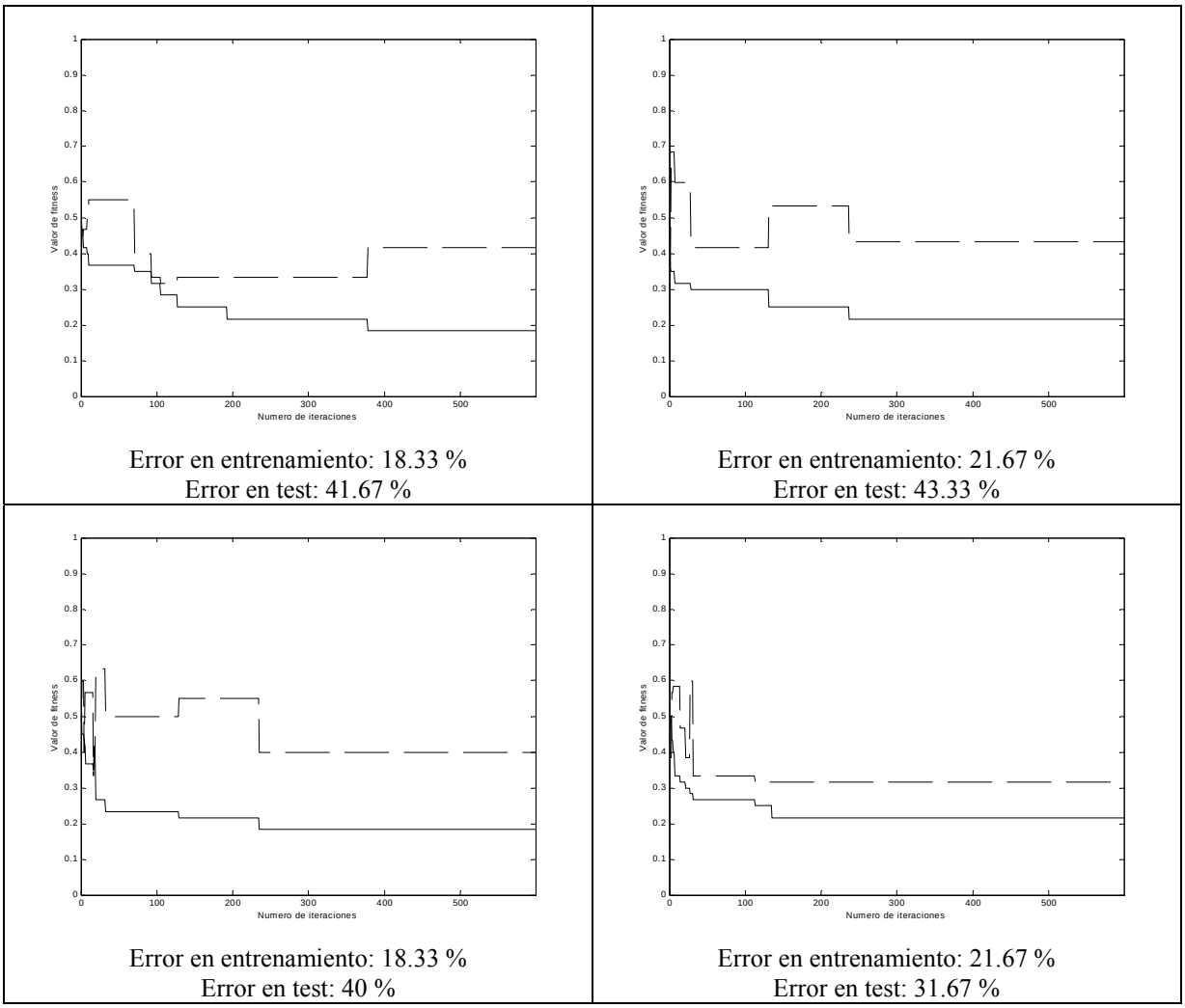
Error en entrenamiento: 15 %
Error en test: 48.33 %

Error en entrenamiento: 21.67 %
Error en test: 45 %

Error en entrenamiento: 26.67 %
Error en test: 45 %

Error en entrenamiento: 23.33 %
Error en test: 45 %

Figura 7.34. Curvas de test y entrenamiento para *min_iter* = 1 (1).



Error en entrenamiento: 18.33 %
Error en test: 41.67 %

Error en entrenamiento: 21.67 %
Error en test: 43.33 %

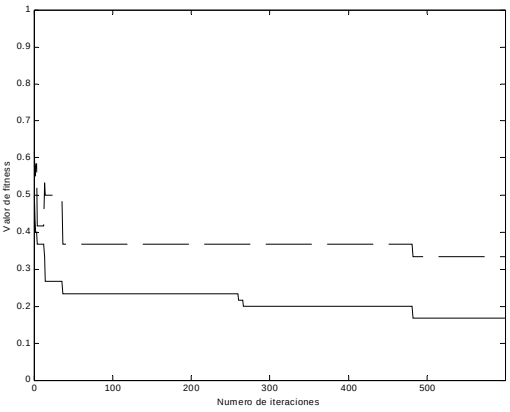
Error en entrenamiento: 18.33 %
Error en test: 40 %

Error en entrenamiento: 21.67 %
Error en test: 31.67 %

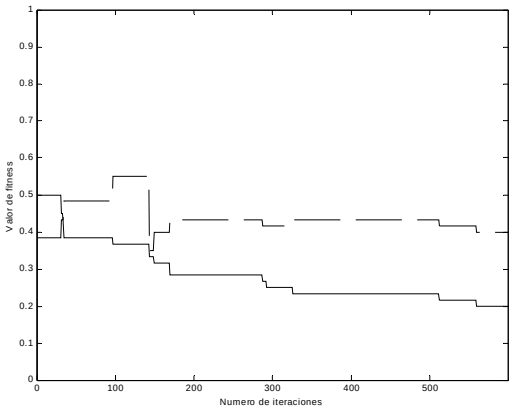
Figura 7.35. Curvas de test y entrenamiento para *min_iter* = 1 (2).

Pruebas con *min_iter* = 2.

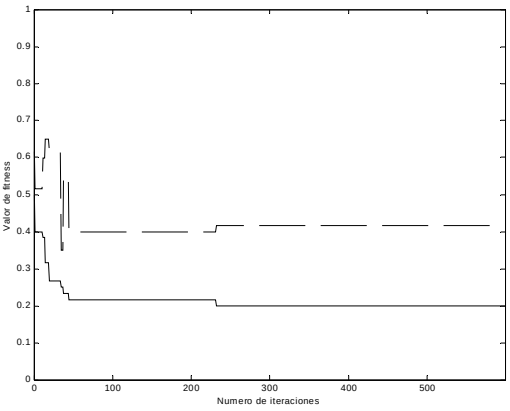
Pasadas dos iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 200 veces.



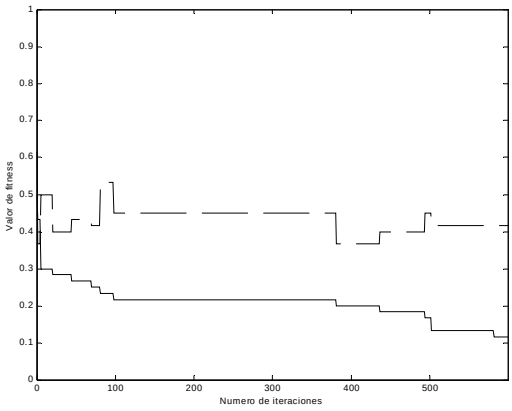
Error en entrenamiento: 16.67 %
Error en test: 33.33 %



Error en entrenamiento: 20 %
Error en test: 40 %



Error en entrenamiento: 20 %
Error en test: 41.67 %



Error en entrenamiento: 11.67 %
Error en test: 41.67 %

Figura 7.36. Curvas de test y entrenamiento para *min_iter* =2 (1).

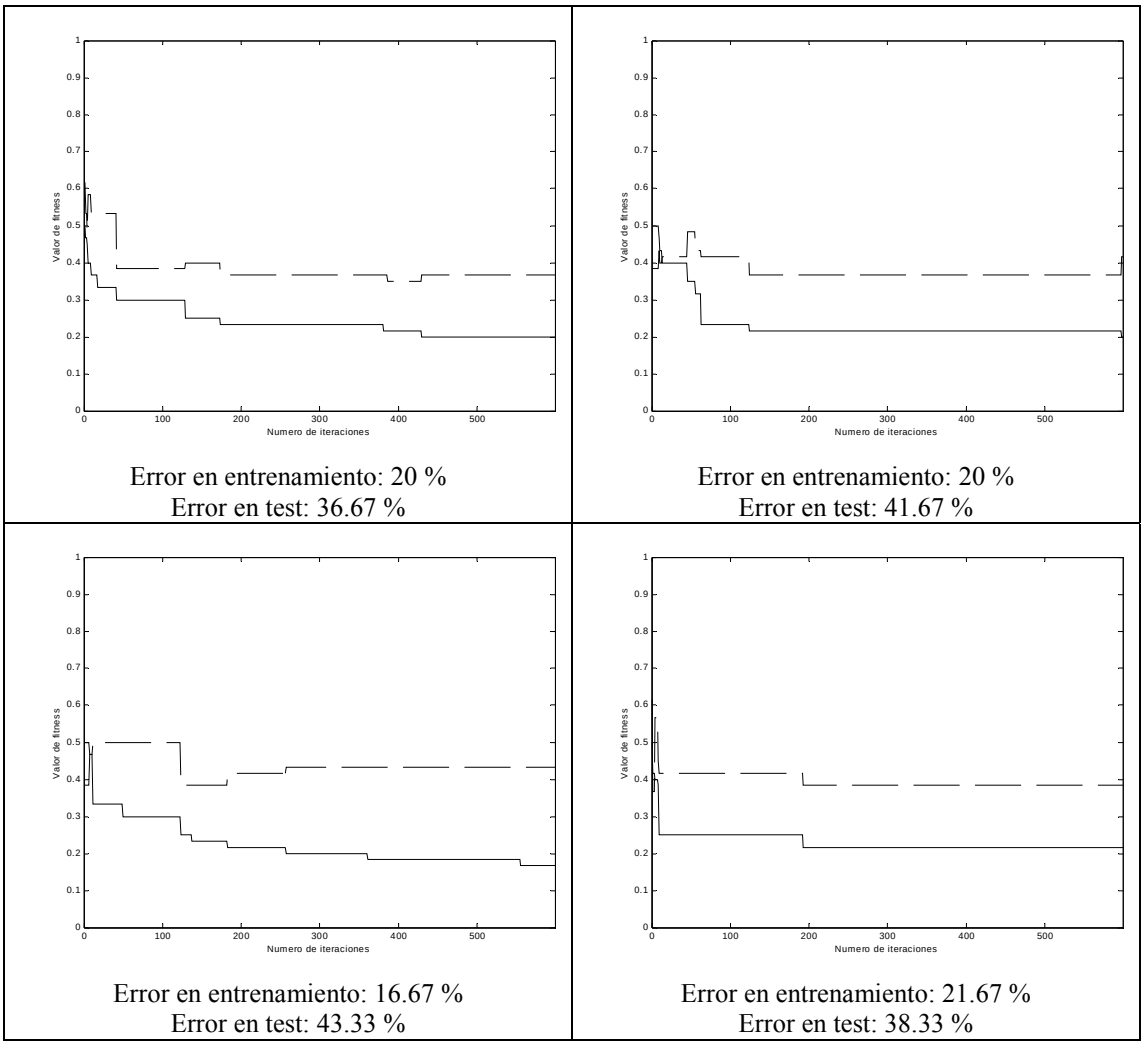
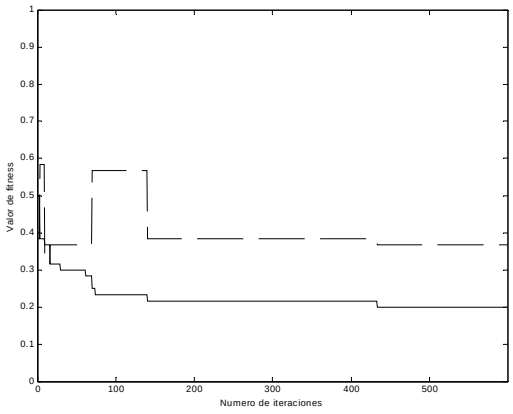


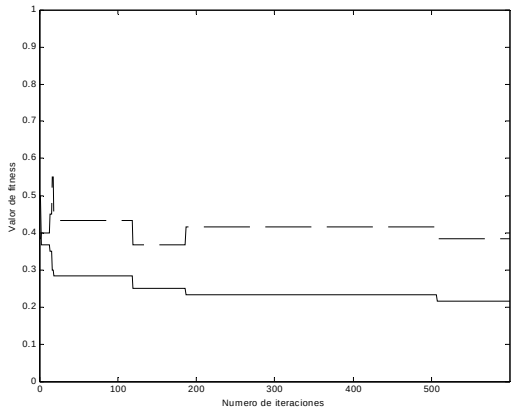
Figura 7.37. Curvas de test y entrenamiento para *min_iter* =2 (2).

Pruebas con *min_iter* = 3.

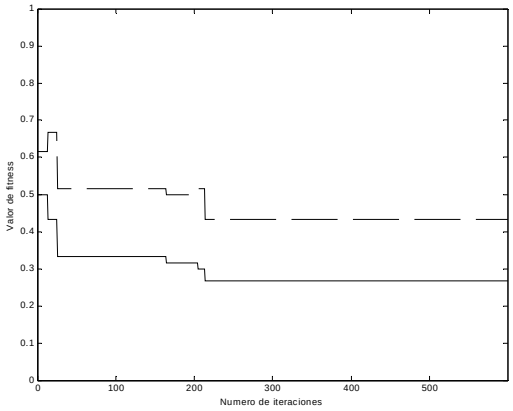
Pasadas tres iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 150 veces.



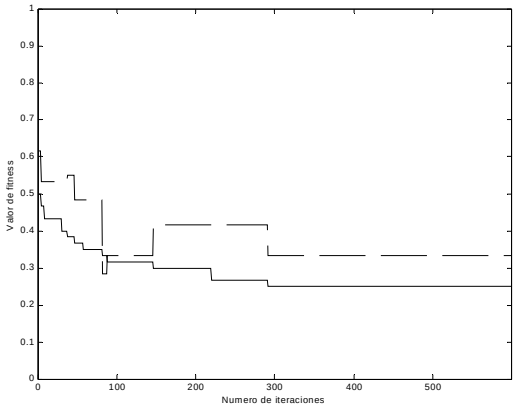
Error en entrenamiento: 20 %
Error en test: 36.67 %



Error en entrenamiento: 21.67 %
Error en test: 38.33 %

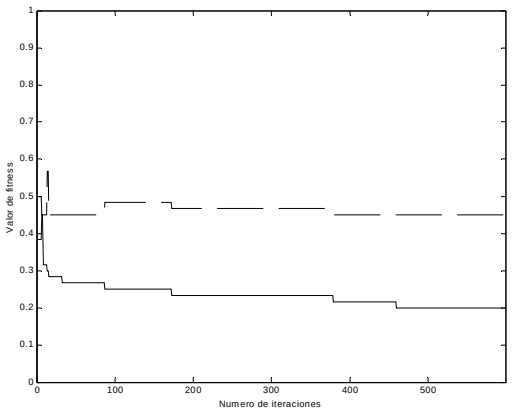


Error en entrenamiento: 26.67 %
Error en test: 43.33 %

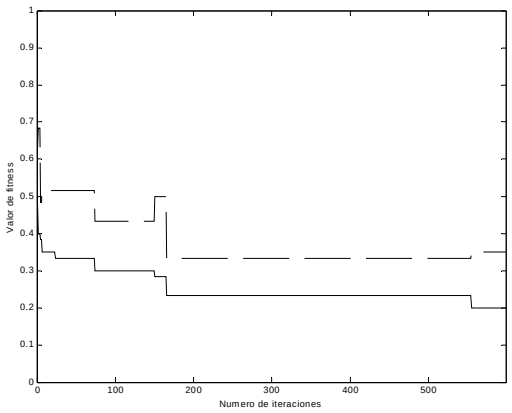


Error en entrenamiento: 25 %
Error en test: 33.33 %

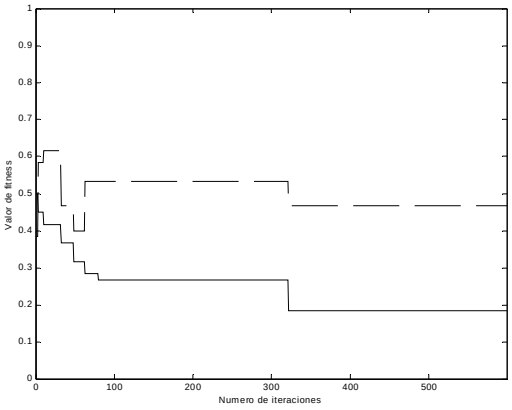
Figura 7.38. Curvas de test y entrenamiento para *min_iter* = 3 (1).



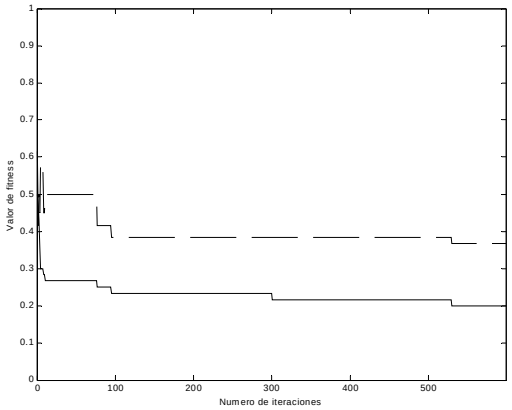
Error en entrenamiento: 20 %
Error en test: 45 %



Error en entrenamiento: 20 %
Error en test: 35 %



Error en entrenamiento: 18.33 %
Error en test: 46.67 %

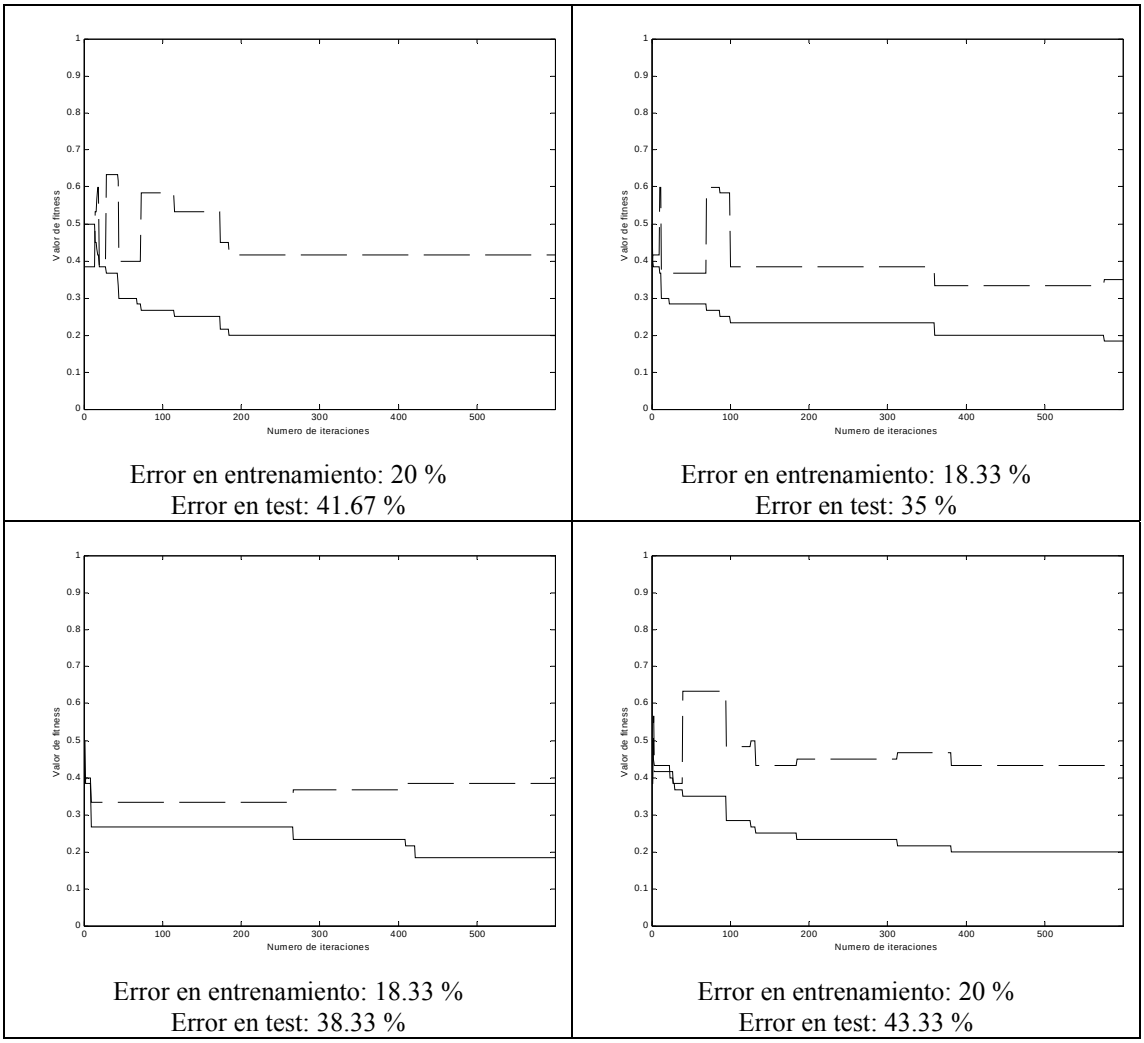


Error en entrenamiento: 20 %
Error en test: 36.67 %

Figura 7.39. Curvas de test y entrenamiento para *min_iter* = 3 (2).

Pruebas con *min_iter* = 5.

Pasadas 5 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 100 veces.



Error en entrenamiento: 20 %
Error en test: 41.67 %

Error en entrenamiento: 18.33 %
Error en test: 35 %

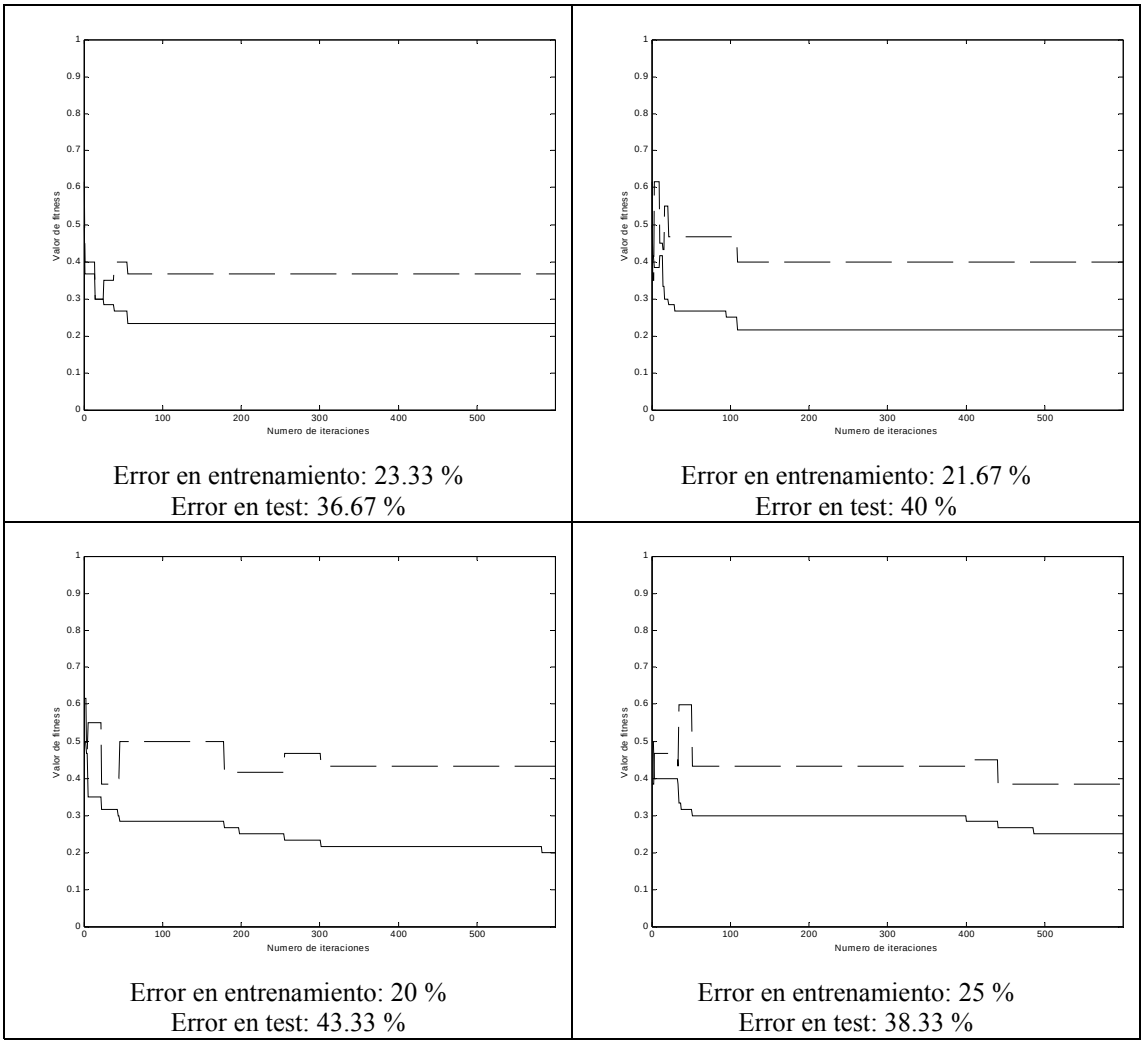
Error en entrenamiento: 18.33 %
Error en test: 38.33 %

Error en entrenamiento: 20 %
Error en test: 43.33 %

Figura 7.40. Curvas de test y entrenamiento para *min_iter* = 5.

Pruebas con *min_iter* = 7.

Pasadas 7 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 75 veces.



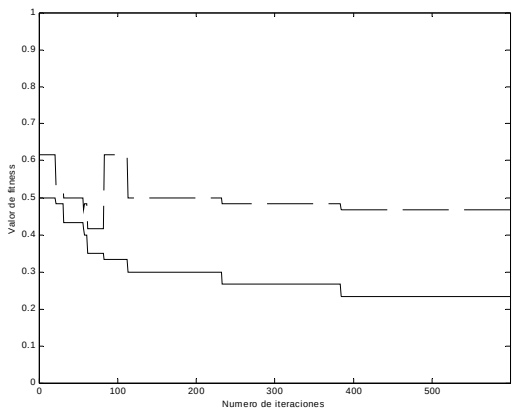
Error en entrenamiento: 23.33 %
Error en test: 36.67 %

Error en entrenamiento: 21.67 %
Error en test: 40 %

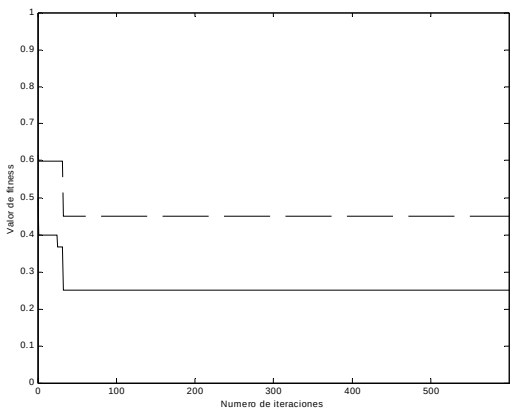
Error en entrenamiento: 20 %
Error en test: 43.33 %

Error en entrenamiento: 25 %
Error en test: 38.33 %

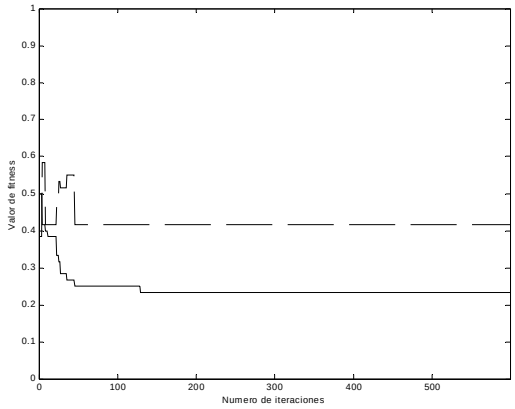
Figura 7.41. Curvas de test y entrenamiento para *min_iter* = 7 (1).



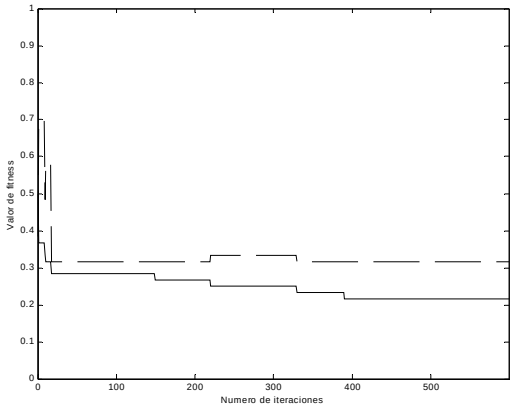
Error en entrenamiento: 23.33 %
Error en test: 46.67 %



Error en entrenamiento: 25 %
Error en test: 45 %



Error en entrenamiento: 23.33 %
Error en test: 41.67 %

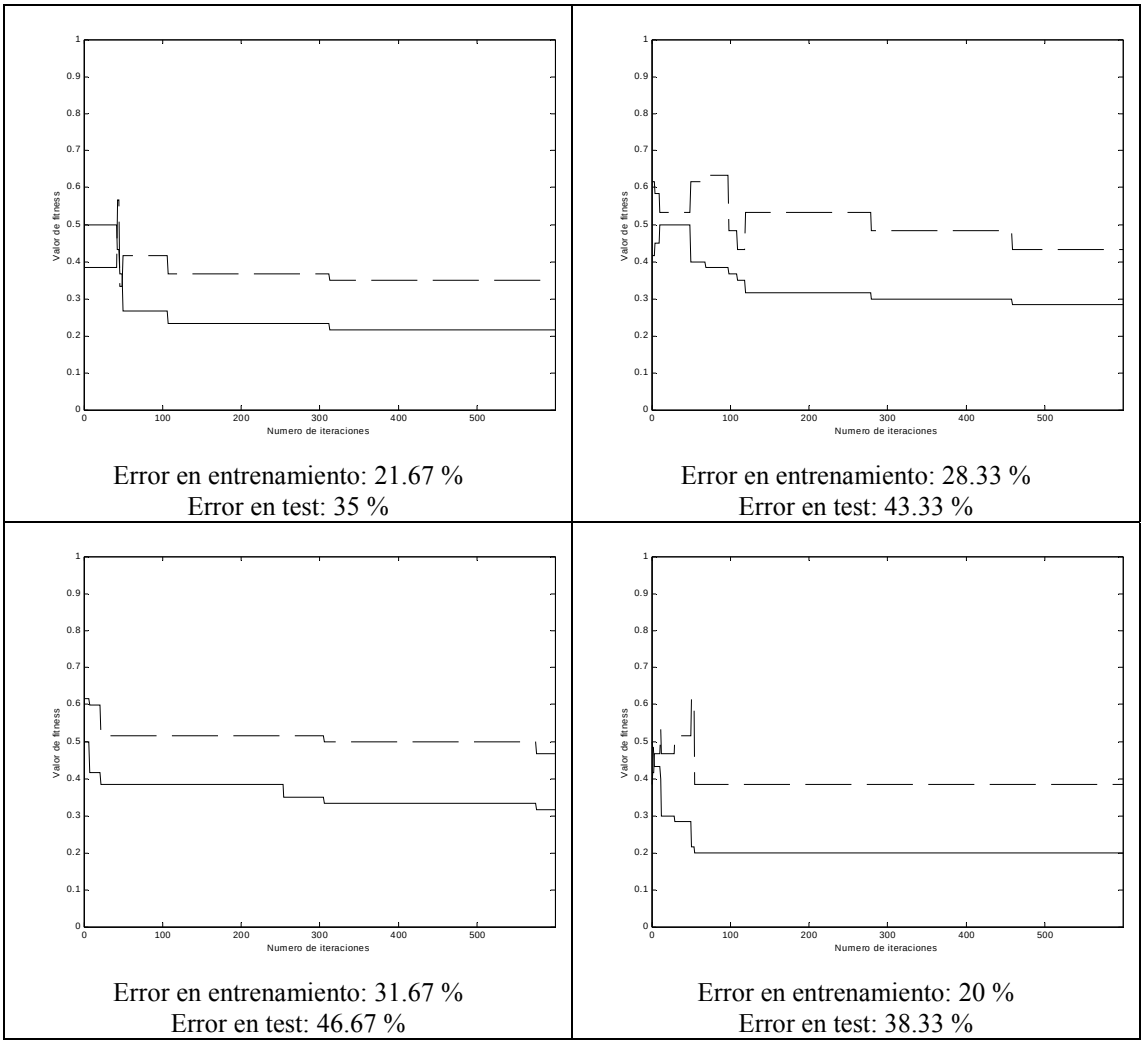


Error en entrenamiento: 21.67 %
Error en test: 31.67 %

Figura 7.42. Curvas de test y entrenamiento para *min_iter* = 7 (2).

Pruebas con *min_iter* = 9.

Pasadas 9 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 60 veces.



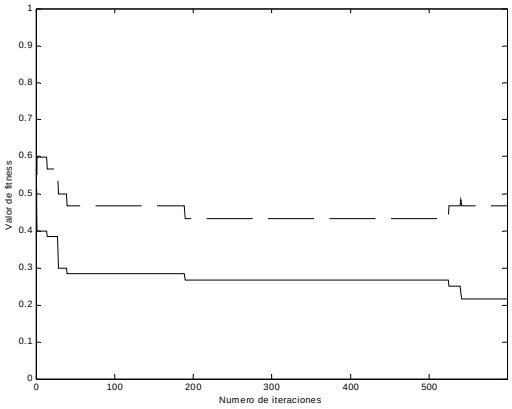
Error en entrenamiento: 21.67 %
Error en test: 35 %

Error en entrenamiento: 28.33 %
Error en test: 43.33 %

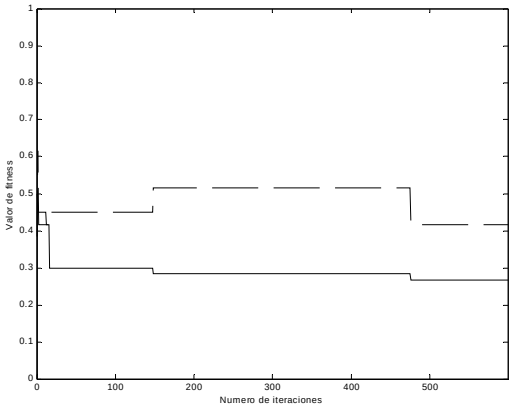
Error en entrenamiento: 31.67 %
Error en test: 46.67 %

Error en entrenamiento: 20 %
Error en test: 38.33 %

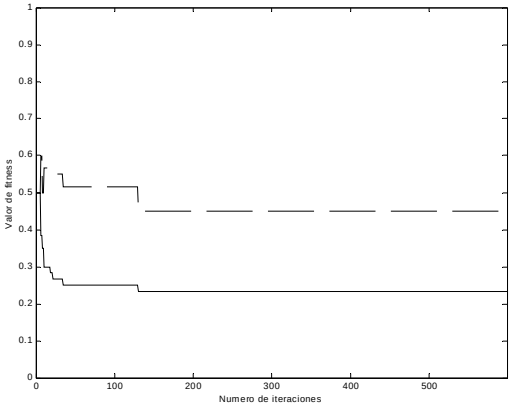
Figura 7.43. Curvas de test y entrenamiento para *min_iter* = 9 (1).



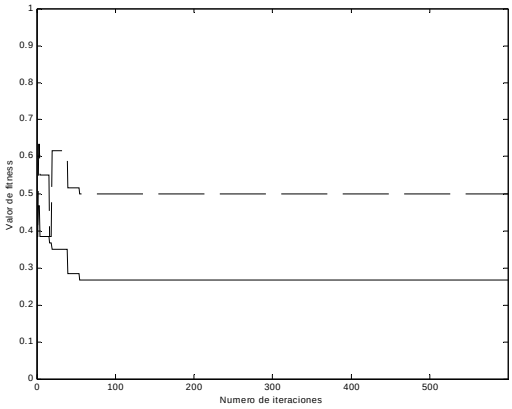
Error en entrenamiento: 21.67 %
Error en test: 46.67 %



Error en entrenamiento: 26.67 %
Error en test: 41.67 %



Error en entrenamiento: 23.33 %
Error en test: 45 %

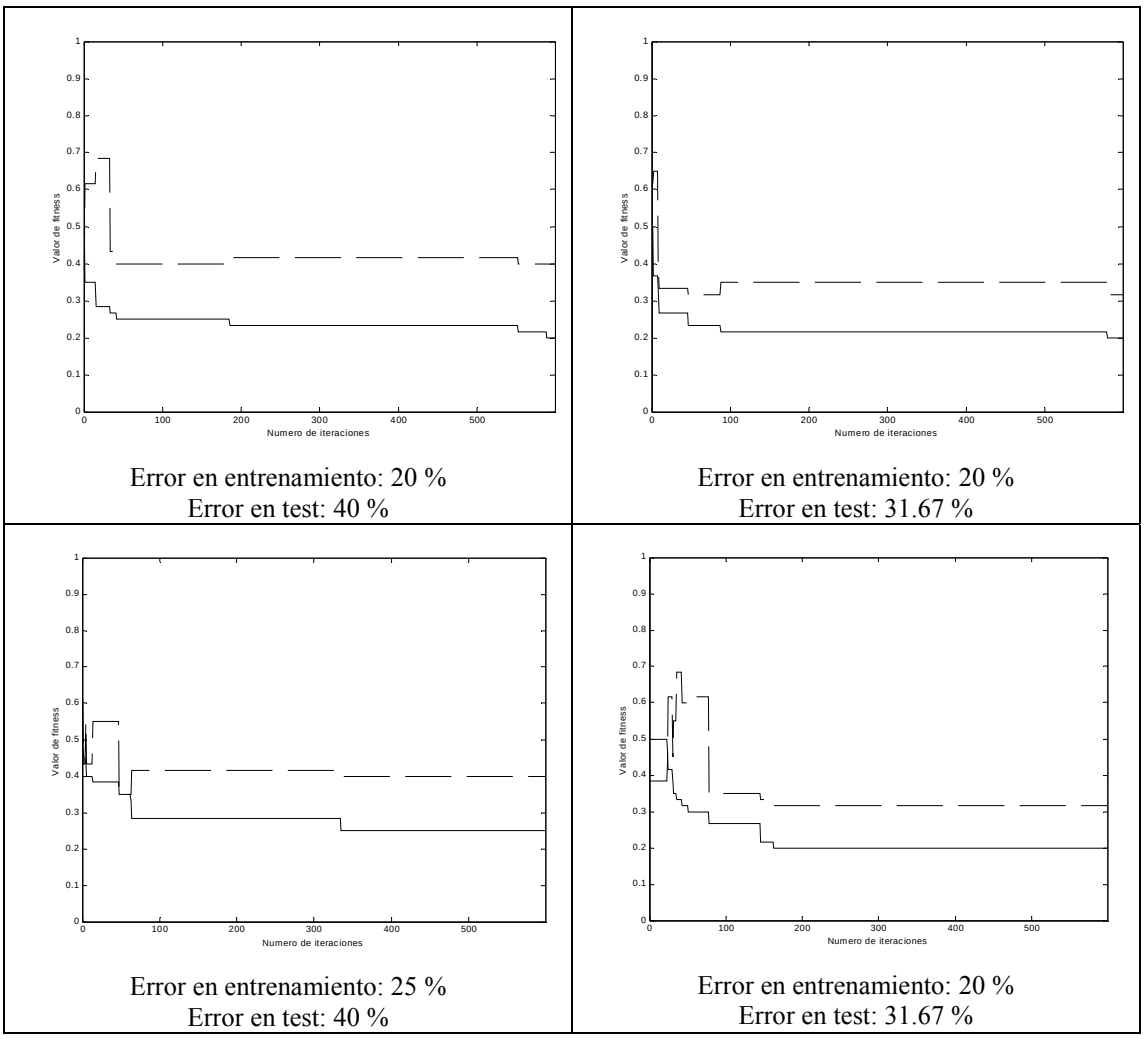


Error en entrenamiento: 26.67 %
Error en test: 50 %

Figura 7.44. Curvas de test y entrenamiento para *min_iter* = 9 (2).

Pruebas con *min_iter* = 10.

Pasadas 10 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 55 veces.



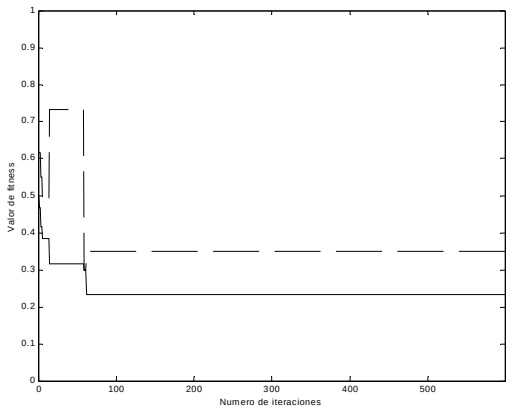
Error en entrenamiento: 20 %
Error en test: 40 %

Error en entrenamiento: 20 %
Error en test: 31.67 %

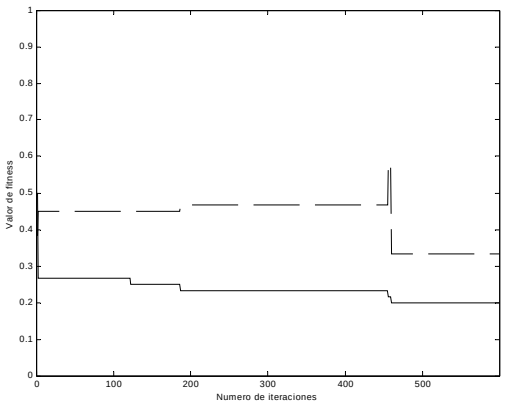
Error en entrenamiento: 25 %
Error en test: 40 %

Error en entrenamiento: 20 %
Error en test: 31.67 %

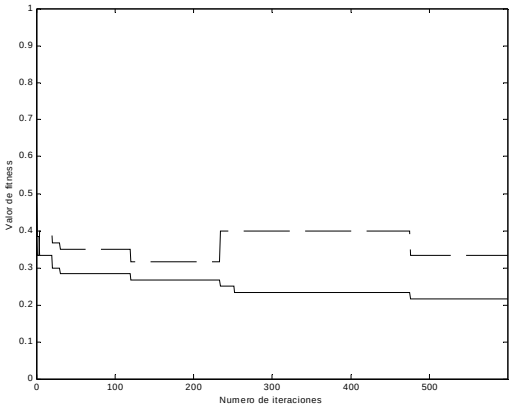
Figura 7.45. Curvas de test y entrenamiento para *min_iter* = 10 (1).



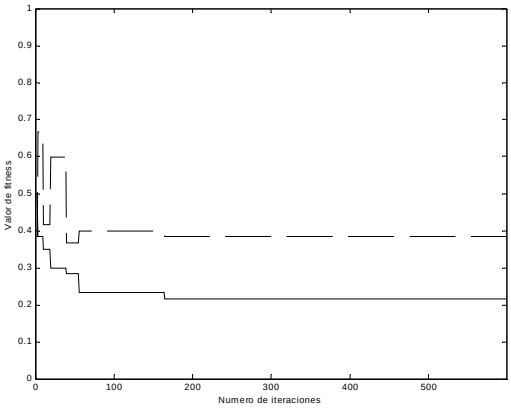
Error en entrenamiento: 23.33 %
Error en test: 35 %



Error en entrenamiento: 20 %
Error en test: 33.33 %



Error en entrenamiento: 21.67 %
Error en test: 33.33 %

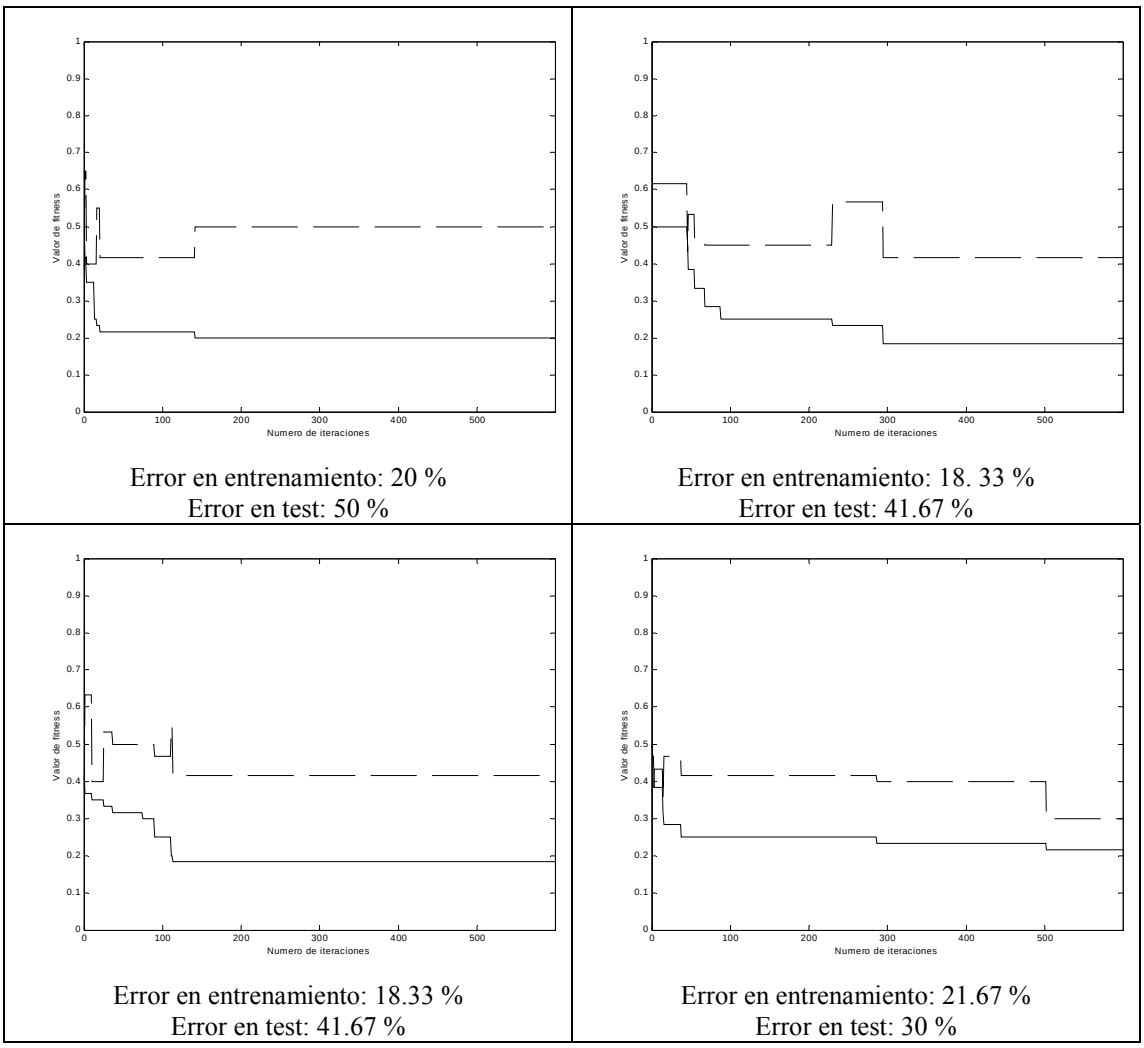


Error en entrenamiento: 21.67 %
Error en test: 38.33 %

Figura 7.46. Curvas de test y entrenamiento para *min_iter* = 10 (2).

Pruebas con *min_iter* = 20.

Pasadas 20 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 29 veces.



Error en entrenamiento: 20 %
Error en test: 50 %

Error en entrenamiento: 18. 33 %
Error en test: 41.67 %

Error en entrenamiento: 18.33 %
Error en test: 41.67 %

Error en entrenamiento: 21.67 %
Error en test: 30 %

Figura 7.47. Curvas de test y entrenamiento para *min_iter* = 20 (1).

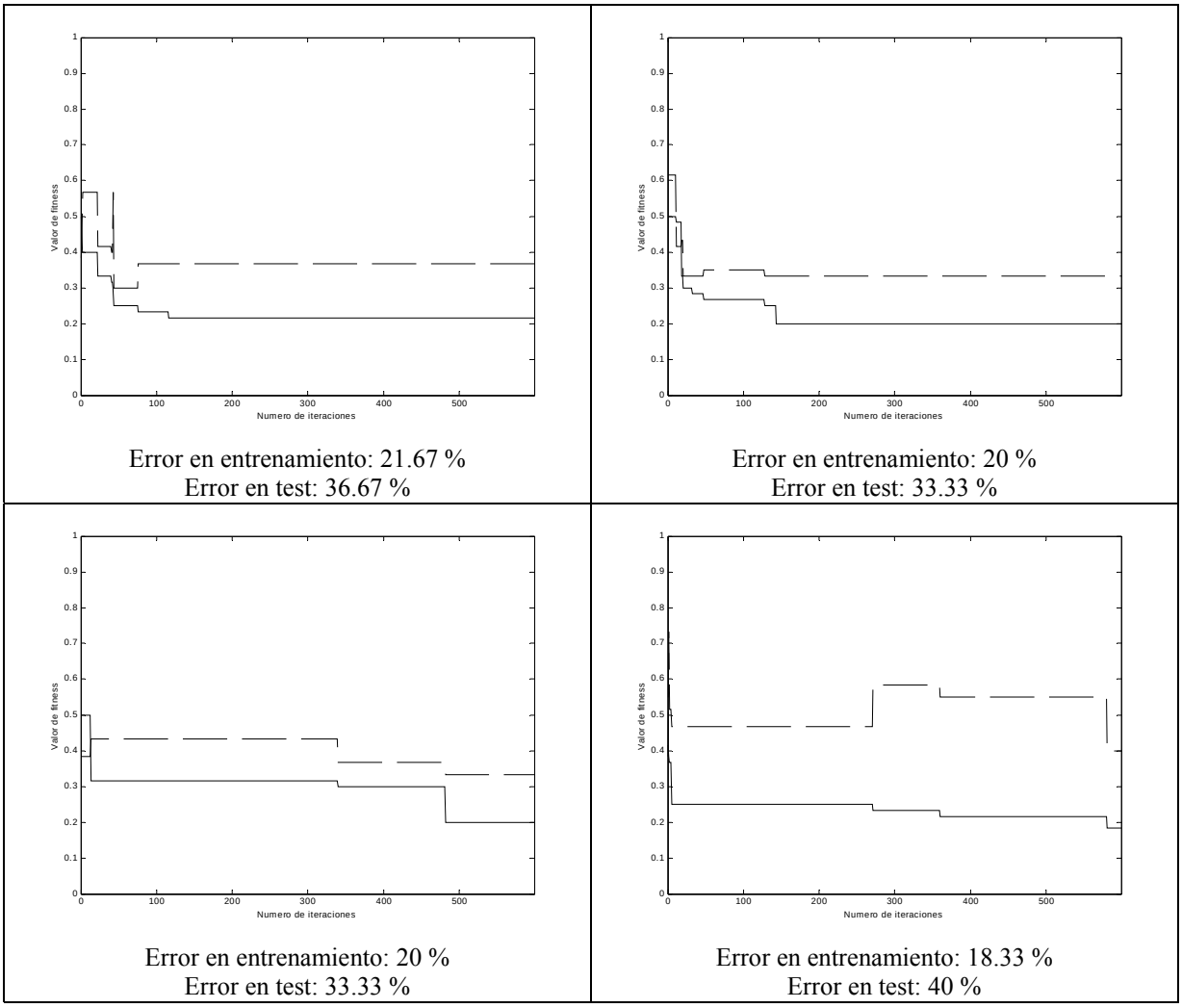
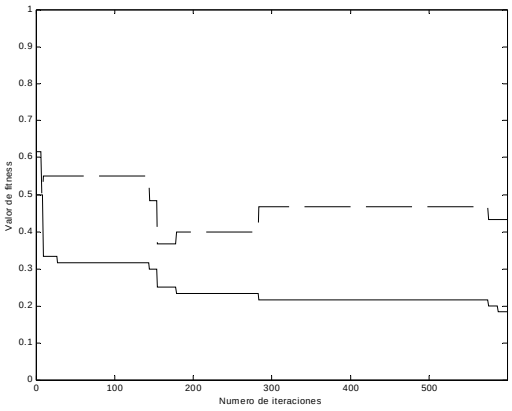


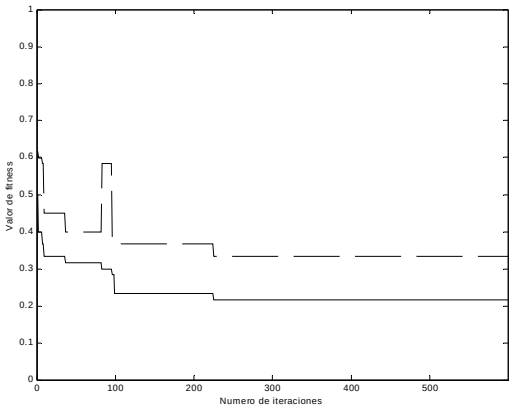
Figura 7.48. Curvas de test y entrenamiento para *min_iter* = 20 (2).

Pruebas con *min_iter* = 50.

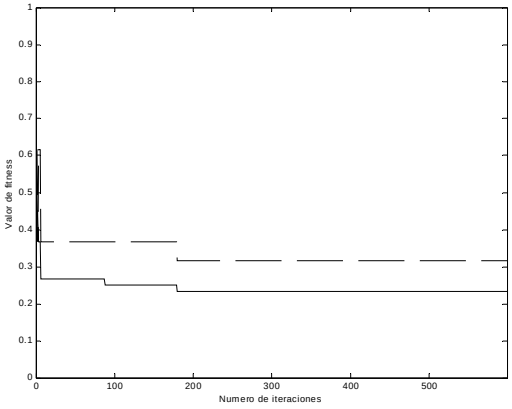
Pasadas 50 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 12 veces.



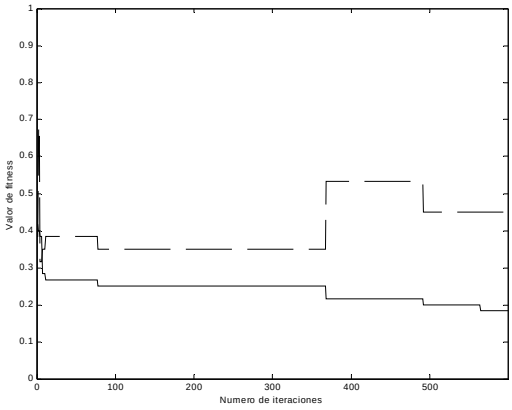
Error en entrenamiento: 18.33 %
Error en test: 43.33 %



Error en entrenamiento: 21.67 %
Error en test: 33.33 %

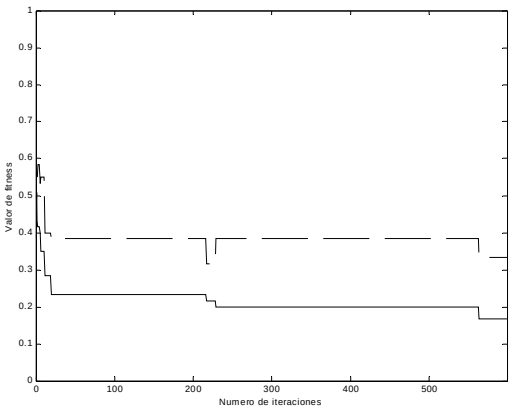


Error en entrenamiento: 23.33 %
Error en test: 31.67 %

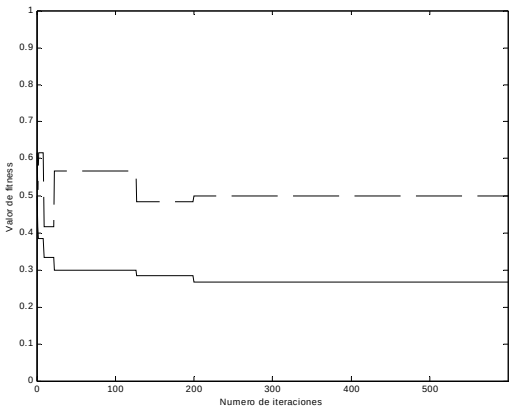


Error en entrenamiento: 18.33 %
Error en test: 45 %

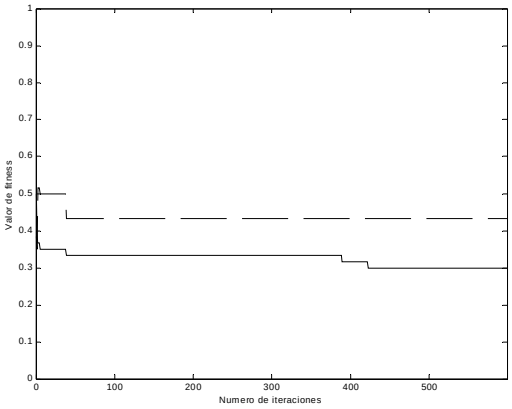
Figura 7.49. Curvas de test y entrenamiento para *min_iter* = 50 (1).



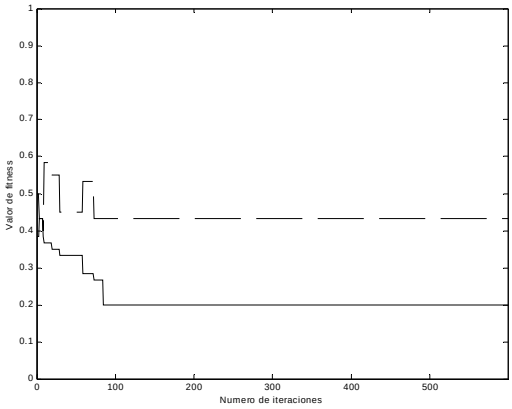
Error en entrenamiento: 16.67 %
Error en test: 33.33 %



Error en entrenamiento: 26.67 %
Error en test: 50 %



Error en entrenamiento: 30 %
Error en test: 43.33 %

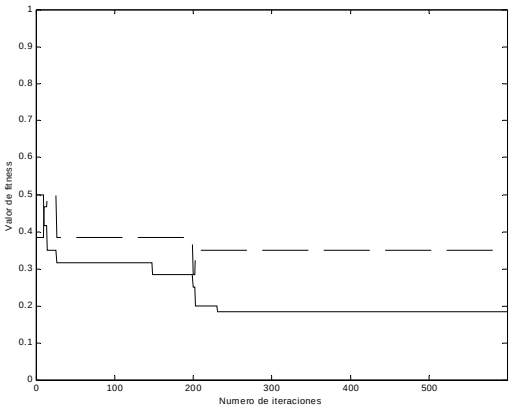


Error en entrenamiento: 20 %
Error en test: 43.33 %

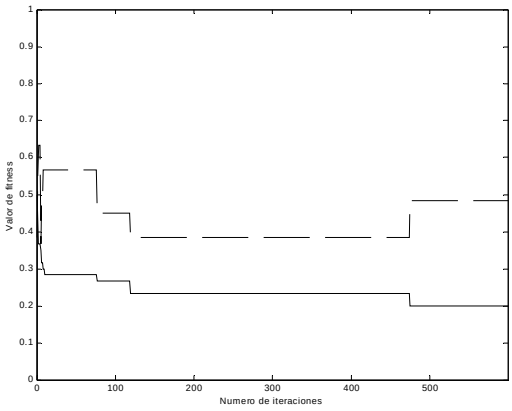
Figura 7.50. Curvas de test y entrenamiento para *min_iter* = 50 (2).

Pruebas con *min_iter* = 70.

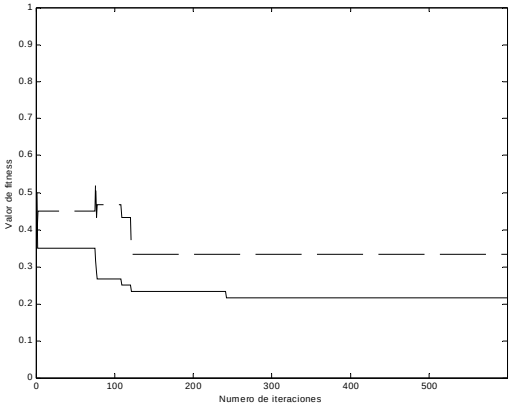
Pasadas 70 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 9 veces.



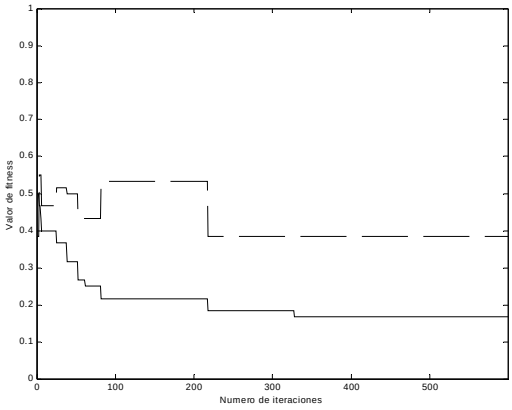
Error en entrenamiento: 18.33 %
Error en test: 35 %



Error en entrenamiento: 20 %
Error en test: 48.33 %

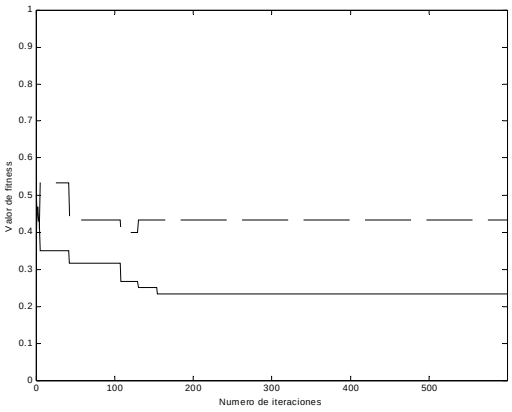


Error en entrenamiento: 21.67 %
Error en test: 33.33 %

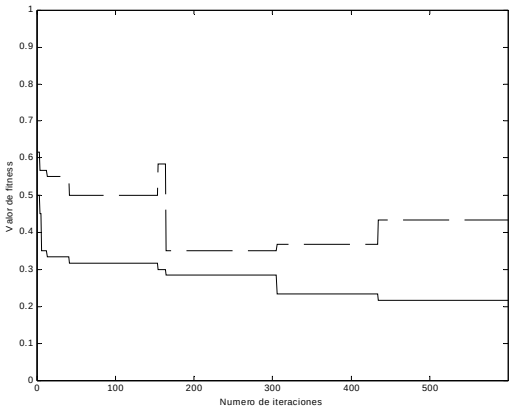


Error en entrenamiento: 16.67 %
Error en test: 38.33 %

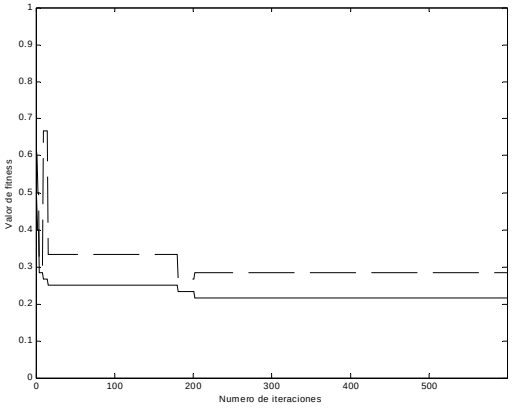
Figura 7.51. Curvas de test y entrenamiento para *min_iter* = 70 (1).



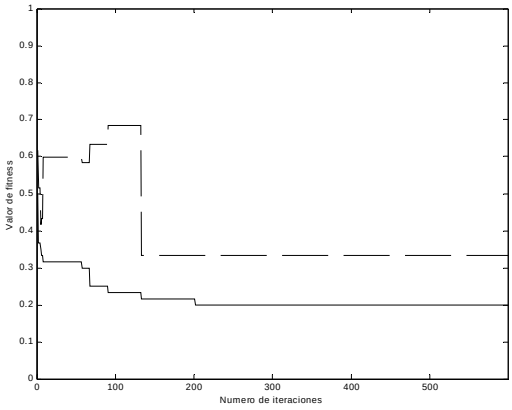
Error en entrenamiento: 23.33 %
Error en test: 43.33 %



Error en entrenamiento: 21.67 %
Error en test: 43.33 %

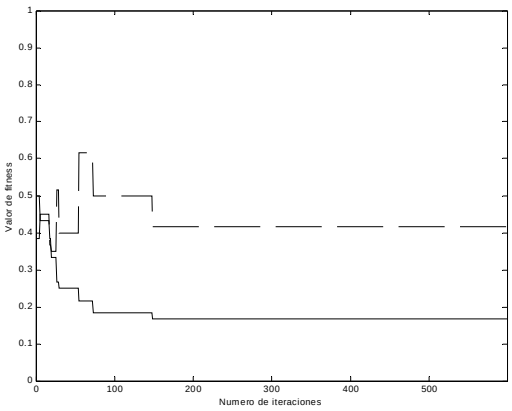


Error en entrenamiento: 21.67 %
Error en test: 28.33 %

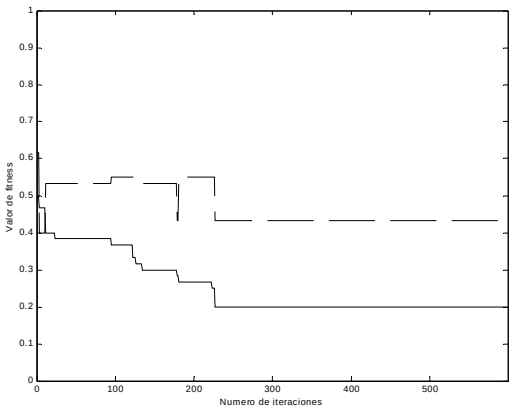


Error en entrenamiento: 20 %
Error en test: 33.33 %

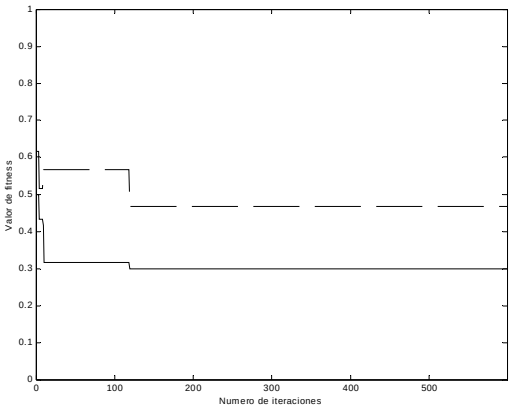
Figura 7.52. Curvas de test y entrenamiento para *min_iter* = 70 (2).



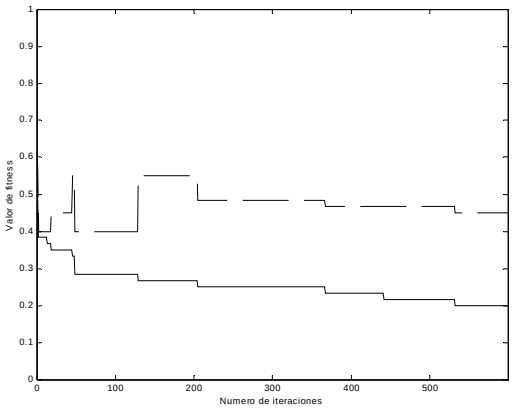
Error en entrenamiento: 16.67 %
Error en test: 41.67 %



Error en entrenamiento: 20 %
Error en test: 43.33 %



Error en entrenamiento: 30 %
Error en test: 46.67 %

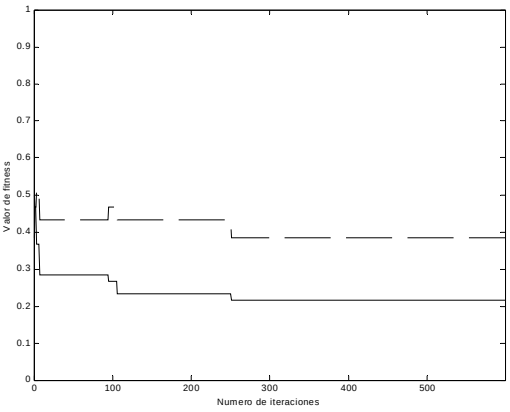


Error en entrenamiento: 20 %
Error en test: 45 %

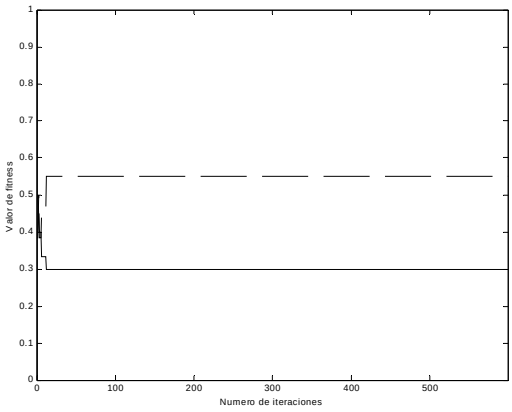
Figura 7.53. Curvas de test y entrenamiento para *min_iter* = 70 (3).

Pruebas con *min_iter* = 100.

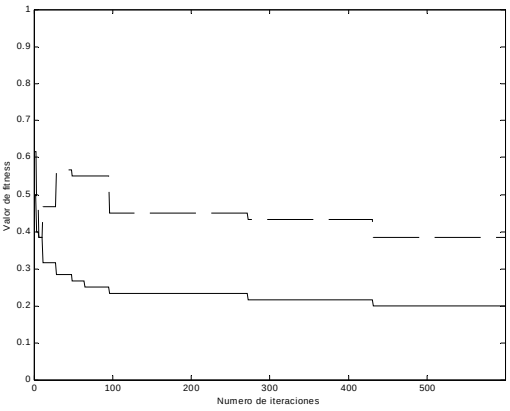
Pasadas 100 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 6 veces.



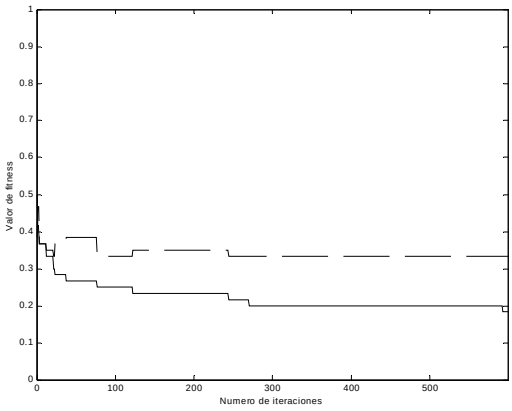
Error en entrenamiento: 21.67 %
Error en test: 38.33 %



Error en entrenamiento: 30 %
Error en test: 55 %



Error en entrenamiento: 20 %
Error en test: 38.33 %



Error en entrenamiento: 18.33 %
Error en test: 33.33 %

Figura 7.54. Curvas de test y entrenamiento para *min_iter* = 100 (1).

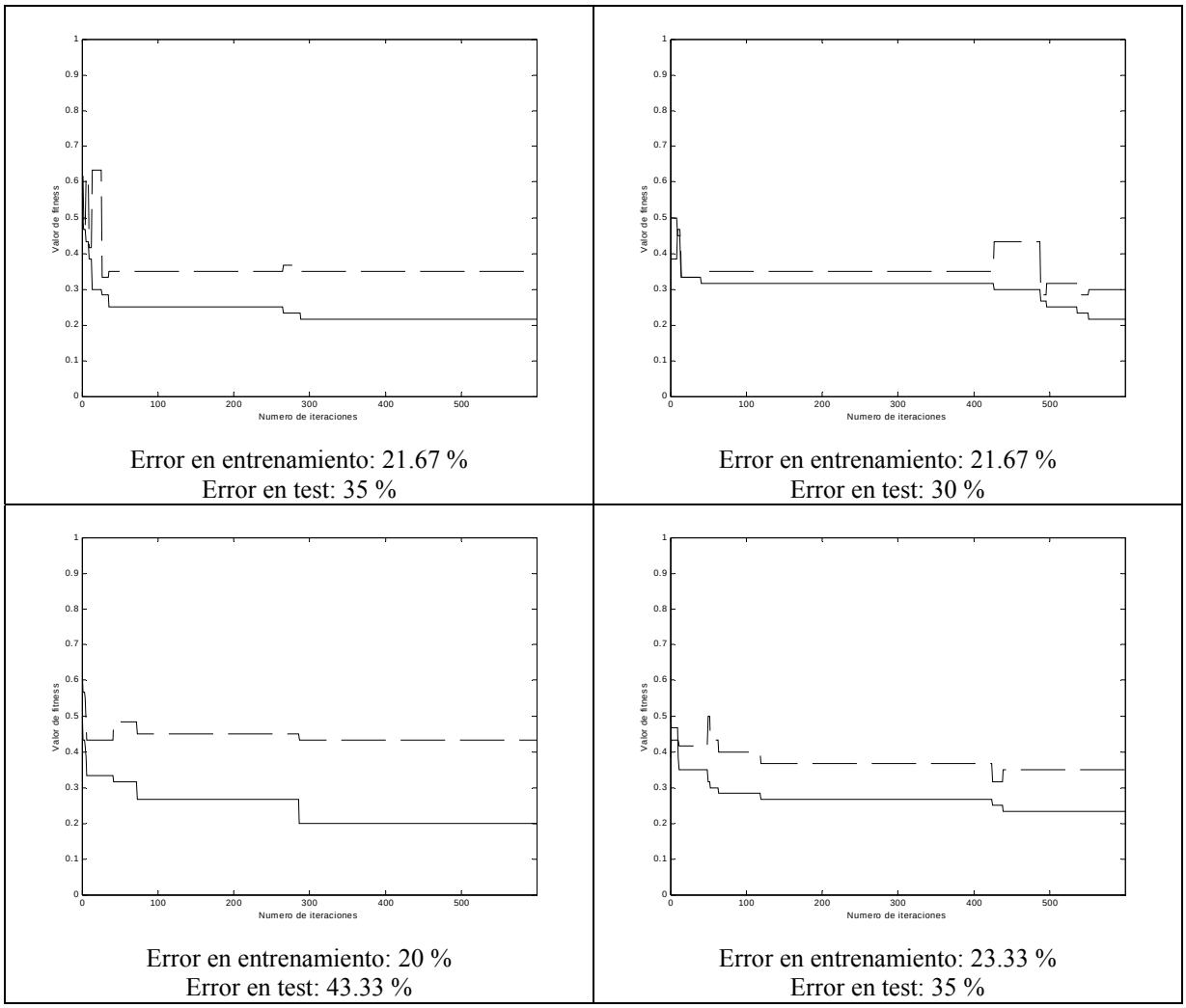


Figura 7.55. Curvas de test y entrenamiento para *min_iter* =100 (2).

7.6.4.4 Discusión de los resultados.

La prueba que presenta mejor comportamiento en la clasificación del conjunto de test es la correspondiente a *min_iter* = 10, en la que el algoritmo genético se dispara 55 veces. Es la prueba que cuenta con mayor número de máquinas entrenadas con eficiencia aceptable, y la que presenta un error medio final menor al clasificar el conjunto de test. La prueba con *min_iter* = 20 presenta un error de entrenamiento bueno y una desviación pequeña comparada con el resto, aunque su error en el conjunto de test es peor.

Este valor intermedio puede indicar que los casos extremos, es decir, disparar el algoritmo genético muchas veces o dispararlo muy pocas veces, no son los más convenientes para aumentar la calidad del aprendizaje como cabría esperar.

Para valores pequeños de *min_iter* (alta frecuencia de disparo de algoritmos genéticos) el comportamiento es errático, pero a partir de *min_iter* = 10 y *min_iter* = 20 aparece una tendencia de decaimiento y luego una cierta recuperación.

Sin embargo, en cuanto al error de test, se puede apreciar en los resultados de estas pruebas, que no hay una diferencia demasiado grande en la eficiencia del algoritmo en función de la frecuencia de disparo de los algoritmos genéticos. Esto puede confirmar nuestra hipótesis inicial de que en solamente 600 iteraciones no se puede apreciar la evolución del aprendizaje, ya que el comportamiento del algoritmo es aún bastante aleatorio. Para estudiar esto más profundamente se realizarán otras pruebas en las que el algoritmo evoluciona durante más iteraciones (2000 iteraciones) en el siguiente experimento .

7.6.5 Tercer estudio de la influencia en el sistema tipo Michigan de la frecuencia de disparo del algoritmo genético.

7.6.5.1 Descripción del experimento.

Se realizan distintos procesos de entrenamiento, para 5 valores distintos del parámetro *min_iter*, en total, 25 máquinas tipo Michigan. En todas estas pruebas se ha partido de la misma máquina inicial y se ha ejecutado el algoritmo durante más iteraciones que en el experimento anterior, llegándose hasta 2000 iteraciones. En esta prueba se han realizado menos entrenamientos debido a que los entrenamientos de 2000 iteraciones son mucho más lentos.

Se generan 60 secuencias con 15 muestras por secuencia para el modelo 1 y 60 secuencias con 15 muestras por secuencia para el modelo 2.

Se construye el conjunto de entrenamiento con 30 secuencias del modelo 1 y 30 secuencias del modelo 2 (en total, 60 secuencias, 30 de cada modelo) y el conjunto de test con las otras 30 secuencias no utilizadas del modelo 1 y las otras 30 secuencias no utilizadas del modelo 2 (en total, 60 secuencias, 30 de cada modelo).

Se utilizarán los mismos parámetros para el sistema tipo Michigan que en los experimentos anteriores.

7.6.5.2 Resumen de resultados.

En la tabla 7.16 se presentan resultados comparativos para las distintas frecuencias de disparo de los algoritmos genéticos. En la figura 7.56 se representan los errores de entrenamiento y test encontrados para los distintos valores de *min_iter* probados: 2, 7, 9, 15 y 70.

Prueba	Media del error final en el entrenamiento	Desviación del error final en el entrenamiento	Media del error final en el test	Desviación del error final en el test
667 alg. genéticos (*min_iter* = 2)	19.3300 %	2.2361	41.6660 %	5.6522
250 alg. genéticos (*min_iter* = 7)	18.6680 %	3.4171	42 %	7.7646
200 alg. genéticos (*min_iter* = 9)	19.6660 %	1.3972	38.6660 %	6.9114
125 alg. genéticos (*min_iter* = 15)	16.9980 %	2.7389	37 %	5.9383
29 alg. genéticos (*min_iter* = 70)	20.3320 %	1.8253	45 %	9.2043

Tabla 7.16. Resumen de resultados relativos al experimento con diferentes frecuencias de disparo del algoritmo genético y 2000 iteraciones.

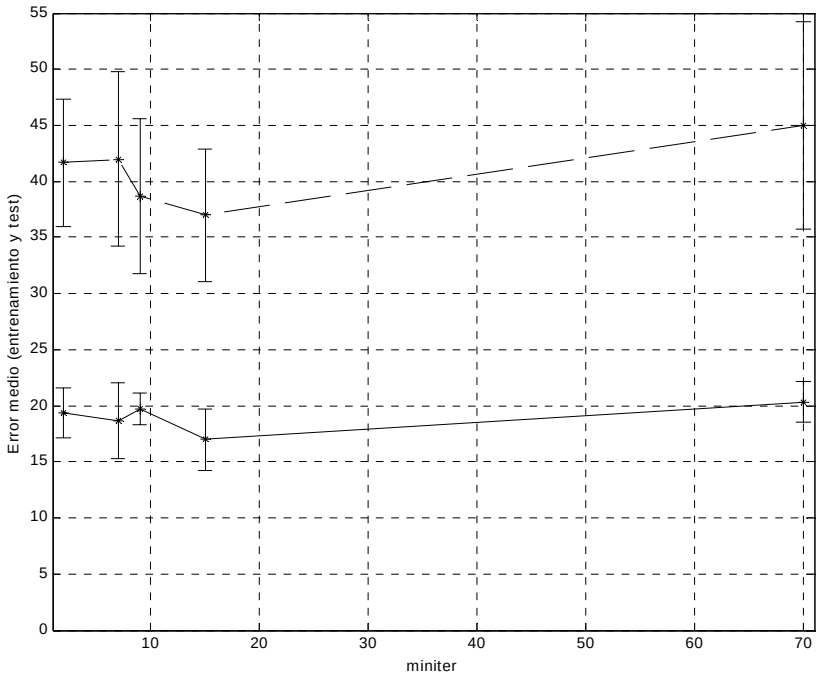


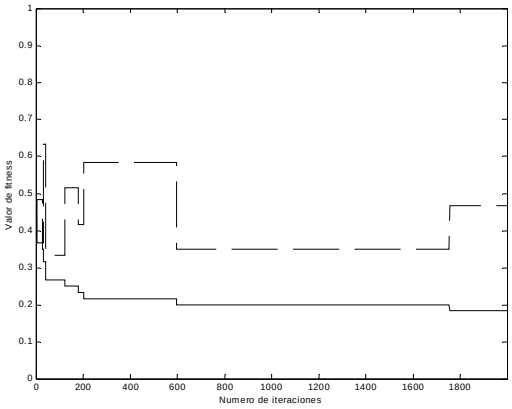
Figura 7.56. Error en el entrenamiento y test para diferentes valores de *min_iter* en 2000 iteraciones.

7.6.5.3 Curvas de entrenamiento y test.

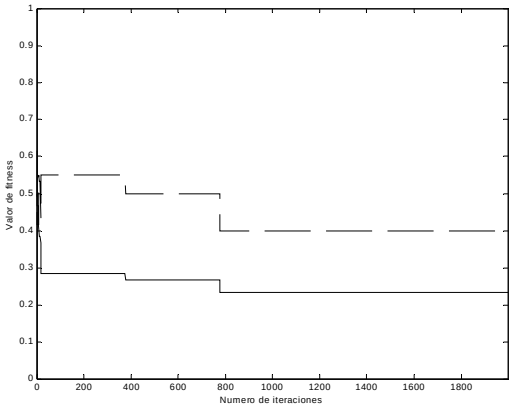
En las figuras comprendidas entre la figura 7.57 y la figura 7.61se presentan las curvas de entrenamiento y test de las distintas pruebas realizadas para los valores de *min_iter* probados.

Pruebas con *min_iter* = 2.

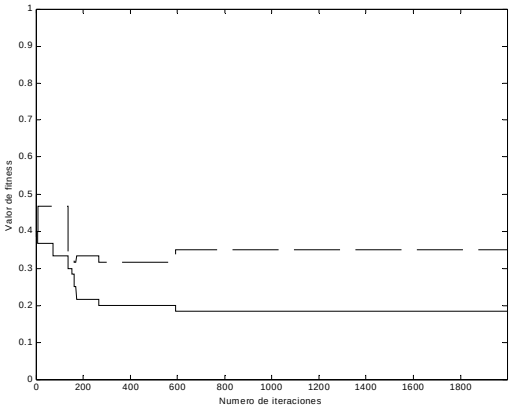
Pasadas 2 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 667 veces.



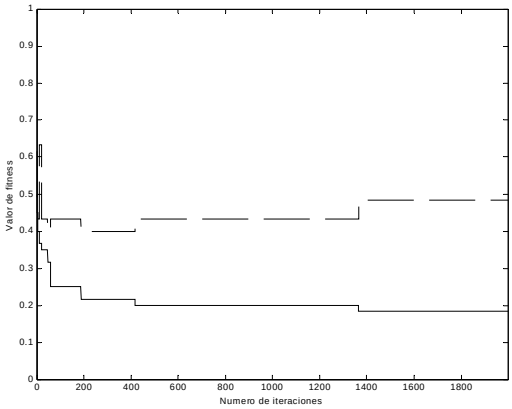
Error en entrenamiento: 18.33 %
Error en test: 46.67 %



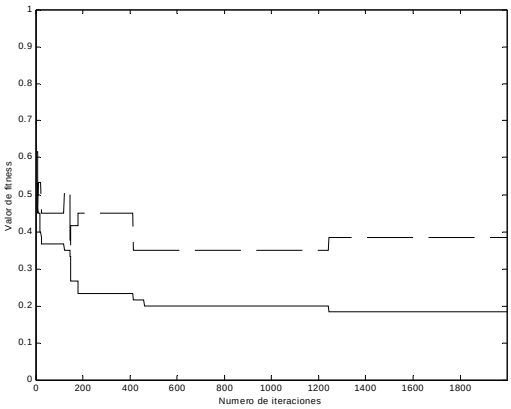
Error en entrenamiento: 23.33 %
Error en test: 40 %



Error en entrenamiento: 18.33 %
Error en test: 35 %



Error en entrenamiento: 18.33 %
Error en test: 48.33 %

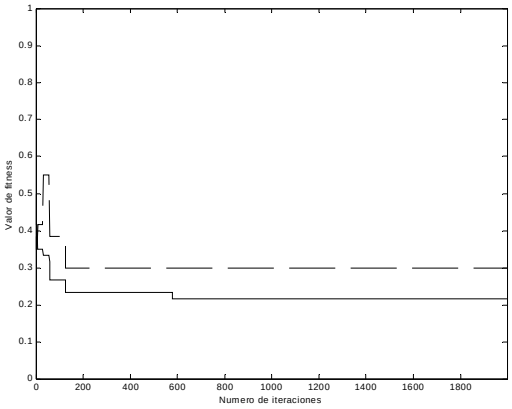


Error en entrenamiento: 18.33 %
Error en test: 38.33 %

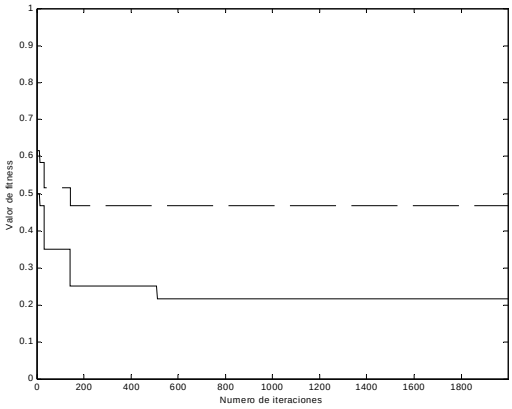
Figura 7.57. Curvas de entrenamiento y test para *min_iter* = 2.

Pruebas con *min_iter* = 7.

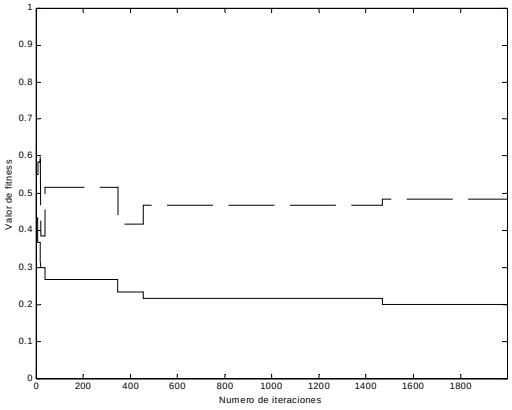
Pasadas 7 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 250 veces.



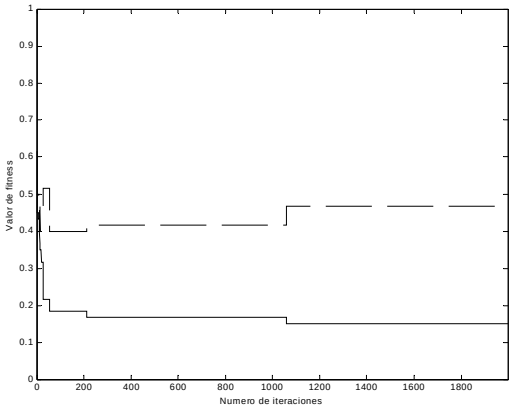
Error en entrenamiento: 21.67 %
Error en test: 30 %



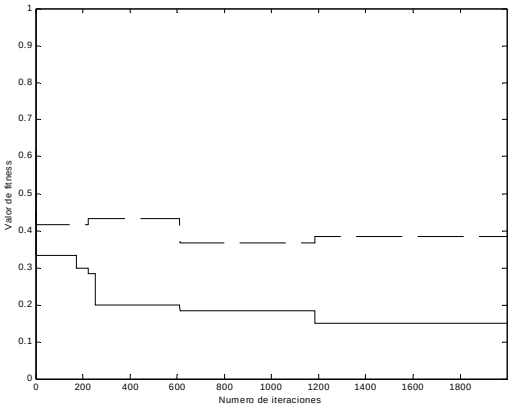
Error en entrenamiento: 21.67 %
Error en test: 46.67 %



Error en entrenamiento: 20 %
Error en test: 48.33 %



Error en entrenamiento: 15 %
Error en test: 46.67 %

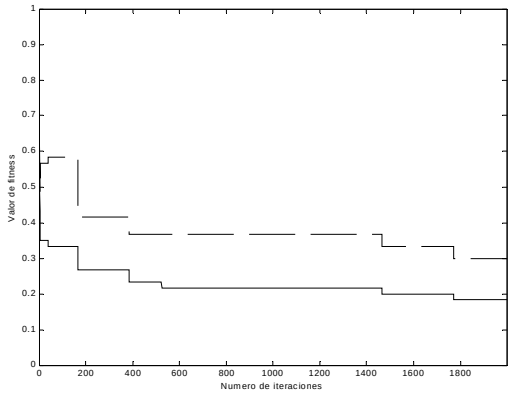


Error en entrenamiento: 15 %
Error en test: 38.33 %

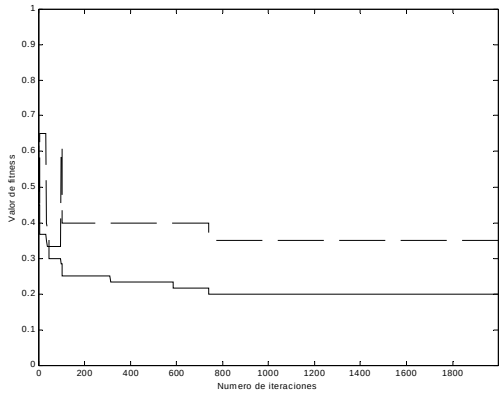
Figura 7.58. Curvas de test y entrenamiento para *min_iter* = 7.

Pruebas con *min_iter* = 9.

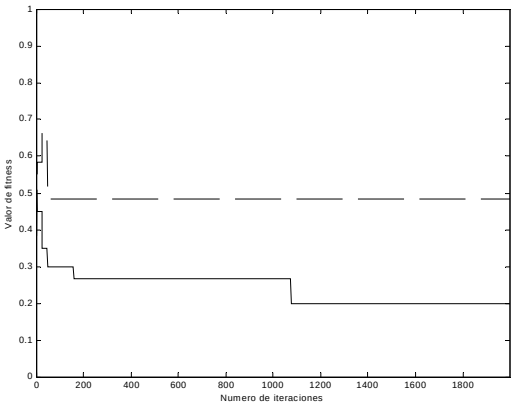
Pasadas 9 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 200 veces.



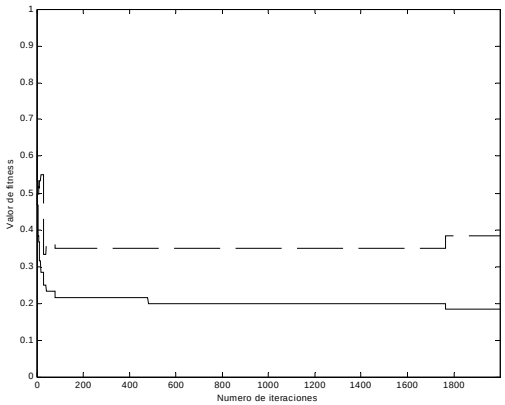
Error en entrenamiento: 18.33 %
Error en test: 30 %



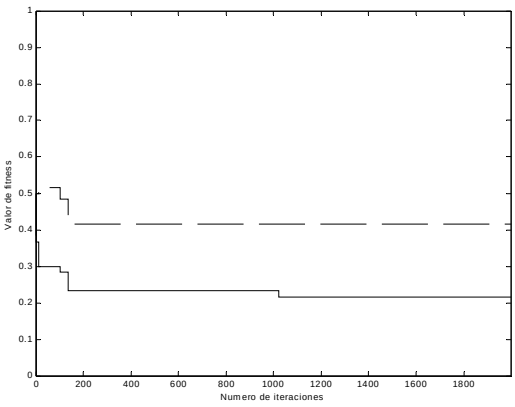
Error en entrenamiento: 20 %
Error en test: 35 %



Error en entrenamiento: 20 %
Error en test: 48.33 %



Error en entrenamiento: 18.33 %
Error en test: 38.33 %

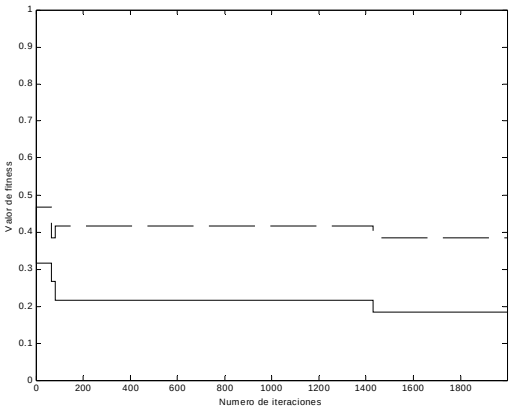


Error en entrenamiento: 21.67 %
Error en test: 41.67 %

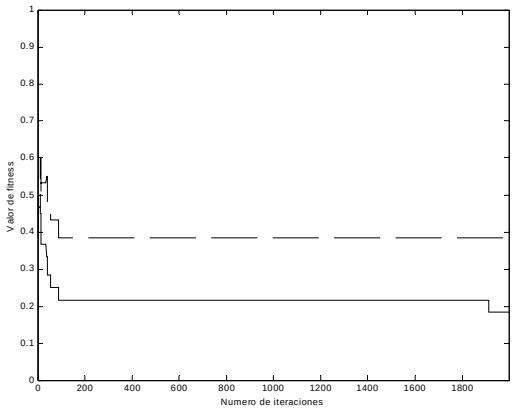
Figura 7.59. Curvas de test y entrenamiento para *min_iter* = 9.

Pruebas con *min_iter* = 15.

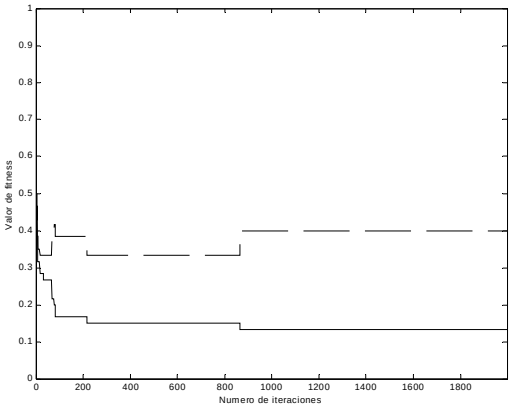
Pasadas 15 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 125 veces.



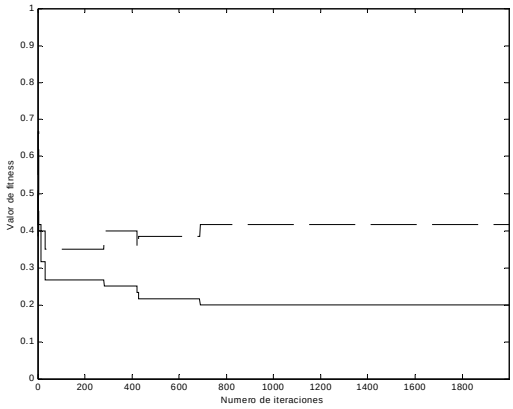
Error en entrenamiento: 18.33 %
Error en test: 38.33 %



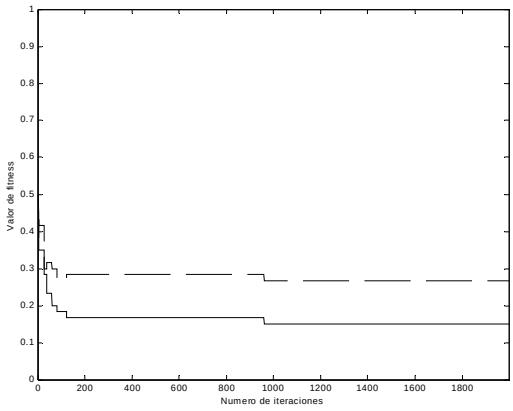
Error en entrenamiento: 18.33 %
Error en test: 38.33 %



Error en entrenamiento: 13.33 %
Error en test: 40 %



Error en entrenamiento: 20 %
Error en test: 41.67 %

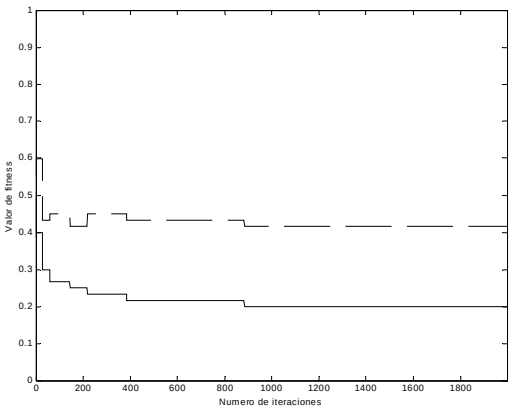


Error en entrenamiento: 15 %
Error en test: 26.67 %

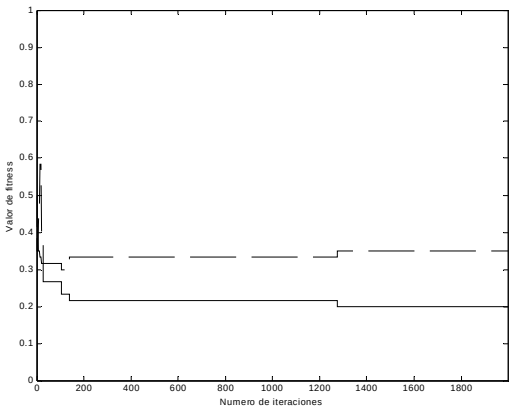
Figura 7.60. Curvas de test y entrenamiento para *min_iter* = 15.

Pruebas con *min_iter* = 70.

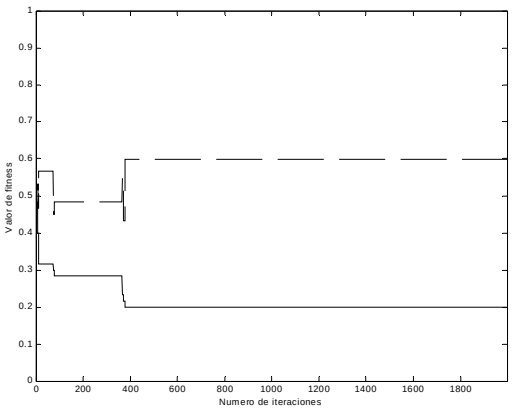
Pasadas 70 iteraciones desde el último algoritmo genético que se ejecutó, se dispara un nuevo algoritmo genético. Para todas estas pruebas el algoritmo genético se dispara 29 veces.



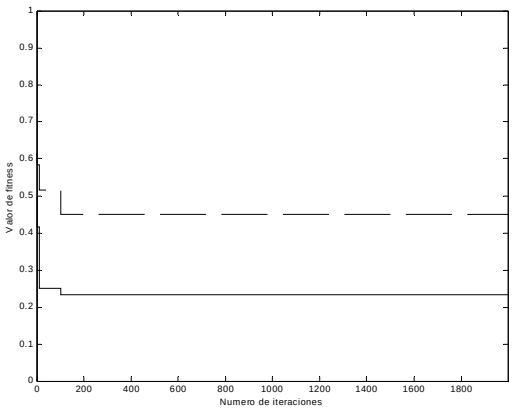
Error en entrenamiento: 20 %
Error en test: 41.67 %



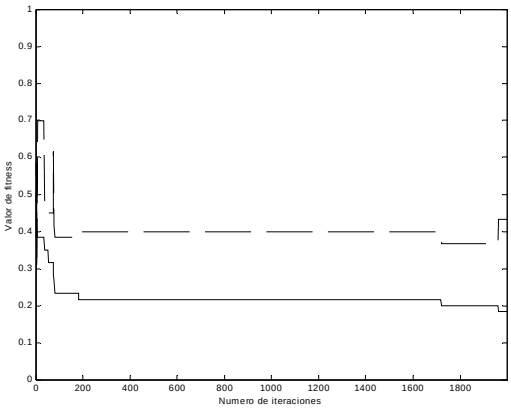
Error en entrenamiento: 20 %
Error en test: 35 %



Error en entrenamiento: 20 %
Error en test: 60 %



Error en entrenamiento: 23.33 %
Error en test: 45 %



Error en entrenamiento: 18.33 %
Error en test: 43.33 %

Figura 7.61. Curvas de test y entrenamiento para *min_iter* = 70.

7.6.5.4 Discusión de resultados.

El objetivo de esta prueba es analizar la influencia que tiene la frecuencia de los algoritmos genéticos sobre la calidad del entrenamiento. Para ello, se controla que el disparo de los algoritmos genéticos se produzca a un ritmo fijo y se deja evolucionar al algoritmo durante más iteraciones que en pruebas anteriores (2000 iteraciones).

Como se aprecia en la figura 7.56, la prueba que presenta mejor comportamiento en la clasificación del conjunto de test es la correspondiente a *min_iter* = 15, en la que el algoritmo genético se dispara 125 veces. Hay coincidencia con los resultados obtenidos para 600 iteraciones, mostrando que existen valores privilegiados para el parámetro *min_iter* en los que el sistema tiende a comportarse mejor en relación a los datos de entrenamiento y a los datos de test.

Se confirma por tanto la existencia de valores privilegiados de la frecuencia de disparo del algoritmo genético, y una tendencia al empeoramiento en cuanto a valor medio del error y mayor variabilidad en el comportamiento hacia altas frecuencias de disparo del algoritmo genético y hacia un empeoramiento del error medio en el entrenamiento y en el test si la frecuencia de disparo se disminuye desde estos valores privilegiados.

7.6.6 Experimentos mediante búsqueda aleatoria.

7.6.6.1 Descripción del experimento.

Se pretenden clasificar las series de datos simuladas basadas en HMM con máquinas finitas de estado borrosas. Las series temporales a clasificar son exactamente las mismas que se usaron en el experimento anterior.

En esta prueba, se realizará una búsqueda aleatoria de las máquinas finitas de estado borrosas que componen el clasificador. El objetivo es comparar los resultados de esta búsqueda aleatoria con las máquinas ya encontradas mediante el aprendizaje evolutivo tipo Michigan y así situar de forma relativa la mejora en la capacidad de búsqueda proporcionada por las características del sistema Michigan.

Se realizan 4 procesos de búsqueda puramente aleatoria durante 600 iteraciones (600 evaluaciones de una máquina de estados borrosa) y una búsqueda puramente

aleatoria durante 25000 iteraciones (25000 evaluaciones de una máquina de estados borrosa).

7.6.6.2 Resumen de resultados.

En la figura 7.62 se muestra la gráfica correspondiente a la media del error de entrenamiento para diferentes frecuencias de disparo del algoritmo genético hasta 600 iteraciones, junto con el error medio de entrenamiento y test en la búsqueda aleatoria limitada a 600 iteraciones (líneas horizontales discontinuas) y el error medio de entrenamiento y test en la búsqueda aleatoria limitada a 25000 iteraciones. Se puede apreciar como la búsqueda aleatoria con 600 iteraciones es peor que el sistema de Michigan con este mismo número de iteraciones. Hay que tener en cuenta que las primeras 600 iteraciones se corresponden con el comienzo de la curva de aprendizaje.

La búsqueda aleatoria con 25000 evaluaciones resulta ser mejor que las pruebas realizadas con 600 iteraciones en el sistema tipo Michigan. De hecho, podemos situar estas primeras 600 iteraciones del sistema Michigan estudiado en la franja comprendida entre la búsqueda aleatoria con 600 y 25000 iteraciones respectivamente.

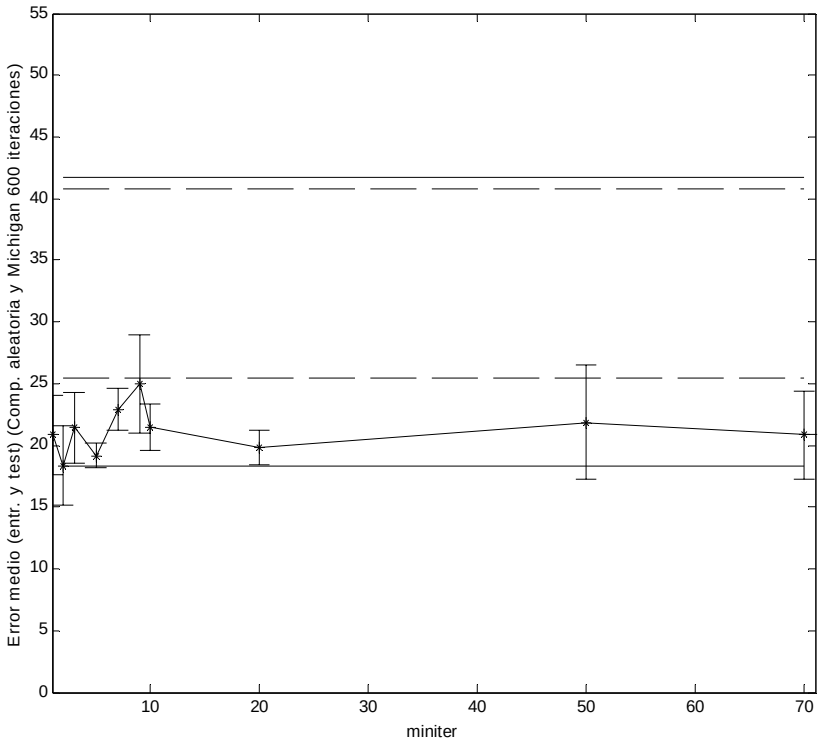


Figura 7.62. Comparativa de las pruebas realizadas hasta 600 iteraciones con el sistema tipo Michigan y búsqueda aleatoria (600 iteraciones - - y 25000 iteraciones -).

Sin embargo, ya vemos en la figura 7.63, que si aumentamos el número de iteraciones del sistema Michigan hasta 2000 iteraciones, alcanzamos los resultados equivalentes de la búsqueda aleatoria con 25000 evaluaciones de la máquina de estados borrosa (un factor 10 en el número de evaluaciones).

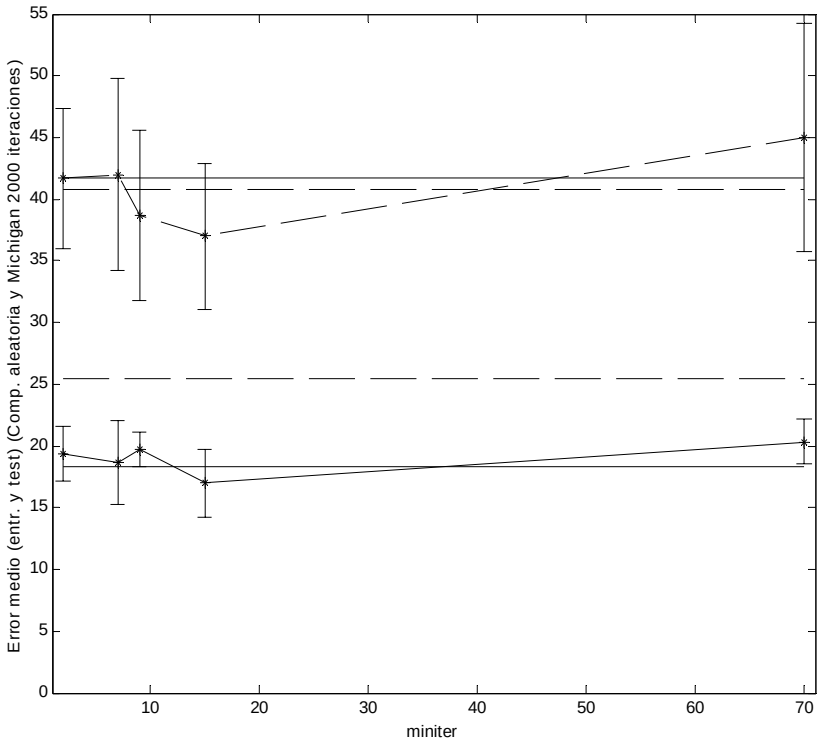


Figura 7.63. Comparativa de las pruebas realizadas hasta 2000 iteraciones con el sistema tipo Michigan y la búsqueda aleatoria (600 iteraciones - - y 25000 iteraciones -).

7.6.7 Estudio del sistema Michigan en relación al número de muestras en la serie temporal.

7.6.7.1 Descripción del experimento.

El objetivo es estudiar la eficiencia de las máquinas Michigan en comparación con la eficiencia del algoritmo basado en la identificación mediante Baum-Welch, en función del número de muestras por serie temporal. Para ello se generan el mismo número de trazas para cada uno de los dos modelos, utilizando un número diferente de muestras por serie temporal para cada experimento. Las series temporales generadas por cada uno de los HMM se agruparán en distintos conjuntos (conjuntos de entrenamiento y conjuntos de test).

Se realiza la identificación de los modelos 1 y 2 con el algoritmo de Baum-Welch utilizando los conjuntos de entrenamiento correspondientes a los distintos experimentos. Una vez realizada la identificación de los dos modelos, se clasifican las series temporales pertenecientes a los conjuntos de entrenamiento y de test asociándolas a uno u otro de los modelos identificados, a partir del cálculo de la probabilidad $P(O \mid \lambda)$.

Al mismo tiempo, se pretende realizar la búsqueda de una máquina finita de estado borrosas con un sistema tipo Michigan para clasificar las conjuntos de series temporales con diferentes longitudes de datos asociadas. Ya hemos comprobado que el sistema basado en Baum-Welch mejora su eficiencia a medida que aumenta el número de muestras por traza, al igual que la máquina de estados borrosa obtenida a partir del enfoque de Pittsburgh. El objetivo ahora es comparar los resultados de este algoritmo con la clasificación realizada por máquinas encontradas mediante el aprendizaje evolutivo tipo Michigan bajo las mismas condiciones, es decir, variando del mismo modo el número de muestras por serie temporal.

Con los conjuntos de entrenamiento de cada experimento se realizan tres procesos de entrenamiento tipo Michigan, y con los conjuntos de test de cada experimento se evaluarán las distintas máquinas encontradas en cada entrenamiento.

Finalmente, se comparará el error cometido por los clasificadores implementados con las máquinas finitas de estado borrosas con el error cometido en la clasificación mediante los modelos reconstruidos por el algoritmo de Baum-Welch.

Los parámetros del sistema de tipo Michigan son los mismos que los usados en los anteriores experimentos. Se realizará el disparo del algoritmo genético a frecuencia constante con un valor para el parámetro *min_iter* = 15. Este valor del parámetro fue el que mejores resultados proporcionó en los experimentos previos con series temporales de longitud 15.

El número de iteraciones realizadas en cada prueba varía. El motivo es que se han realizado los experimentos de manera que el error alcanzado de entrenamiento sea similar al error de entrenamiento alcanzado por los sistemas tipo Pittsburgh evaluados con los mismos conjuntos de entrenamiento.

7.6.7.2 Resumen de resultados.

En la tabla 7.17 se muestra una comparación entre los resultados de clasificación de las máquinas tipo Michigan y los resultados de clasificación de las reconstrucciones de Baum-Welch obtenidos para cada experimento. En la figura 7.64 se muestra una representación gráfica de estos datos. Los modelos mencionados en la tabla se corresponden a: HMM 1: 30 muestras /secuencia, HMM 2: 45 muestras /secuencia, HMM 3: 70 muestras /secuencia, HMM 4: 100 muestras /secuencia.

	Error en el entr. con Baum- Welch	Error en el test con Baum- Welch	Media del error en el entr. de las máquinas Michigan	Desviación del error en el entr.	Media del error en el test de las máquinas Michigan	Desv. del error en el test
HMM 1	20 %	23.33 %	15.56 %	0.96	23.34 %	2.89
HMM 2	23.33 %	26.67 %	13.89 %	0.96	29.45 %	4.81
HMM 3	23.33 %	25 %	11.11 %	3.47	20.56 %	1.93
HMM 4	11.67 %	8.33 %	9.44 %	0.96	16.11 %	4.81

Tabla 7.17. Resultados comparativos Baum-Welch y sistema tipo Michigan para diferentes longitudes de las series de datos.

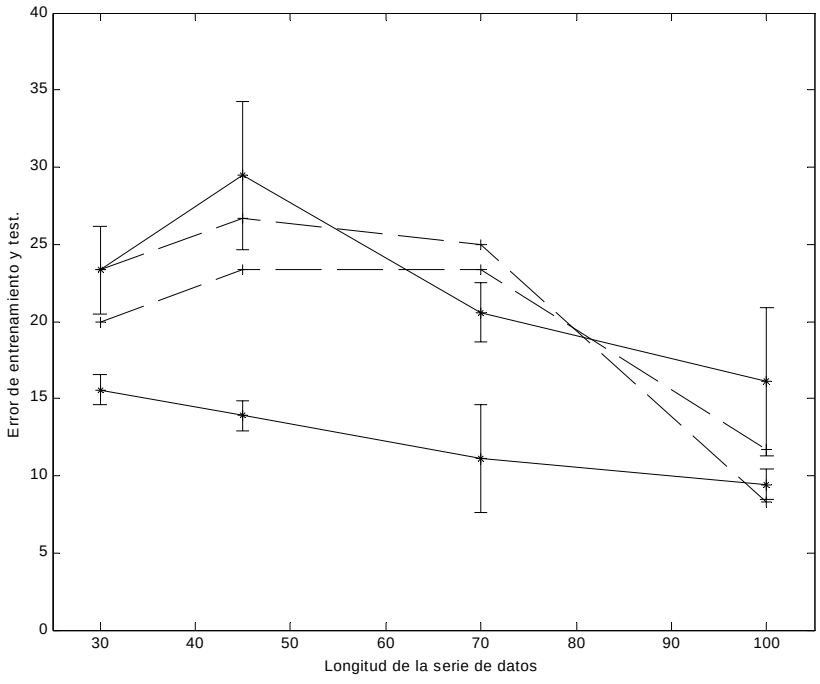
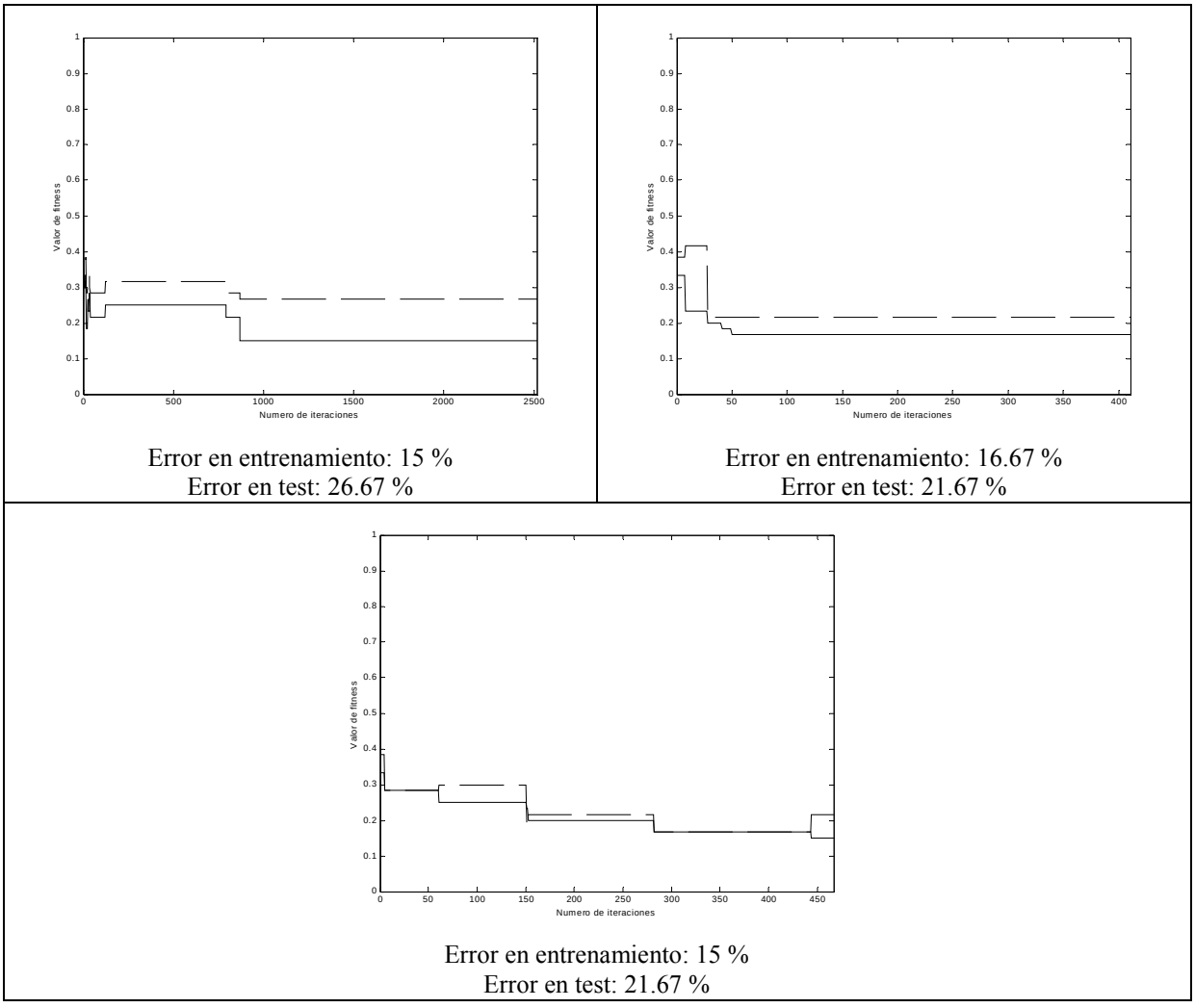


Figura 7.64. Representación gráfica de los datos de la tabla 7.13. En trazo continuo, error medio en el entrenamiento (gráfica inferior) y test (gráfica superior) para el sistema Michigan. En trazo discontinuo se presentan los errores para el conjunto de entrenamiento y conjunto de test para el algoritmo de Baum-Welch.

7.6.7.3 Curvas de entrenamiento y test.

Pruebas con 30 muestras por secuencia.

Se generan 60 secuencias con 30 muestras por secuencia para el modelo 1 y 60 secuencias con 30 muestras por secuencia para el modelo 2. El conjunto de entrenamiento está compuesto por 60 trazas de 30 muestras cada una, 30 del modelo 1 y 30 del modelo 2. El conjunto de test está compuesto por 60 trazas de 30 muestras cada una, 30 del modelo 1 y 30 del modelo 2 de las no utilizadas para construir el conjunto de entrenamiento. En la figura 7.65 se muestran las curvas de cada prueba.



Error en entrenamiento: 15 %
Error en test: 26.67 %

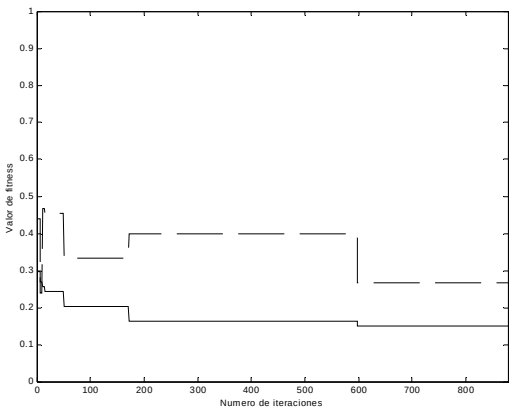
Error en entrenamiento: 16.67 %
Error en test: 21.67 %

Error en entrenamiento: 15 %
Error en test: 21.67 %

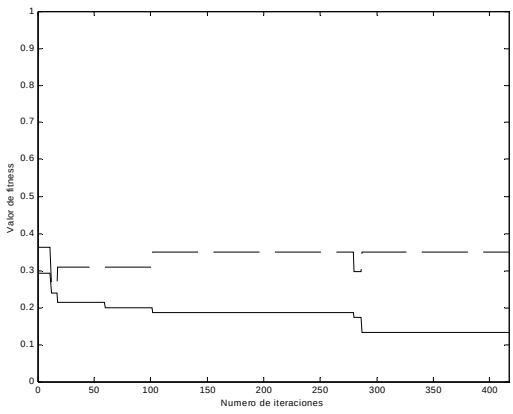
Figura 7.65. Curvas de entrenamiento (-) y test (--) para series temporales de longitud 30.

Pruebas con 45 muestras por secuencia.

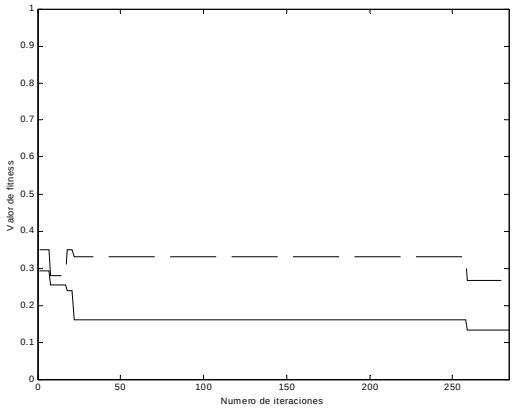
Se generan 60 secuencias con 45 muestras por secuencia para el modelo 1 y 60 secuencias con 45 muestras por secuencia para el modelo 2. El conjunto de entrenamiento está compuesto por 60 trazas de 45 muestras cada una, 30 del modelo 1 y 30 del modelo 2. El conjunto de test está compuesto por 60 trazas de 45 muestras cada una, 30 del modelo 1 y 30 del modelo 2 de las no utilizadas para construir el conjunto de entrenamiento. En la figura 7.66 se muestran las curvas de estas pruebas.



Error en entrenamiento: 15 %
Error en test: 26.67 %



Error en entrenamiento: 13.33 %
Error en test: 35 %

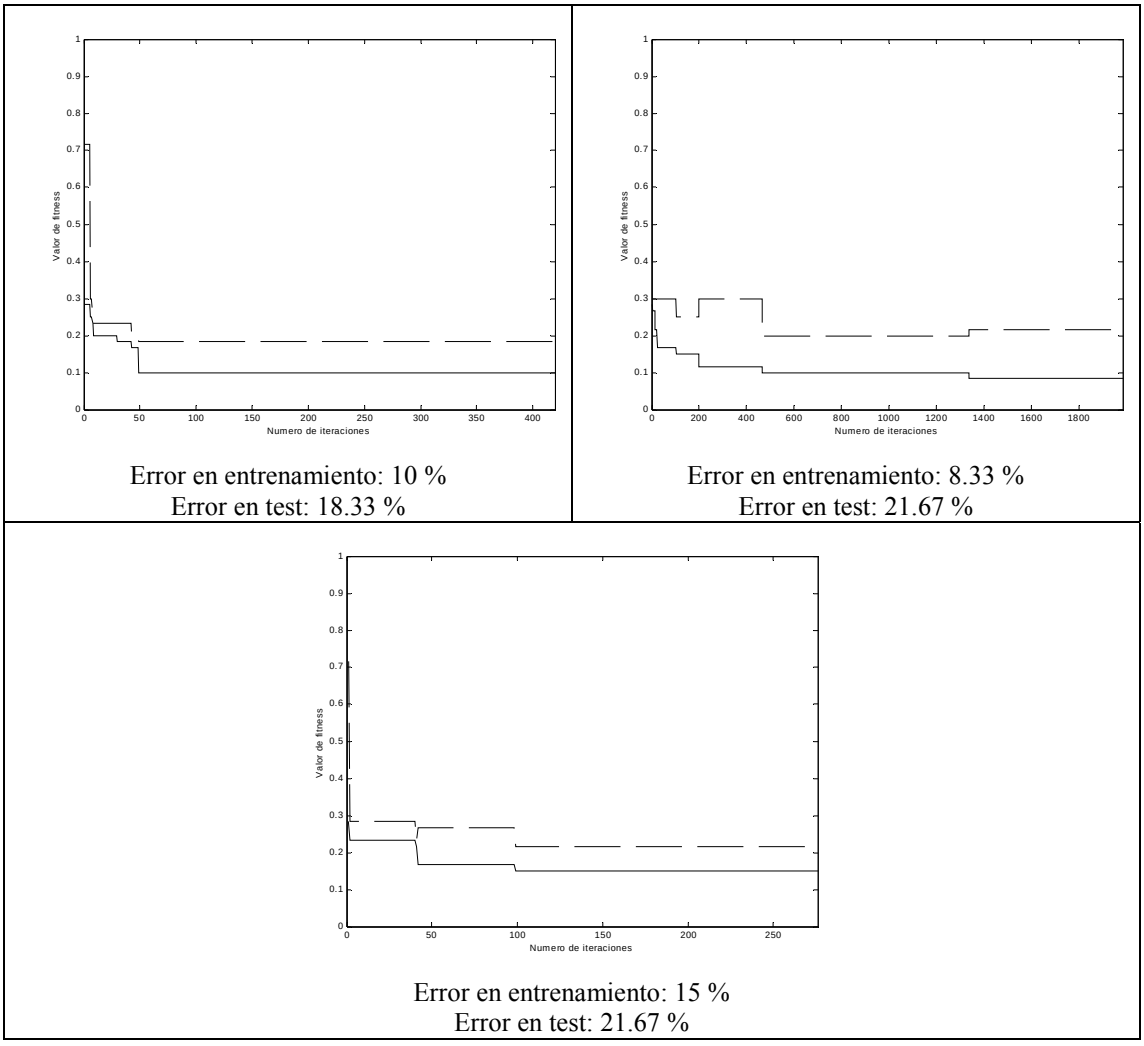


Error en entrenamiento: 13.33 %
Error en test: 26.67 %

Figura 7.66. Curvas de entrenamiento (-) y test (--) para series temporales de longitud 45.

Pruebas con 70 muestras por secuencia.

Se generan 60 secuencias con 70 muestras por secuencia para el modelo 1 y 60 secuencias con 70 muestras por secuencia para el modelo 2. El conjunto de entrenamiento está compuesto por 60 trazas de 70 muestras cada una, 30 del modelo 1 y 30 del modelo 2. El conjunto de test está compuesto por 60 trazas de 70 muestras cada una, 30 del modelo 1 y 30 del modelo 2 de las no utilizadas para construir el conjunto de entrenamiento. En la figura 7.67 se muestran las curvas de estas pruebas.



Error en entrenamiento: 10 %
Error en test: 18.33 %

Error en entrenamiento: 8.33 %
Error en test: 21.67 %

Error en entrenamiento: 15 %
Error en test: 21.67 %

Figura 7.67. Curvas de entrenamiento (-) y test (--) para series temporales de longitud 70.

Pruebas con 100 muestras por secuencia.

Se generan 60 secuencias con 100 muestras por secuencia para el modelo 1 y 60 secuencias con 100 muestras por secuencia para el modelo 2. El conjunto de entrenamiento está compuesto por 60 trazas de 100 muestras cada una, 30 del modelo 1 y 30 del modelo 2. El conjunto de test está compuesto por 60 trazas de 100 muestras cada una, 30 del modelo 1 y 30 del modelo 2 de las no utilizadas para construir el conjunto de entrenamiento. En la figura 7.68 se muestran las curvas de estas pruebas.

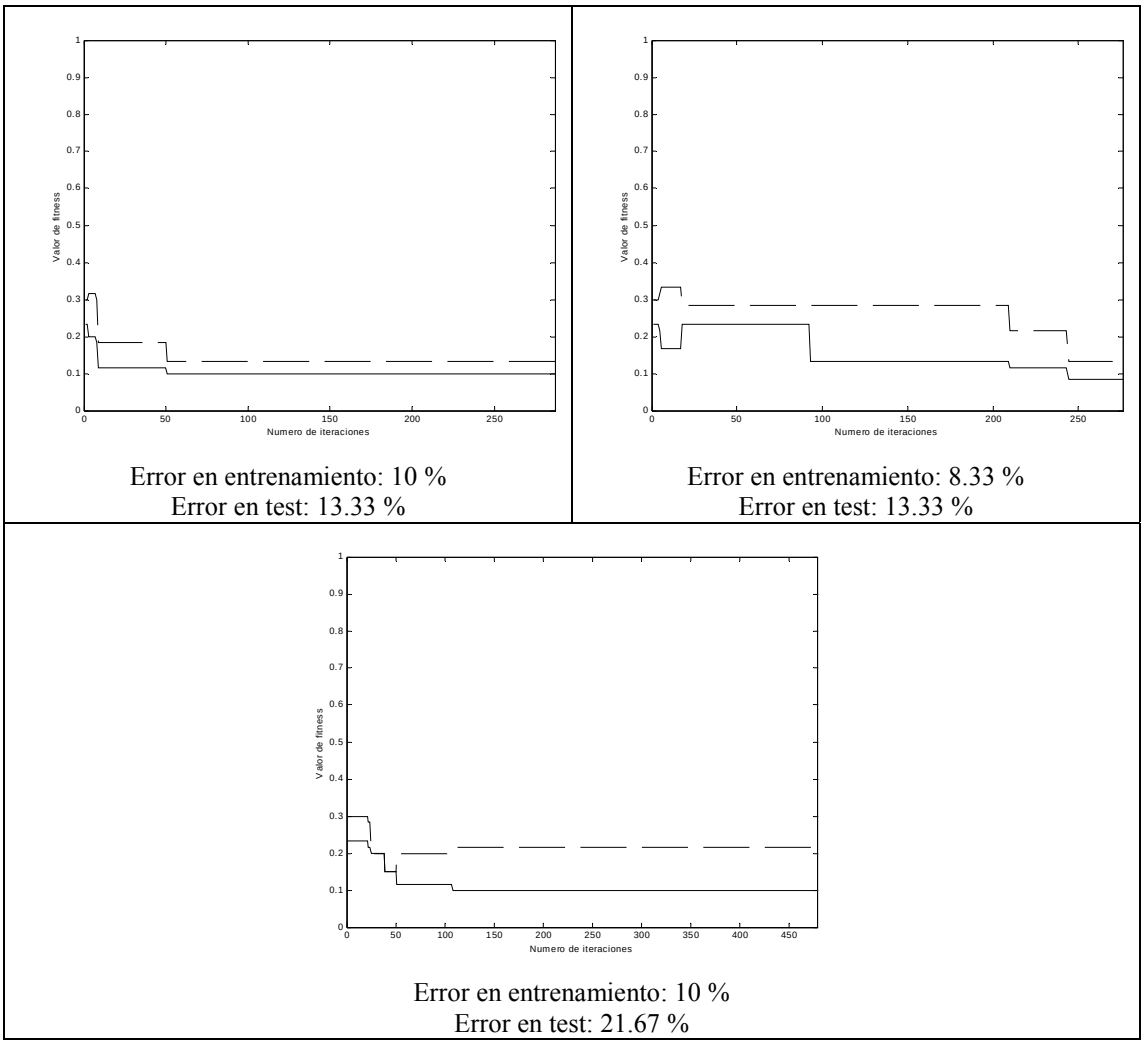


Figura 7.68. Curvas de entrenamiento (-) y test (--) para series temporales de longitud 100.

7.6.7.4 Discusión de resultados.

Se puede comprobar que las máquinas llegan a tener valores de eficiencia muy similares a los obtenidos con el algoritmo de Baum-Welch, y en algunos casos, ofrecen mejores resultados de clasificación en el conjunto de entrenamiento, en el conjunto de test o en

ambos conjuntos a la vez. De la figura 7.64 se deduce que con longitudes medias para la serie de datos, el sistema tipo Michigan ajusta mejor los datos de entrenamiento y presenta errores similares a Baum-Welch para el conjunto de test. Sin embargo, cuando la serie temporal se hace suficientemente larga, el algoritmo de Baum-Welch mejora claramente con un error en el entrenamiento y test que tiende a 0.

Recordemos que el algoritmo de Baum-Welch está específicamente diseñado para identificar modelos ocultos de Markov, por lo tanto, tiene ventajas sobre las máquinas Michigan en este problema concreto de clasificación. Por este motivo, son tan destacables los resultados de las máquinas que tienen mejor eficiencia que el algoritmo de Baum-Welch, para series temporales de tamaño medio. Recordemos que lo que se pretende con los resultados de la clasificación obtenidos con el algoritmo de Baum-Welch es tener una medida de eficiencia de referencia con la que comparar los resultados obtenidos en la clasificación con las máquinas de estados borrosas.

Si comparamos estos resultados con los obtenidos para el sistema tipo Pittsburgh con 50 iteraciones, veremos que los errores obtenidos con el conjunto de test son similares.

Es importante observar que las curvas de test siguen mejor a las de entrenamiento a medida que aumenta el número de muestras por trazas, lo que indica que se aprende el modelo subyacente en los datos. El número de iteración del algoritmo en el que se consigue un error de entrenamiento similar al correspondiente para Pittsburgh varía según la prueba realizada, no obstante hay que recordar que el concepto de iteración para los sistemas tipo Michigan es diferente que para los sistemas tipo Pittsburgh: en el primer caso sólo tendremos que evaluar una máquina de estados borrosa mientras que en el segundo de los casos hay que evaluar una población completa formada en los experimentos realizados por 200 máquinas de estado borrosas. De esta forma, y examinando las curvas de entrenamiento, podemos encontrar una ventaja notable desde el punto de vista computacional en los sistemas tipo Michigan, al menos en lo que se refiere a encontrar soluciones similares a las obtenidas por los sistemas tipo Pittsburgh en 50 iteraciones en las condiciones experimentales establecidas.

7.7 Conclusiones.

En este capítulo se ha presentado un estudio para contrastar la capacidad de los algoritmos propuestos a la hora de clasificar series temporales basadas en modelos de Markov ocultos. Las principales conclusiones que se han podido extraer de las pruebas realizadas son:

- Ambos sistemas, enfoques Pittsburgh y Michigan, pueden ser entrenados y generalizar a partir de conjuntos de entrenamiento basados en modelos de Markov donde las clases se diferencian en las matrices de probabilidad de transición.

- Al igual que ocurre con la clasificación basada en la identificación del modelo por el método de Baum-Welch, la longitud de la serie temporal juega un papel importante que debe ser tenido en cuenta a la hora de analizar los resultados del algoritmo de clasificación.

- Las máquinas borrosas obtenidas en los sistemas tipo Pittsburgh y Michigan tienden a obtener mejores errores de entrenamiento y errores similares de test que el algoritmo basado en la identificación del HMM, para secuencias de longitud intermedia (30 – 70 muestras en los experimentos con los modelos utilizados).

- El solape entre poblaciones utilizado en el sistema tipo Pittsburgh, permite aliviar la carga computacional, pero hay que establecer valores adecuados en el porcentaje de solapamiento, ya que si estos son demasiado grandes el problema de la convergencia prematura tiende a empeorar los resultados de entrenamiento y test.

- El operador de mutación juega un papel relevante en el tipo de sistema Pittsburgh implementado, especialmente si se tiene en cuenta que la utilización de poblaciones solapadas afecta ya a la diversidad de la población.

- El ritmo de disparo del algoritmo genético en el sistema tipo Michigan adquiere una frecuencia irregular si se utiliza como método de disparo un número mínimo de meta-reglas del conjunto de encaje que sobrepasen un umbral de antigüedad.

- Si se fuerza una frecuencia regular de disparo para el algoritmo genético en el sistema tipo Michigan, se observan valores privilegiados en donde se incrementa la probabilidad de obtener entrenamientos aceptables y mejores valores para el error de entrenamiento y el error de test.

- La situación relativa del algoritmo tipo Michigan respecto a un algoritmo de búsqueda aleatoria simple ha servido para demostrar la validez de sus mecanismos de búsqueda más allá de la pura exploración del espacio de búsqueda. El sistema de búsqueda aleatoria simple requiere de unas 25000 evaluaciones para obtener un resultado similar al obtenido con el sistema Michigan transcurridas 2000 iteraciones.

En resumen, este estudio supone una validación de la hipótesis de que el tipo de clasificador propuesto basado en la máquina de estados borrosa puede ser utilizado en la clasificación de series temporales basadas en procesos de tipo Markov.

Los principales inconvenientes encontrados en la utilización de los algoritmos son:

- La complejidad computacional del proceso de búsqueda, aunque este extremo se ve aliviado en gran medida con la utilización de sistemas tipo Michigan.

- La adecuada elección de los parámetros para el sistema. En este caso, es el sistema Michigan el que más problemas puede presentar dado el gran número de parámetros a establecer.

En el próximo capítulo, veremos como el clasificador basado en la máquina de estados borrosa puede ofrecer ventajas respecto a Baum-Welch en la clasificación de series temporales de datos reales, donde el número de datos por serie temporal no será elevado y donde los datos no responden exactamente a un modelo HMM.

Capítulo 8

Clasificación de datos reales mediante máquinas de estados borrosas: aplicación al análisis de imágenes de citologías.

8.1 Introducción.

La presente tesis aborda la problemática del reconocimiento de patrones con máquinas finitas de estados borrosas. Mediante los métodos evolutivos de búsqueda ya descritos (sistemas tipo Pittsburgh y sistemas tipo Michigan), se pretende encontrar una máquina óptima que sea capaz de reconocer distintos patrones. En el capítulo anterior, para verificar y validar estos clasificadores, se aplicaron las máquinas al reconocimiento de patrones simulados (reconocimiento de series temporales generadas por modelos ocultos de Markov). En este capítulo se aplica esta metodología a un problema real: el reconocimiento de patrones en imágenes de citologías médicas. Además, se compara la

efectividad de este método como clasificador de patrones con otros métodos ya existentes, tanto supervisados (redes neuronales con propagación hacia delante y el algoritmo de Baum-Welch) como no supervisados (clustering borroso), y se realiza una evaluación de los clasificadores obtenidos desde el punto de vista médico mediante el análisis de las curvas ROC.

Existen varios métodos para la detección del cáncer. La biopsia (método quirúrgico) es el más eficaz, pero es invasivo, costoso y consume mucho tiempo. Los sistemas de diagnóstico basados en el análisis de imágenes digitales pueden permitir un diagnóstico muy aproximado sin necesidad de intervenciones quirúrgicas, y por lo tanto, son métodos muy utilizados en la práctica médica. La citología es uno de estos métodos. Nuestro objetivo es clasificar correctamente núcleos de células sanas y núcleos de células patológicas en imágenes digitalizadas de citologías. La característica que se ha utilizado para realizar esta clasificación es la distribución de cromatina en el núcleo, que es el factor que determina el aspecto visual de la textura del mismo.

La clasificación de muestras de tejido y citologías, que se desarrolla en buena parte por inspección visual mediante el microscopio óptico, puede ser mejorada mediante la utilización de técnicas de análisis y procesamiento digital de imágenes junto a métodos de extracción y selección de características y diseño de clasificadores. Dentro de este procedimiento de automatización juegan un papel relevante las técnicas de clasificación de texturas, desde el punto de vista del diagnóstico y la prognosis a nivel nuclear y a nivel del tejido.

La utilización de clasificadores que igualan y mejoran los resultados obtenidos por inspección visual queda patente en multitud de publicaciones, patentes y equipos comerciales, por ejemplo, en [Rodenacker, 2001], [Burger et al., 1981], [Weyn et al., 1999].

En este trabajo, la primera aproximación a este problema de reconocimiento de patrones y clasificación fue realizada con imágenes de cáncer de mama, imágenes correspondientes a la prueba médica de aspiración por aguja fina (en inglés, *Fine Needle Aspirate* – FNA) [Estévez et al., 2002a]. Las imágenes utilizadas en estos experimentos preliminares fueron tomadas de la base de datos de cáncer de mama de la Universidad de Wisconsin [Wolberg, 1992], una base de imágenes diseñada para la validación de algoritmos de clasificación, publicada por el Dr. William H. Wolberg de la Universidad de Wisconsin.

Tras estos experimentos iniciales, se comprobó la validez de la metodología de clasificación propuesta en esta tesis. A partir de este momento, en todos los experimentos realizados se cuenta con la colaboración del Dr. D. Lucio Díaz Flores y de su equipo investigador del Departamento de Anatomía Patológica del Hospital Universitario de Canarias. Este equipo nos cede las imágenes de citologías digitalizadas para esta investigación y nos proporciona asesoramiento y ayuda en su clasificación, desde su conocimiento experto del dominio. Las imágenes analizadas bajo supervisión de estos expertos son imágenes de citologías de fluidos peritoneales y pleurales.

El objetivo de esta última parte de investigación realizada no es diseñar un sistema clasificador de células malignas y benignas en citologías, sino comprobar que el sistema recurrente borroso que se ha investigado es sensible a series de datos reales que describen la distribución de cromatina en el núcleo celular. La creación de un sistema clasificador requiere de la integración de muchas más características celulares, por lo que en el futuro se pretende analizar la capacidad discriminadora de las máquinas de estados borrosas actuando en conjunción con otros parámetros típicos como la relación núcleo-citoplasma y otros parámetros relativos a la textura.

Los motivos por los que se ha elegido esta aplicación real son varios. En primer lugar, la trayectoria del grupo de investigación en el que se enmarca este trabajo se caracteriza por un gran número de colaboraciones con otros grupos del área de la medicina, existiendo una experiencia importante de la que partir [Sigut, 2001], [Moreno et al., 2001a], [Moreno et al., 2001b], [Moreno et al., 2000], [Moreno et al, 1995a], [Moreno et al., 1995b], [Piñeiro et al., 2002], [Piñeiro et al., 2001], [Piñeiro et al., 2000], [Piñeiro et al., 1998a], [Sánchez, 1993]. En segundo lugar, porque la utilización de series de datos para describir características espaciales globales en los núcleos celulares es una idea novedosa que a priori parecía tener ciertas posibilidades. Y, por último, por la trascendencia social del problema investigado.

Encontrar estructuras en series de datos es un problema bien conocido que encuentra aplicaciones en muchos campos donde el reconocimiento de patrones es necesario. Para este propósito, se han usado modelos estadísticos lineales (como el ARMAX) y no lineales (como las redes neuronales). Sin embargo, hemos elegido otra aproximación a este problema basado en los sistemas de inferencia borrosos, porque este método nos proporciona información simbólica sobre los motivos por los que las

texturas son clasificadas en una u otra categoría. Este hecho es muy importante en áreas como la automatización del diagnóstico médico.

A continuación, se exponen los principales problemas que surgen en el análisis y clasificación de los núcleos celulares en función de su distribución de cromatina en estas imágenes, las soluciones propuestas y los resultados de los experimentos ya comentados sobre las imágenes de cáncer de mama, peritoneo y pleura.

8.2 Descripción del problema.

Como se ha comentado en la introducción, en la presente investigación se pretende analizar y detectar patologías en imágenes digitales obtenidas a partir de microscopía óptica de citologías conjugando la utilización de técnicas ya firmemente establecidas en la literatura científica y la inclusión de aspectos novedosos en el procedimiento y en la aplicación de nuevas características y clasificadores. Es necesario estudiar las etapas que componen este proceso: definición inicial del problema de clasificación, segmentación, extracción de características, diseño de clasificadores y validación. En las siguientes secciones se hace una descripción detallada de la problemática que conlleva cada etapa y de las soluciones propuestas.

Por otra parte, es importante citar que en esta investigación la herramienta utilizada es Matlab, de la compañía Mathworks. Desde el punto de vista computacional, las soluciones propuestas requieren otro tipo de implementación, pero se ha elegido trabajar con Matlab debido a que, desde el punto de vista de la investigación, es una herramienta que facilita la depuración de los algoritmos y el análisis de los datos.

8.2.1 Definición inicial del problema.

La definición inicial del problema implica la realización de un proceso de selección con los especialistas de un conjunto de problemas de referencia a resolver. Una parte muy importante de este proceso es el establecimiento de grados de dificultad asociados a los problemas de referencia desde el punto de vista de la complejidad de los algoritmos y de los mejores resultados publicados en la literatura especializada. Esta clasificación es importante para facilitar el proceso de desarrollo, depuración, verificación y validación de algoritmos. Además, en nuestro caso concreto, se ha procurado incluir un grupo de

problemas especial donde mediante la literatura científica se constate la importancia de la descripción de la textura del núcleo como un aspecto importante en la clasificación, admitiendo descriptores locales como características a considerar en el proceso de clasificación.

Por otro lado, en la problemática asociada al empleo de los protocolos de análisis y clasificación de histologías y citologías, uno de los puntos principales es la reproducibilidad, que se ve afectada por las condiciones experimentales. Uno de los requerimientos es describir las condiciones experimentales asociadas a los detalles de la preparación de la muestra, el montaje óptico, adquisición de imágenes, etcétera, diferenciando entre las condiciones experimentales que se pueden fijar y aquellas que no. En este trabajo ha sido el equipo especialista el encargado de fijar estas condiciones experimentales, para facilitar la aplicación de los métodos propuestos.

Es importante recordar la importancia de tener en cuenta los métodos de preparación de las muestras, para establecer el tipo de procesamiento. Esto es especialmente importante en el caso de la utilización de marcadores inmunohistoquímicos, ya que estos reactivos permiten destacar componentes particulares en la célula o tejido mediante el cambio de alguna de sus características, de especial interés para nosotros el color.

8.2.2 El problema de la segmentación.

Es preciso, como paso previo a la realización de la extracción de características de los núcleos a clasificar, realizar el proceso denominado *segmentación*. Esto es, la medición de características de una imagen se establece tomando ciertas unidades fundamentales o sub-regiones correspondientes a diferentes tipos o estructuras. Ejemplos típicos de subregiones utilizadas en histometría y citometría son las denominadas fondo, célula, y dentro de ésta, el citoplasma y el núcleo. El proceso de segmentación establece entonces una clasificación de los píxeles de la imagen digital en estas subregiones.

El procedimiento más básico de segmentación se basa en la umbralización de una imagen en niveles de gris [Sahoo et al., 1988]. Los umbrales pueden ser fijos para toda la imagen (umbral estático) [Weszka, 1978], [Kittler y Illingworth, 1985], o variar dependiendo de la zona y características (umbral dinámico) [Chow y Kaneko, 1972], [Wu et al., 1995].

Los denominados algoritmos de crecimiento de regiones comienzan con un conjunto de píxeles semilla o regiones de crecimiento que son aumentadas añadiendo a una región píxeles que cumplen algún criterio de similaridad con la región [Adams y Bischof, 1994].

Los algoritmos división–unión (*split and merge*) comienzan dividiendo de forma progresiva la imagen en partes cada vez más pequeñas disjuntas hasta que se cumple un criterio de similaridad entre los píxeles que forman las subregiones. Entonces se aplica un procedimiento de reunificación entre regiones vecinas basándose en un criterio de homogeneidad [Chou et al., 1992].

La segmentación basada en la representación de la imagen en un espacio de colores también ha sido abordada por numerosos investigadores. En ese sentido y en el campo que nos ocupa, se hace especial uso de las propiedades de las tinturas empleadas en las preparaciones de las muestras, empleando técnicas de segmentación basadas en el umbral sobre una representación de la imagen donde se ha realizado una transformación de forma que cada píxel da cuenta de la importancia relativa de un color. Estas técnicas son muy dependientes de la preparación de la muestra, especialmente en el caso de utilización de marcadores inmunohistoquímicos. Otras técnicas más sofisticadas basadas en el color, son la descomposición en regiones recursiva usando discriminantes basados en el color [Ohta et al., 1980] o la utilización de la transformada de componentes principales y algoritmos de clustering ([Umbaugh et al., 1993], [Schmid y Fischer, 1997]).

Otra técnica de segmentación se basa en la clasificación de los píxeles de la imagen por métodos supervisados o no supervisados como el algoritmo k-means o el ISODATA [Duda y Hart, 1973]. En este tipo de técnicas a cada píxel se le debe asociar un vector de características [Ossen et al., 1994].

La estimación del gradiente en las imágenes también juega un papel importante en diversas técnicas de segmentación. En este sentido, es bastante común utilizar la derivada del operador Gaussiano como filtro para realizar esta estimación. Una vez realizada la estimación de la magnitud del gradiente en la imagen, se pueden emplear técnicas como el cálculo de los bordes o los algoritmos de segmentación basados en la transformada *watershed* [Haris et al., 1998], [Gauch, 1999]. En este último caso el gradiente de la imagen en cada píxel se considera como si fuera la altura de una superficie en 3D. Las regiones se forman simulando la “inundación” de dicha superficie

por un líquido. Esta inundación comienza en algunos mínimos locales seleccionados y avanza al ir rebasando las barreras establecidas por los máximos locales. Esta técnica presenta ventajas frente a las que se basan en localizar bordes, ya que se generan contornos cerrados. El resultado de la técnica suele ser una imagen sobresegmentada, por lo que a posteriori hay que emplear técnicas para unir subregiones. La transformada *watershed* también puede tener aplicación en la obtención de descriptores para las texturas como se explicará mas adelante.

Las técnicas de detección de bordes [Canny, 1986], [Marr y Hildreth, 1980], [Perona y Malik, 1990] presentan el inconveniente de producir contornos no cerrados y puntos falsos en donde es difícil discernir su pertenencia o no al contorno de alguna región.

Los algoritmos basados en búsqueda radial simplifican la tarea de la detección del borde al restringir esta búsqueda para cada punto del borde a una línea que parte de un punto preestablecido, tomando cada vez un ángulo que se incrementa progresivamente [Golston et al., 1990], [Jarkans et al., 1980].

Finalmente citaremos las técnicas denominadas de contornos activos. Estas técnicas fueron introducidas por [Kass et al., 1987]. El contorno activo representa el contorno de un objeto mediante una curva parametrizada que se puede deformar a partir de una posición y forma inicial hasta un contorno final. El problema de encontrar el contorno final es equivalente a un problema de minimización de energía. El funcional de la energía se basa en propiedades características de la imagen, de forma que cuando el contorno se deforme y llegue a un mínimo, esta deformación estará relacionada con el contenido de la imagen.

La segmentación en citologías preparadas con tintura es una tarea que puede ser acometida por métodos basados en la umbralización [Borst et al., 1979], y en el caso de muestras para análisis histométrico, la segmentación es una tarea donde se producirá la intervención del experto [Jütting et al., 1999], [Minkus et al., 1997], [Rodenacker et al., 1992].

En la presente investigación se ha implementado un método semiautomático para segmentar los núcleos de las imágenes estudiadas basado en contornos adaptativos. A continuación, se describe este método. Es importante destacar que se han utilizado métodos ya establecidos de segmentación, en donde a pesar de la automatización alcanzada, se deben combinar con la extracción manual de los núcleos. En general

hemos constatado la necesidad de una investigación más profunda en las técnicas de segmentación automática, ya que dada la dificultad presentada por las imágenes, este paso puede ser un cuello de botella de cara a la construcción de un sistema automatizado. Sin embargo, ya que nuestro máximo interés se centra en la aplicación de los sistemas borrosos recurrentes en la clasificación de los datos reales y no en el modo de segmentar las imágenes, no se ha profundizado en este aspecto.

El proceso comienza remuestreando la imagen con el objetivo de reducir su tamaño en un factor 0.1 (la imagen resultante es 0.1 veces la original). Para esto utilizamos la interpolación del vecino más cercano (*nearest neighbor interpolation*). Un ejemplo de este paso inicial se muestra en la figura 8.1.

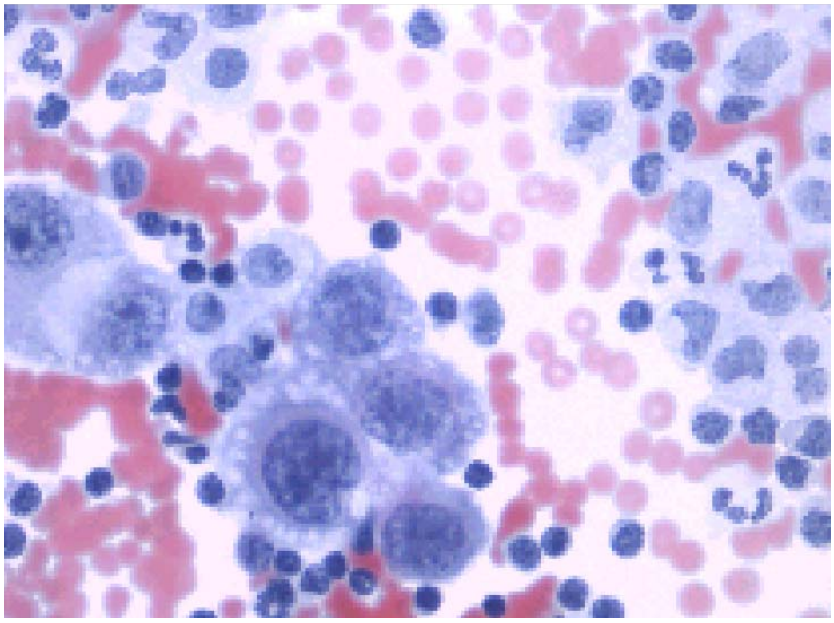


Figura 8.1. Imagen original disminuida.

Pasamos la imagen a escala de grises y la normalizamos (cada píxel tendrá un valor entre 0 y 1). El resultado se muestra en la figura 8.2.

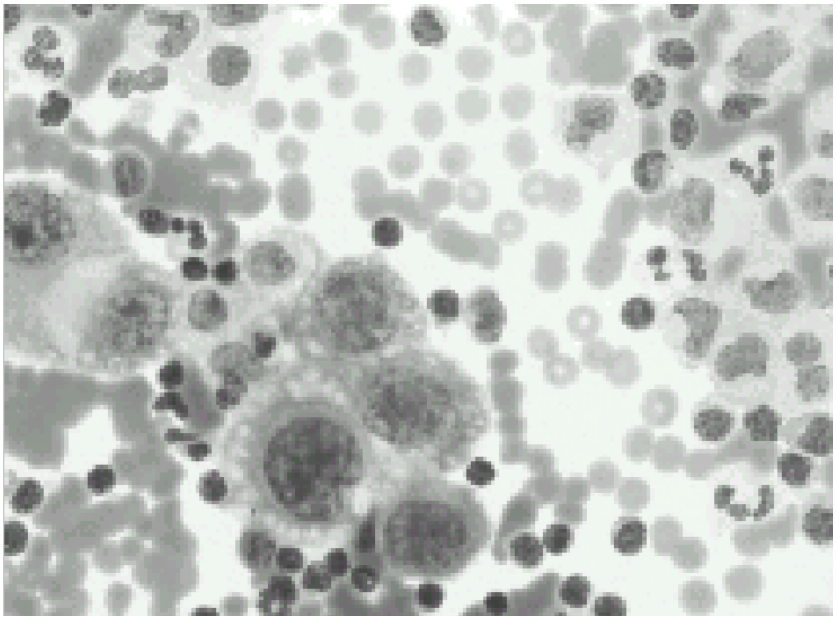


Figura 8.2. Imagen en escala de grises y normalizada.

En el siguiente paso realizamos un contraste (figura 8.3).

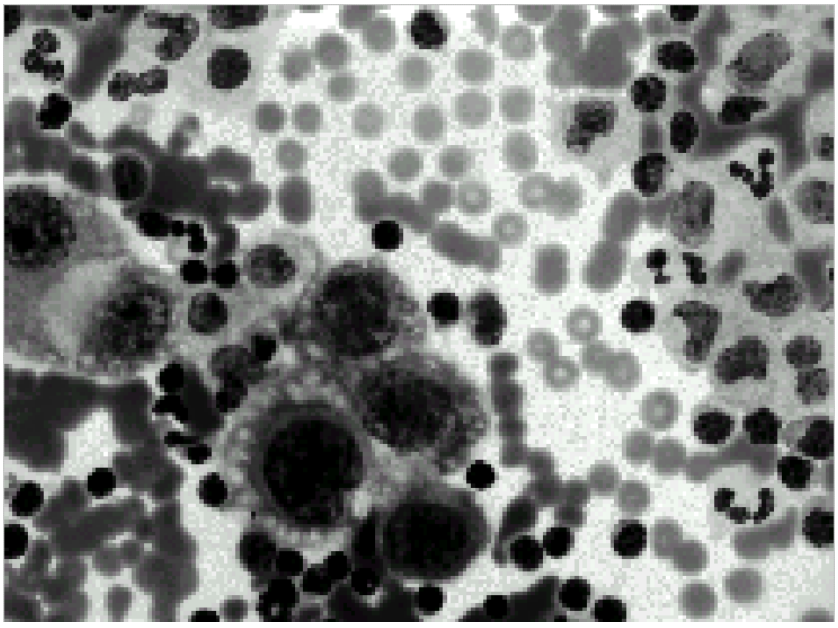


Figura 8.3. Imagen contrastada.

A esta imagen contrastada le aplicamos un filtro de difusión anisotrópico y repetimos el proceso de contraste para la imagen resultante, tal y como se puede apreciar en la figura 8.4.

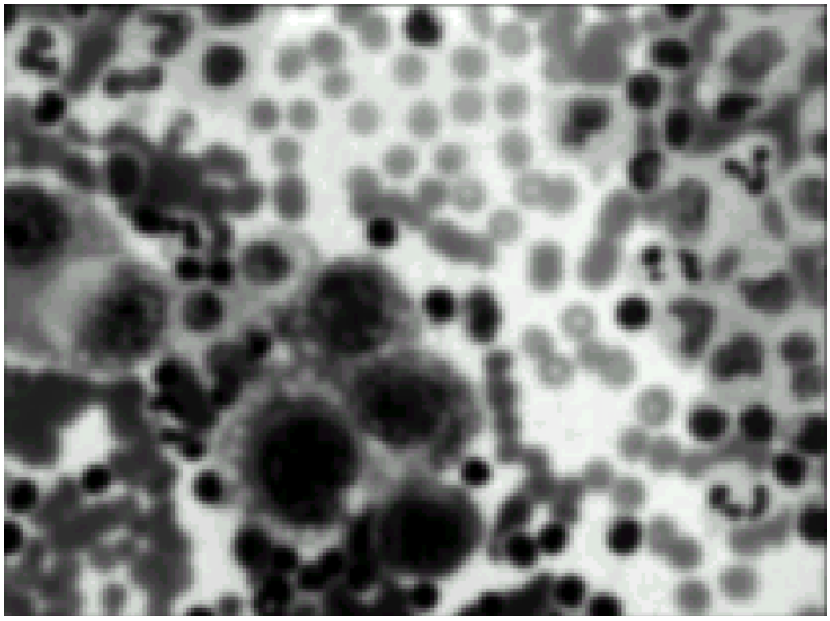


Figura 8.4. Imagen filtrada y contrastada.

Con el objetivo de destacar las células de interés, se lleva a cabo una umbralización (figura 8.5), para destacar en la imagen filtrada y contrastada las zonas de interés, dándoles un valor mayor que 1 a los píxeles correspondientes (figura 8.6). Finalmente, normalizamos de nuevo el resultado (figura 8.7).

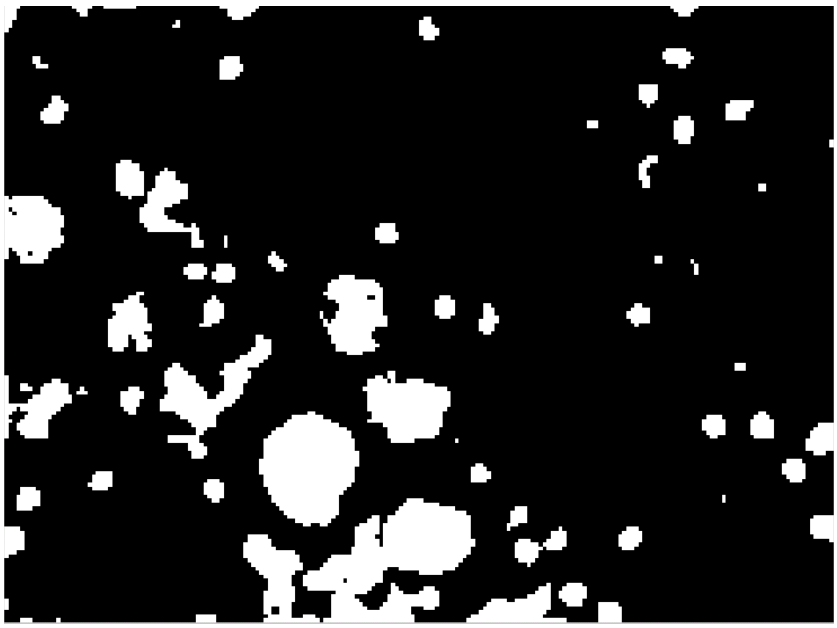


Figura 8.5. Imagen umbralizada.

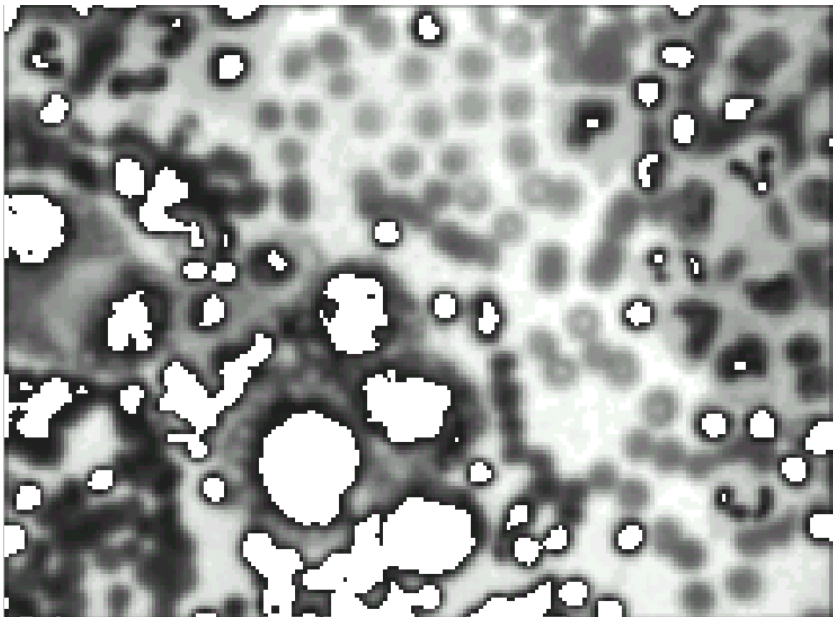


Figura 8.6. Imagen con zonas destacadas.

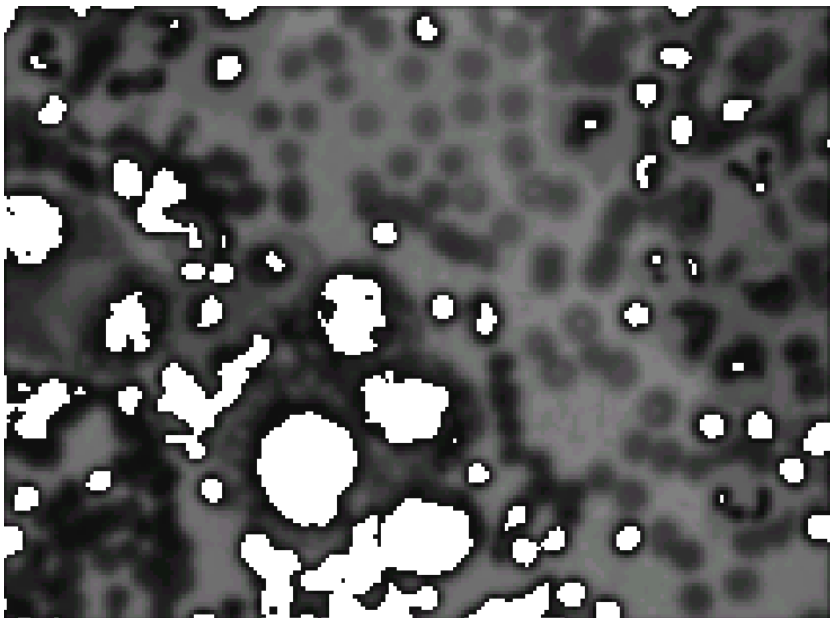


Figura 8.7. Imagen con zonas destacadas normalizada.

A esta última imagen le aplicamos un detector de bordes, en concreto el detector de Canny (figura 8.8), suavizamos estos bordes (figura 8.9) y estimamos el campo de fuerza existente en la imagen. Estos dos últimos pasos (suavización de bordes y estimación del campo de fuerza) son necesarios para la técnica de los contornos adaptativos.

Figura 8.8. Detección de bordes en la imagen.

Figura 8.9. Suavizado de bordes en la imagen.

En la siguiente etapa del proceso se requiere la intervención del usuario. Se muestra la imagen original con los bordes resaltados con el objetivo de que el usuario

seleccione con el cursor los núcleos que desea extraer para un posterior estudio. En la figura 8.10 se muestra la imagen ofrecida al usuario.

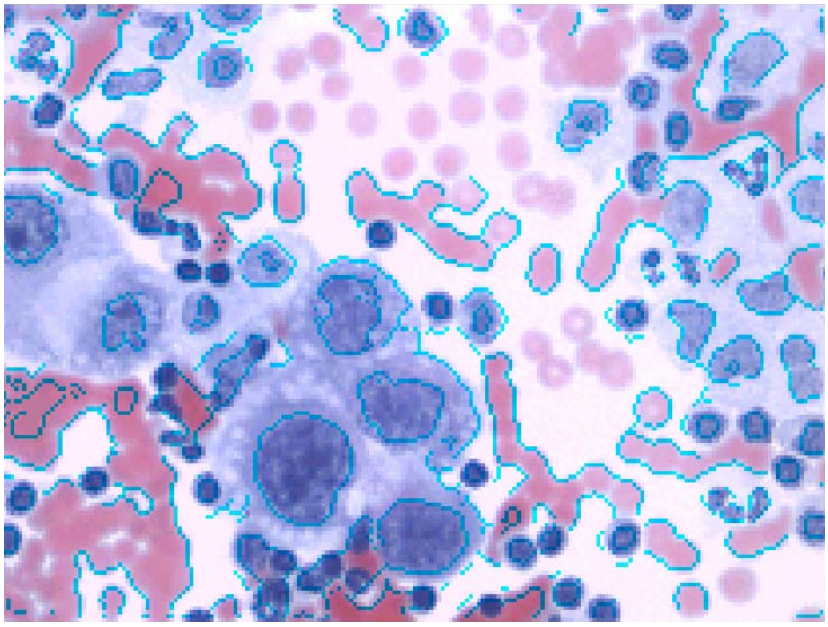


Figura 8.10. Imagen original con bordes realzados.

Una vez seleccionados los núcleos, se procede a su extracción utilizando previamente la técnica de los contornos adaptativos para aislarlos. El resultado final de esta técnica se muestra en la figura 8.11.

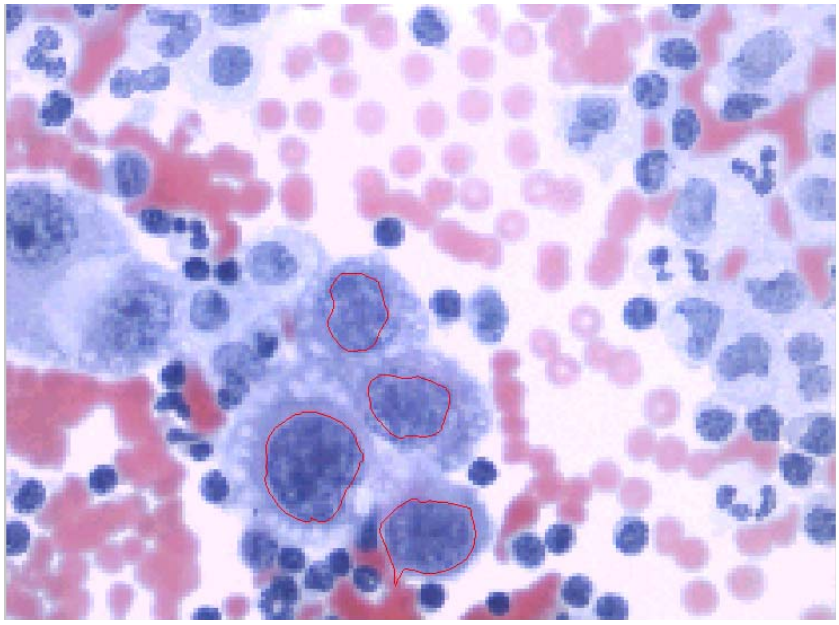


Figura 8.11. Núcleos aislados de células seleccionadas.

8.2.3 El problema de la extracción de características.

Tras el pre-procesamiento de la imagen y la segmentación de la misma, se puede realizar la medida cuantitativa de características. El número de características básicas que se pueden usar en la práctica es bastante grande, pero muchas de ellas están correlacionadas (por ejemplo, el número de gránulos o islas de intensidad y el tamaño de la célula).

En el caso de histometría y citometría podemos dividir las características en cuatro grupos: características de intensidad, características de forma, textura de la subregión y textura de la muestra. Podemos encontrar una descripción bastante completa de las características usuales en [Rodenacker, 2001].

Las características de intensidad dan una idea de la acción del tinte sobre el objeto en la muestra. Los valores de gris de las regiones segmentadas se transforman en densidades ópticas mediante un proceso de calibración y posteriormente se calculan parámetros como valor medio, suma, desviación estándar, skew y kurtosis. Estas características son especialmente importantes en el caso del uso de marcadores inmunohistoquímicos, siendo necesaria la adecuación del algoritmo a las propiedades del marcador.

Las características de forma obtienen valores a partir del contorno de los objetos y relacionados con la forma de los mismos. Por ejemplo para un núcleo celular se puede obtener la relación de áreas entre el núcleo y el citoplasma, el perímetro, el radio del mayor círculo inscrito, parámetro de forma, etcétera. En [Hu, 1962] se da un conjunto de momentos invariantes que pueden ser usados también como descriptores de forma.

Nos detendremos más en las características de textura de una subregión, ya que en este campo vamos a centrar la mayor parte de la investigación. La medida de la distribución de cromatina en el núcleo celular tiene una gran importancia en el diagnóstico. Por este motivo en este trabajo se dirigirá buena parte del esfuerzo investigador en el estudio de parámetros de este tipo para la detección de cambios en la distribución de cromatina en el núcleo celular.

La relación entre la alteración de la distribución de cromatina y la aparición de una patología es objeto de estudio desde hace años. Por citar sólo algunos ejemplos, en [Singh et al., 2000] se presentan evidencias indicando que la regulación incorrecta de la estructura de cromatina inhibe las rutas normales de diferenciación celular y estimula la proliferación incontrolada de células. En [Weyn et al., 1999] las características relacionadas con la textura de la cromatina fueron los mejores indicadores para el diagnóstico del mesothelioma maligno. En [Burger. et al., 1986] se analiza la relación entre la distribución de cromatina y el diagnóstico de citologías de cervix.

El análisis de la distribución de cromatina es muy importante para la detección de los denominados MAC (*Malignant Associated Changes*) de mucho interés para del diagnóstico de patologías en sus fases más tempranas [Hallinan, 1999] y en una variedad de pruebas y tejidos: citologías de la cervix [Bibbo et al., 1981], tejido del colon [Bibbo et al, 1990], tiroides [Lerma-Puertas et al., 1989], mama [Palcic et al., 1993], [Susnik et al., 1995], cáncer de pulmón [Palcic et al., 1998], laringe [Dreyer et al., 1999]. En muchos de estos trabajos se constata que la característica más importante para señalar la existencia de MAC es la textura nuclear.

Siguiendo el estudio de [Rodenacker, 2001], la textura se ha descrito con parámetros basados en dos aproximaciones al problema. Por una parte tenemos algoritmos que tratan de describir la textura mediante medidas heurísticas, imitando la percepción del experto. La segunda alternativa es la utilización de estadística de segundo orden para su modelado.

Comencemos por el segundo bloque, el de los operadores clásicos. El operador gradiente aplicado a la imagen permite obtener una aproximación al campo vectorial del gradiente, interpretable como el campo de velocidades de cambio en la imagen. Por otra parte el operador laplaciano puede ser interpretado como una medida de la velocidad de cambio del gradiente y da lugar a un campo escalar (un valor para cada píxel). El laplaciano se suele obtener por convolución con un kernel (matriz) de tamaño r. De las características más importantes obtenidas tras la aplicación del operador laplaciano es la desviación estándar que da cuenta de la intensidad de las partículas en la imagen de tamaño coincidente con el tamaño del kernel r [Smith, 1989].

Otro filtro que se suele aplicar para obtener características texturales es el filtro de la mediana. Se trata de un filtro no lineal de suavizado. Suaviza aquellas partículas con un tamaño hasta la mitad de la ventana utilizada en el proceso de suavizado. La transformación que se emplea es la diferencia entre la imagen original y la imagen procesada con el filtro de la mediana, conservando así las partículas de interés [Rodenacker et al., 1981]. A la diferencia obtenida se le denomina imagen de textura plana. Las características de tipo *run-length* y co-ocurrencias [Haralick et al., 1973] son aplicadas frecuentemente tanto sobre la imagen de extinción, como sobre la imagen de textura plana [Yogesan y Schulerud, 1998], [Schulerud, 1997], [Weyn, et al., 1999]. Como vemos este tipo de operadores obtiene información sobre la variabilidad de las intensidades de gris dentro de los objetos analizados a partir de medidas locales.

Existen otras técnicas que persiguen simular la capacidad del experto para percibir diferentes texturas. Se trata normalmente de operadores que permiten obtener información acerca de la variabilidad de la densidad óptica de la muestra a partir de medida globales, como el operador de la transformada *watershed*, descrita anteriormente. Esta transformación tiene especial valor ya que puede establecer el número de partículas en la cromatina así como la zona de influencia de cada partícula [Rodenacker, 2001]. Los resultados obtenidos a partir de la aplicación de la transformación *watershed* pueden ser usados para aislar las partículas observables en el núcleo celular y aplicar un tipo de descriptores en donde se trata de plasmar la distribución de las partículas mediante el denominado análisis de la estructura sintáctica, que también se aplica a nivel de tejidos para describir la distribución de células.

La imagen que la transformada *watershed* ofrece de la cromatina es una imagen granular plana en el sentido de que la superficie del núcleo es subsegmentada en compartimentos aislados. Sin embargo, en esta investigación se explora otro tipo de enfoque donde la superficie del núcleo se parece más a una representación topográfica del terreno descrita mediante curvas de nivel, tal y como se expone a continuación.

Una vez que se han aislado los núcleos a clasificar de la imagen original, se pasan a escala de grises en imágenes independientes. El efecto de la tintura de la muestra se suaviza pasando un filtro pasa-baja a cada imagen. En el siguiente paso, se lleva a cabo la medida de la textura de cada núcleo diseñada en este trabajo. Esta medida constituirá la traza que representa a ese núcleo y será la futura entrada del sistema clasificador.

Para obtener dicha traza, se realiza un mapa topográfico o mapa de contornos del núcleo. Con este mapa es posible encontrar una característica importante de la textura del núcleo consistente en calcular una medida de complejidad que refleja cómo están distribuidos los contornos en el mapa. Los mapas con los que trabajamos tienen los contornos distribuidos en N escalas o niveles distintos. En la figura 8.12 se muestran los resultados de todo este procedimiento a un núcleo benigno y a un núcleo maligno.

Para la realización de la medida de complejidad se han desarrollado dos estrategias distintas:

- La primera medida diseñada recoge globalmente la complejidad de la textura del núcleo mediante la construcción de un árbol homotópico con los niveles del mapa de contornos del núcleo. En esta aproximación, una zona del núcleo especialmente marcada puede contener otras, estableciéndose así una estructura jerárquica en forma de árbol, donde la raíz del mismo es el núcleo celular completo y las hojas pasan a ser las partículas más finas obtenidas a partir del establecimiento de estas curvas de nivel. Se trata también de una descripción basada en características globales más que locales al igual que la transformada *watershed*. El seguimiento de la estructura de árbol, permite cuantificar el cambio de una magnitud inicialmente subjetiva como es la complejidad de la estructura de cromatina en diferentes escalas espaciales. Esta aproximación se puede encontrar en [Estévez et al., 2002a].

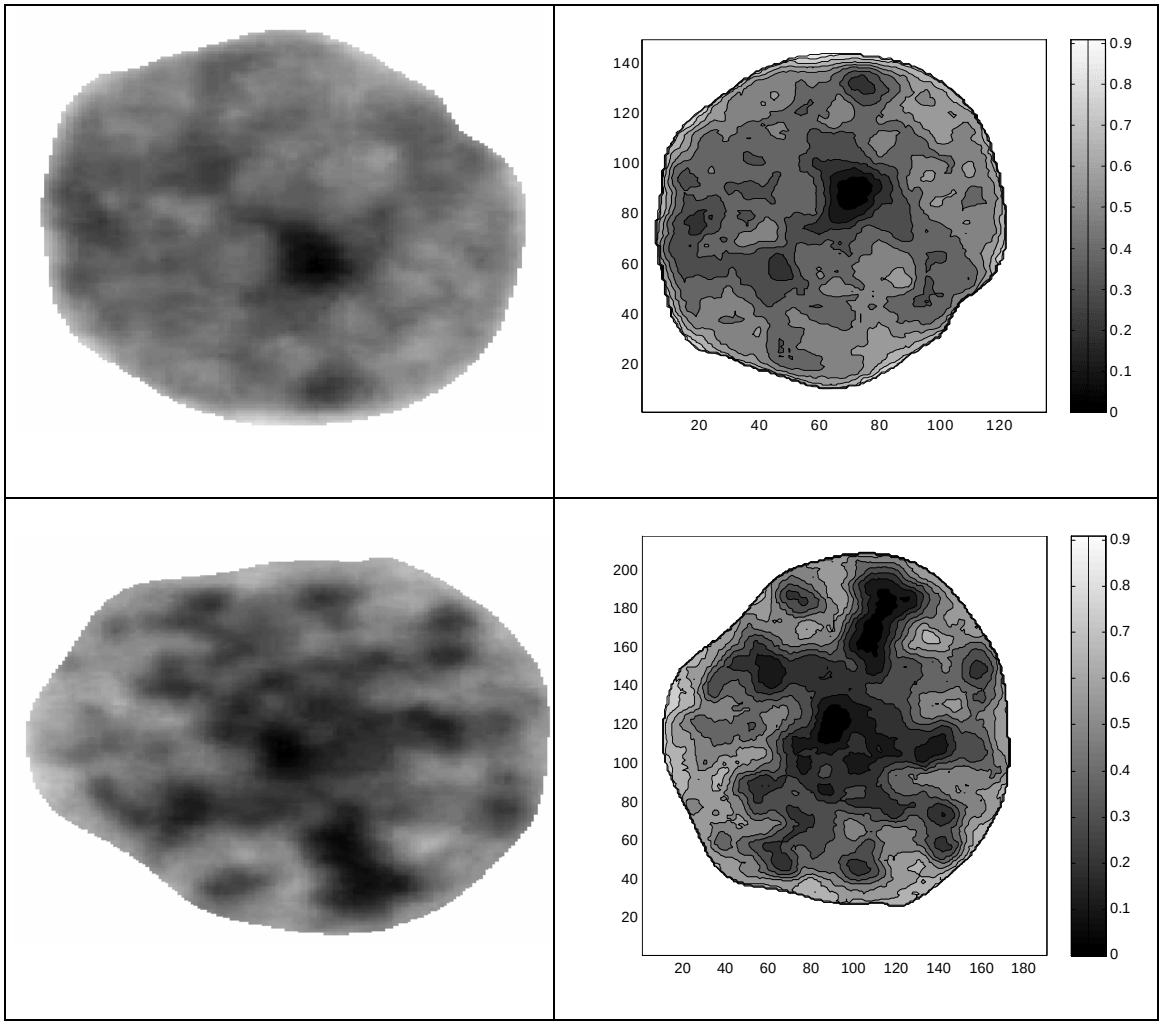


Figura 8.12. Núcleo aislado y mapa del núcleo para un núcleo benigno (primera columna) y otro maligno (segunda columna).

Para construir este árbol de complejidad se recorren los contornos del mapa, comenzando por los contornos o niveles más externos y avanzando hacia los más internos. El árbol derivado de un mapa de contornos es un estructura de árbol que se construye mediante la relación binaria "el contorno A es el nivel que soporta al contorno B", es decir, el contorno B está contenido o incluido dentro del contorno A y no existe ningún otro contorno C que incluya a B. De este modo, el contorno más externo del núcleo es la raíz del árbol, y los contornos más pequeños son las hojas del árbol.

El árbol que se obtiene se convierte en una serie de datos: una secuencia ordenada de valores, donde cada elemento representa la información relativa a un nivel de soporte. La medida aplicada es sencilla: cada valor es el número de

ramas en el correspondiente nodo del árbol. La traza final está constituida por estos valores normalizados.

En la figura 8.13 se presenta un ejemplo de este procedimiento de construcción del árbol. En la parte superior este ejemplo, se puede apreciar que, según el convenio establecido, estudiando los contornos A, B, C, D y E, A contiene a todos los demás, pero A no es el contorno que los soporta a todos. El contorno A soporta a C y a D, y éstos, a su vez soportan a otros (C soporta a B y D soporta a E).

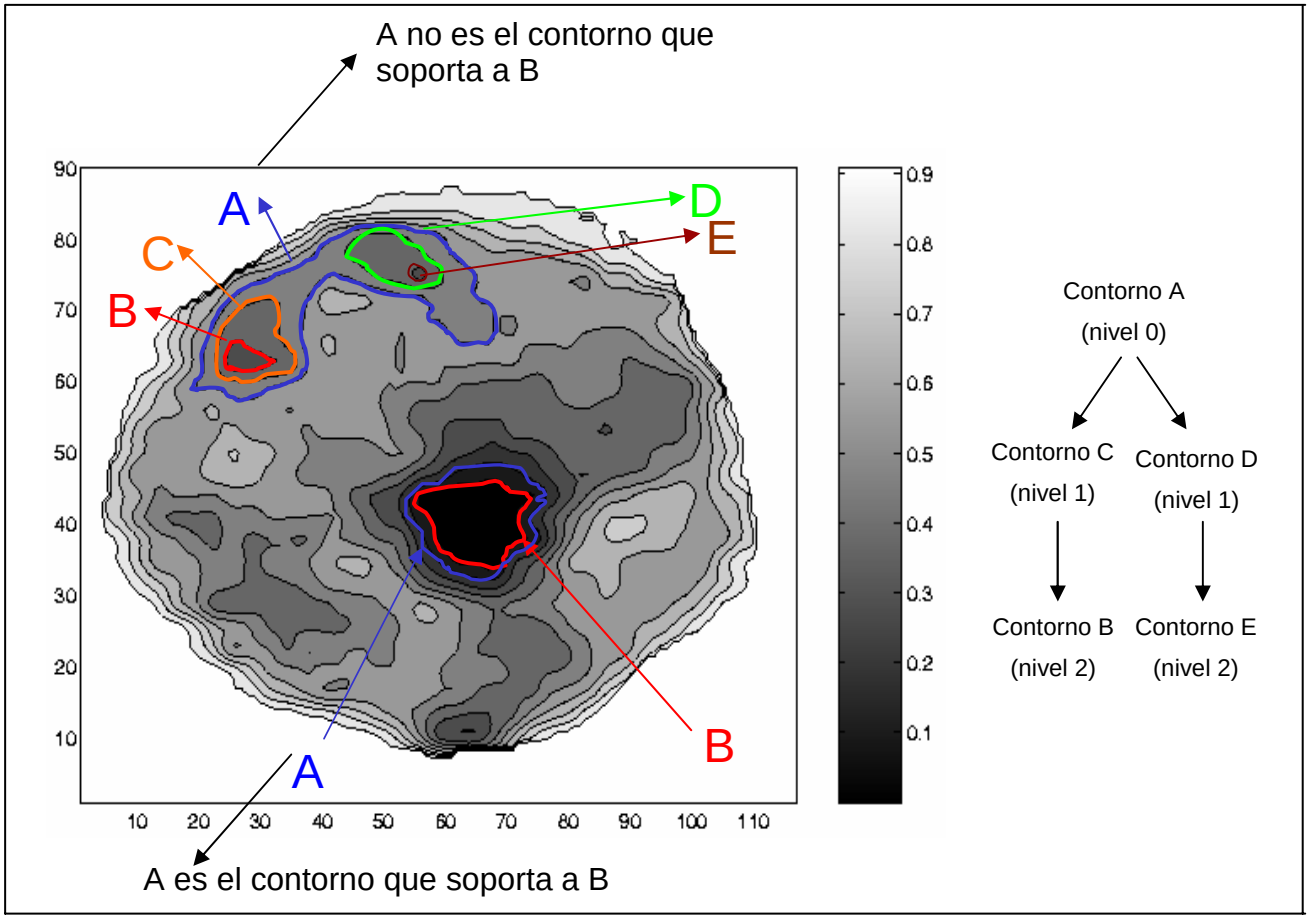


Figura 8.13. Ejemplo de construcción de árbol homotópico.

- En la segunda aproximación de esta medida de complejidad se simplifica el proceso notablemente. La medida de complejidad se obtiene simplemente contando los contornos que hay en cada nivel y normalizando todos esos valores. Este vector normalizado constituye, en el caso en que los mapas se construyan con *N* niveles, una traza de *N* valores que representa la textura del

núcleo y que será la entrada externa al sistema FFSM. Esta aproximación se puede encontrar en [Estévez et al., 2002b]. Con este método se produce una traza normalizada que recorre diferentes escalas espaciales. Esta traza puede considerarse una representación intermedia con un grado de capacidad de diferenciación entre núcleos normales y anormales.

La razón de desarrollar estos descriptores es que hemos observado que las texturas de células benignas tienen áreas grises más homogéneas y contornos más concéntricos que las texturas de células malignas. Por lo tanto, la complejidad de la estructura de los árboles está directamente relacionada con la complejidad de la distribución de estas áreas en el núcleo. En los primeros experimentos realizados se utilizó la primera aproximación basada en los árboles homotópicos. Como los resultados fueron aceptables, se llegó a la conclusión de que esta medida podía contener información suficiente sobre la naturaleza de los núcleos. También se desarrolló una versión más simplificada de esta medida, que es la que se ha expuesto en segundo lugar. En las pruebas desarrolladas con esta segunda medida se pudo comprobar su validez, y que, en algunos casos, incluso daba lugar a mejores resultados que la primera.

A simple vista parece que los valores absolutos de cualquiera de estas dos medidas de complejidad podrían ser buenos parámetros para separar las células benignas y malignas. Sin embargo, después de aplicar los sistemas borrosos de clasificación supervisados sobre las trazas absolutas, se obtuvieron resultados deficientes en la clasificación de los conjuntos de test (baja capacidad de generalización). La razón de esto podría ser la existencia de una dependencia entre esta característica y otras características de la célula, independientemente de su naturaleza benigna o maligna.

Descartada esta posibilidad, se investigó la medida de distribución de la complejidad a través de las diferentes escalas. La idea consiste en estudiar la estructura de la traza obtenida como un todo, donde la complejidad varía de un nivel a otro, por lo tanto, la traza obtenida se normaliza, dividiendo todos sus valores por el máximo valor de la traza. La utilización de trazas normalizadas mejora sensiblemente la capacidad de generalización, por lo que concluimos que existía una fuerte relación entre la clase del núcleo analizado y la forma **relativa** en la que se distribuye el conjunto de contornos detectados entre los diferentes niveles y escalas.

Es interesante destacar que las técnicas descritas también se pueden aplicar al análisis de la estructura sintáctica del tejido o texturas de las muestras. En este caso los centroides calculados sobre los núcleos que forman la muestra de tejido son utilizados para obtener diferentes descriptores a partir de diagramas de Voronoi, el grafo de Gabriel o el *minimum spanning tree* (MST) (ver [Weyn et al., 1999]).

El sistema implementado en este trabajo permite extraer características sobre la distribución de la cromatina en el núcleo a diferentes escalas, descriptor que ha permitido obtener buenos resultados en los estudios de clasificación realizados. Es importante destacar que los resultados obtenidos de la mejora en los métodos de entrenamiento de los sistemas recurrentes, incidirá no sólo en el dominio de aplicación sino también en otros muchos donde el reconocimiento de series temporales o series de datos es una factor clave. La aplicación de nuevos clasificadores y nuevas características de la textura de carácter global, como el mapa de contornos, sobre la estructura de la cromatina nuclear es un campo de investigación que adquiere cada vez mayor importancia al aparecer publicaciones donde se pone de manifiesto su relevancia en el diagnóstico de patologías [Singh et al., 2000], [Einstein et al., 1998], [Estévez et al., 2002a], [Estévez et al., 2002b].

8.2.4 El problema del diseño de clasificadores

Una vez que se dispone del conjunto de características elegidas de acuerdo con los criterios anteriormente expuestos, se llega a la fase de clasificación. En esta fase la principal dificultad radica en el diseño del clasificador más adecuado para el problema. Abordaremos el problema del diseño de un clasificador desde la perspectiva de un problema de diseño general como una búsqueda en un espacio amplio de posibles diseños que deben satisfacer ciertos requisitos y no violar determinadas restricciones. En el caso particular de los clasificadores, el número de posibles diseños es enorme sin más que tener en cuenta los diferentes tipos que existen (estadísticos, redes neuronales, árboles de decisión, ... [Ripley, 1996], [Duda et al., 2001]) y todas las posibilidades que resultan de variar los parámetros que los definen. Se trata de una tarea compleja que suele resolverse siguiendo un procedimiento de prueba y error de diferentes tipos de clasificadores. Por otro lado, a pesar de que en teoría cuanto mayor es el número de características, mejor debería ser la clasificación (al disponer de más información), lo

cierto es que un número muy elevado de características puede tener consecuencias prácticas no deseables. Esto se debe al hecho de disponer de un conjunto de datos de entrenamiento finito y generalmente escaso que complica notablemente el proceso de diseño del clasificador mermando sus prestaciones finales [Fukunaga, 1990].

En este trabajo, los clasificadores elegidos son sistemas borrosos recurrentes. La búsqueda de los clasificadores más óptimos se lleva a cabo mediante las estrategias evolutivas descritas en los anteriores capítulos: sistemas tipo Pittsburgh y sistemas tipo Michigan. El objetivo de estos sistemas en esta ocasión es encontrar la máquina de estados borrosa capaz de reconocer dos tipos de texturas diferenciadas y clasificarlas correctamente. Para tal fin, y siguiendo el procedimiento anteriormente descrito, se forma un conjunto de entrenamiento y otro de test, con trazas de células que previamente han sido catalogadas como “benignas” o “malignas” por un experto del dominio.

En las pruebas realizadas se usan conjuntos de entrenamiento de tamaño moderado, para que los experimentos se puedan realizar sin grandes exigencias computacionales. Esta condición inicial sirve para depurar la estructura y métodos de entrenamiento de los sistemas recurrentes así como la propia adecuación de los descriptores. Además, para el estudio de la textura nuclear, se diseñan conjuntos de entrenamiento representativos de las clases, procurando incluir la variedad de posibilidades que ocurren de forma típica, fijando las condiciones experimentales.

Por lo tanto, dos de los objetivos de este trabajo son: por una parte la investigación sobre la eficacia de la aplicación de los sistemas recurrentes obtenidos mediante aprendizaje inductivo en la discriminación de texturas, como es el caso de la distribución de cromatina a partir de su descripción como mapa de contornos, y por otra, la relación de la distribución de la complejidad en la estructura de cromatina a lo largo de diferentes escalas espaciales y el carácter anómalo de la célula. La detección de cambios en texturas mediante sistemas recurrentes diseñados mediante aprendizaje inductivo es una aplicación novedosa.

8.2.5 El problema de la validación.

En esta fase se trabaja conjuntamente con los expertos del dominio. Se intenta validar tanto los resultados en el análisis y clasificación de las imágenes, como la sistemática de

la definición de los protocolos y experimentos, técnicas de visualización de resultados y métodos de representación de las características extraídas. En este trabajo nos hemos limitado a la validación del sistema de clasificación que se ha planteado. Para validar los clasificadores obtenidos se utiliza un conjunto de test, compuesto por trazas correspondientes al dominio en estudio y diferentes a las utilizadas en los conjuntos de entrenamiento. Además, se realizan validaciones cruzadas.

En las siguientes secciones se presenta un estudio comparativo a nivel de complejidad computacional, convergencia, bondad de la clasificación y características de los sistemas borrosos resultantes. Además, se expone otra comparativa de los resultados de clasificación obtenidos entre estos sistemas y otros métodos de clasificación convencionales. Dada la naturaleza del problema, se evalúan los clasificadores bajo estudio desde el punto de vista médico, mediante el análisis de curvas ROC.

8.3 Resultados.

En esta sección se presentan los resultados de los experimentos realizados sobre la aplicación de los sistemas tipo Pittsburgh y tipo Michigan en la búsqueda de sistemas borrosos recurrentes para la clasificación de núcleos en imágenes de citologías médicas.

En primer lugar, se comentan los resultados de las pruebas preliminares que se hicieron con imágenes digitalizadas correspondientes a la prueba médica de aspiración por aguja fina de tejido de mama. Con estas imágenes se utilizaron sistemas tipo Pittsburgh. En segundo lugar, se presentan los resultados obtenidos en la aplicación de los sistemas tipo Pittsburgh en imágenes de citologías correspondientes al tejido peritoneal. En este punto se realiza un análisis más exhaustivo de estos resultados, mediante una comparativa con otros métodos de clasificación convencionales y una evaluación de los clasificadores obtenidos con curvas ROC. Por último, se muestran los resultados concernientes a la aplicación de los sistemas tipo Michigan en imágenes de citologías correspondientes a la pleura, junto con un análisis detallado análogo al anterior.

8.3.1 Clasificación de núcleos en imágenes de tejido de mama.

8.3.1.1 Descripción del problema.

A pesar de toda la investigación científica que se ha realizado sobre esta patología, el cáncer de mama continúa siendo la forma más común en la que el cáncer se manifiesta y la segunda mayor causa de muerte entre las mujeres. Las oportunidades de supervivencia a esta enfermedad se incrementan por la detección temprana de la misma, y una detección temprana depende de la exactitud del diagnóstico.

Existen tres métodos para diagnosticar el cáncer de mama: mamografía, aspiración por aguja fina con interpretación visual y biopsia. Como ya se mencionó en la introducción de este capítulo, la biopsia es el método más exacto, aunque, al tratarse de un procedimiento quirúrgico, es invasivo y costoso. Los sistemas de diagnóstico basados en el análisis de imágenes digitalizadas permiten realizar un diagnóstico exacto en muchos casos, sin necesidad de realizar una biopsia.

La prueba de aspiración por aguja fina (en inglés, *fine needle aspirate* – FNA) se realiza de la siguiente manera: primero, una muestra de fluido se toma del pecho de la paciente. Este procedimiento implica el uso de una pequeña aguja para tomar el fluido directamente de un bulto o masa que se encuentra en el pecho, que previamente ha sido detectado por examinación propia y/o mamografía. El fluido se coloca en un porta-objetos y se le aplica sustancias especiales para resaltar los núcleos de las células. Las imágenes de estas preparaciones se transfieren a una estación de trabajo mediante una cámara montada sobre un microscopio [Mangasarian et al., 1995].

Con imágenes de estas características se realizaron los primeros experimentos. Las imágenes utilizadas en estos experimentos preliminares fueron tomadas de la base de datos de cáncer de mama de la Universidad de Wisconsin [Wolberg, 1992], una base de imágenes diseñada para la validación de algoritmos de clasificación, publicada por el Dr. William H. Wolberg de la Universidad de Wisconsin. En esta base de imágenes, los casos malignos fueron confirmados por biopsia, mientras que los benignos fueron confirmados tanto por biopsia como por posteriores examinaciones médicas periódicas.

La base se compone de 569 imágenes de aspiraciones por aguja fina que contienen células epiteliales (212 con cáncer y 357 con enfermedad fibroquística). El área en las muestras que fue digitalizada se seleccionó procurando minimizar el solapamiento de los núcleos.

Las imágenes para el análisis digital se generaron con una cámara de color JVC TK-1070U montada sobre un microscopio Olympus. Las preparaciones fueron proyectadas en una cámara con un objetivo de 63× y un ocular de 2.5×. Las imágenes se capturaron con una tarjeta de adquisición de imágenes de color ComputerEyes/RT (Digital Vision, Inc., Dedham MA 02026) como ficheros Targa 512×480. Las imágenes resultantes se almacenaron en memoria como matrices de dos dimensiones, donde cada píxel tiene un valor entre 0 y 255, representando la intensidad de la luz en ese punto [Wolberg et al., 1993]. En la figura 8.14 se muestran 2 imágenes procedentes de esta base de imágenes, una que contiene núcleos catalogados como benignos y otra con núcleos catalogados como malignos.

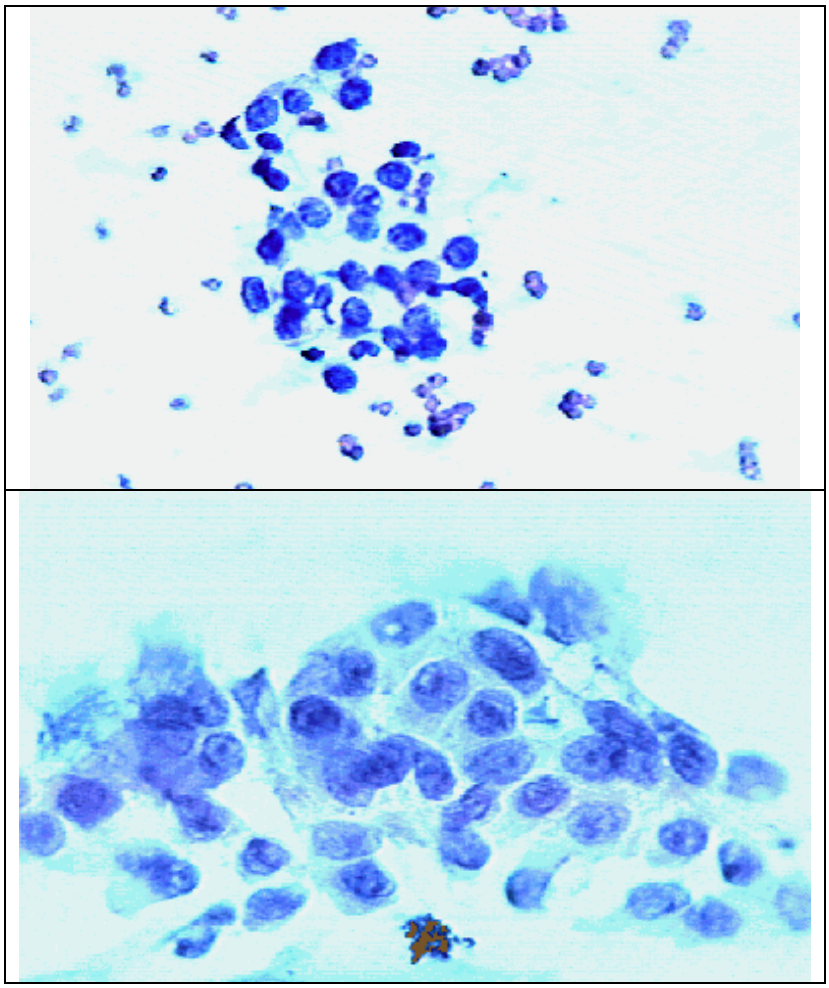


Figura 8.14. Imágenes procedentes de la base de datos de cáncer de mama de la Universidad de Wisconsin. Imagen superior: núcleos catalogados como benignos. Imagen inferior: núcleos catalogados como malignos.

Sobre estas imágenes se aplica el clasificador compuesto por el sistema borroso recurrente diseñado en este trabajo. Para ello, seleccionamos algunos núcleos benignos

y malignos de estas imágenes con el objetivo de construir un conjunto de entrenamiento para nuestro sistema de aprendizaje supervisado tipo Pittsburgh. El clasificador obtenido se validará posteriormente con un conjunto de test, compuesto por núcleos de estas imágenes distintos a los pertenecientes al conjunto de entrenamiento.

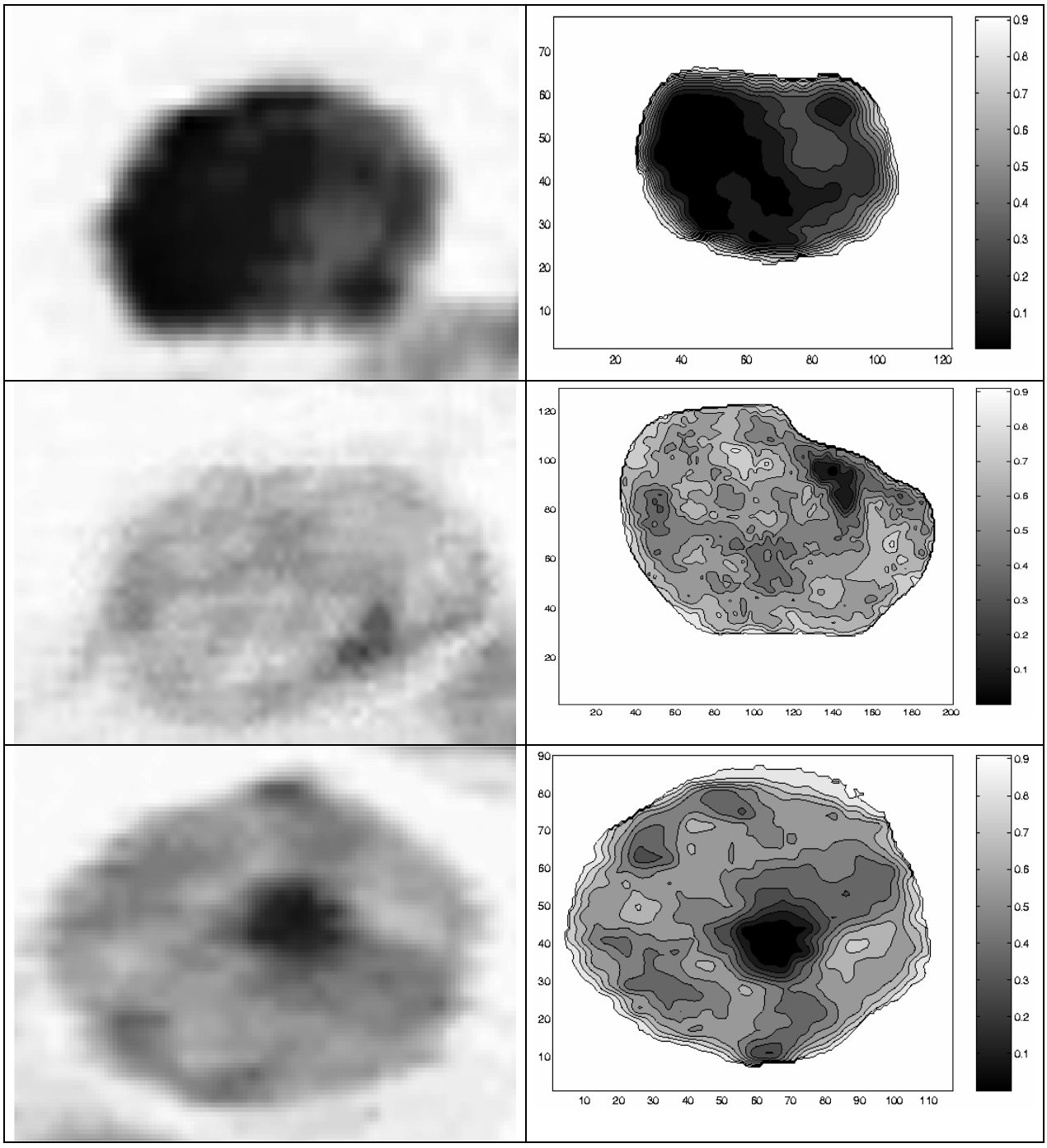


Figura 8.15. Ejemplo de núcleos bajo estudio y sus correspondientes mapas de contornos: núcleo benigno (primera fila), núcleo maligno (segunda fila) y núcleo de naturaleza no tan clara (tercera fila).

Tras el proceso de segmentación, se procede a la extracción de características. Como ya se ha comentado, la característica con la que se trabaja es la textura del núcleo,

que es extraída mediante el procedimiento ya descrito basado en el análisis de los mapas topográficos o de contornos de los núcleos. Para construir la traza de cada núcleo en estos experimentos se utilizó la primera aproximación, en la que se construía un árbol homotópico (árbol jerárquico). En la figura 8.15 se muestran los resultados de este procedimiento: núcleo benigno, núcleo maligno y núcleo de naturaleza no tan clara, segmentados, con sus correspondientes mapas de contornos.

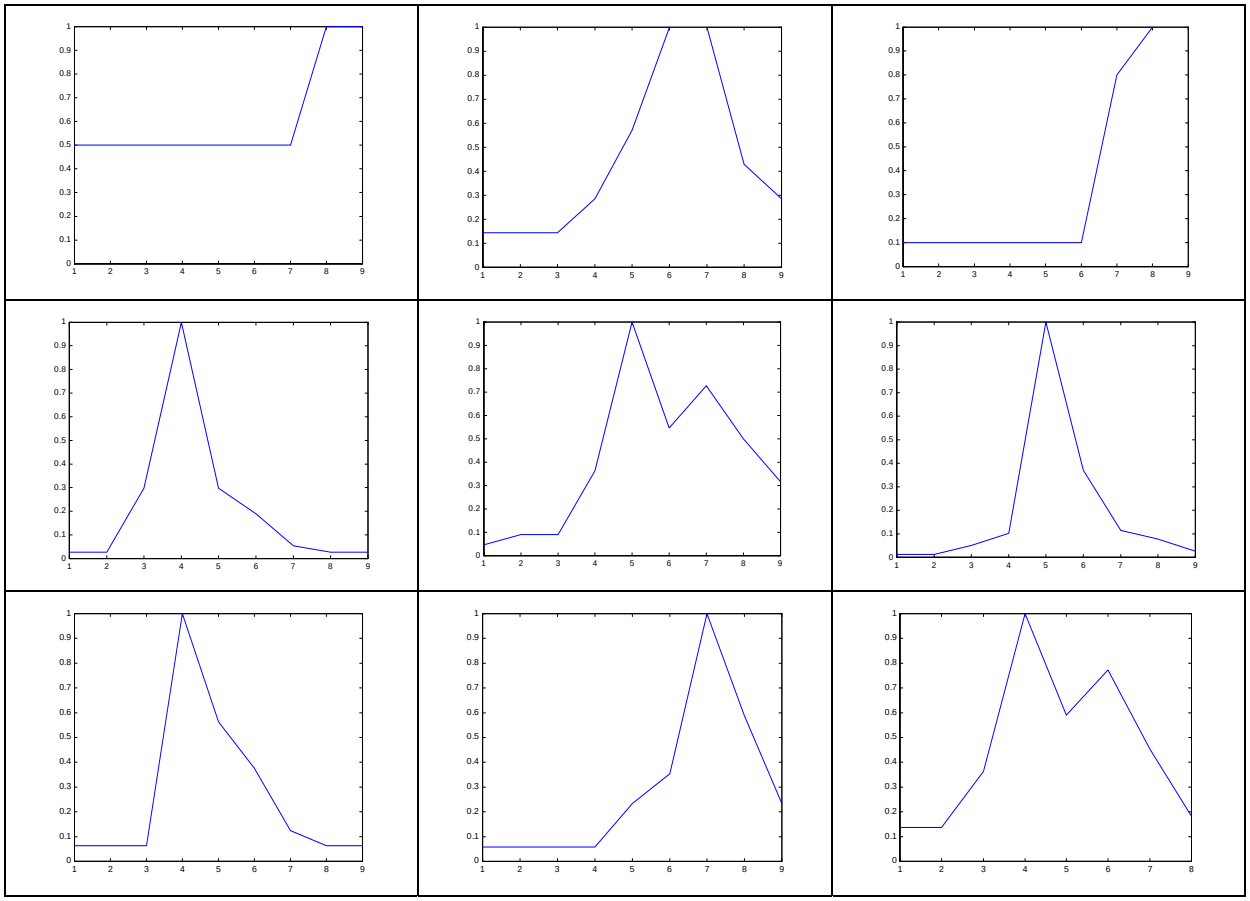


Figura 8.16. Trazas normalizadas para células benignas (primera fila), células malignas (segunda fila) y células de naturaleza no tan clara (tercera fila). Eje X: índice del elemento en la traza. Eje Y: valor normalizado de la traza.

Las trazas que se obtienen a partir de estos mapas de contornos son las entradas del sistema clasificador. Cada traza es una serie de datos que refleja cómo cambia la distribución de cromatina en el núcleo en diferentes escalas espaciales. En la figura 8.16 se muestran varias trazas correspondientes a núcleos benignos, núcleos malignos y núcleos de apariencia no tan clara. Observando esta figura se aprecia que el problema de clasificación es complicado, ya que no parece que exista un patrón claramente definido para cada clase a simple vista.

8.3.1.2 Experimentos realizados y resultados.

Una vez obtenidas las trazas que componen el conjunto de entrenamiento, se procede a aplicar un sistema de búsqueda evolutiva tipo Pittsburgh para encontrar el sistema borroso recurrente capaz de clasificar los núcleos en dos clases. Recordemos que el sistema borroso buscado es una máquina finita de estados borrosa (FFSM).

Los algoritmos genéticos son optimizadores globales que tienen la capacidad de llevar a cabo una búsqueda en el espacio de entrada. Son muy flexibles porque la función de aptitud puede incluir líneas de diseño y restricciones y la naturaleza del proceso de búsqueda evita mínimos locales. Por esta razón han sido usados en muchas ocasiones para encontrar configuraciones adecuadas y parámetros en sistemas borrosos [Cordón et al., 2001]. En nuestro caso, el algoritmo genético se usa para realizar una búsqueda en un espacio compuesto por modelos FFSM. Los detalles del algoritmo de búsqueda de los sistemas tipo Pittsburgh se ha descrito detalladamente en el capítulo 6.

Es importante destacar que el experimento que se presenta a continuación fue una prueba preliminar en esta investigación. Como en el momento de su realización aún no contábamos con el asesoramiento directo de un experto del dominio, se trabajó con cada preparación catalogada como “benigna” o “maligna” considerando que todos los núcleos pertenecientes a ellas se correspondían también a estas categorías, ya que no disponíamos de la información individual de cada célula. Es evidente que las células de las preparaciones catalogadas como “benignas” son todas benignas, pero tuvimos que asumir que todas las células de las preparaciones “malignas” tenían esa naturaleza. Esta última suposición conlleva error, ya que pueden existir células benignas localizadas en preparaciones donde existen células malignas.

El clasificador basado en un FFSM se entrena en el proceso de aprendizaje con el conjunto de entrenamiento. Para medir la calidad de este aprendizaje, es necesario comprobar la habilidad del clasificador FFSM para clasificar las trazas pertenecientes al conjunto de test. El clasificador se aplica a núcleos individuales, célula a célula, no imagen a imagen.

Resultados del entrenamiento.

La salida de la etapa de entrenamiento es la FFSM junto con los parámetros del algoritmo de clustering (vector $\vec{v}$ y $\vec{a}$).

El conjunto de entrenamiento se compuso de 17 núcleos catalogados como benignos y 18 núcleos catalogados como malignos. El proceso de aprendizaje se desarrolló bajo las condiciones descritas en la tabla 8.1.

Parámetro	Valor
Tamaño de la población (*num_maquinas*)	200
Umbral de parada del algoritmo (*umbral_fitness*)	0.1
Nivel que debe alcanzar el estado de detección para considerar que está activado a alta (*param_alta*)	0.7
Parámetro de selección (*alfa*)	0.5
Probabilidad para la reproducción (*p1*)	5 %
Probabilidad para la mutación (*p2*)	75 %
Probabilidad para el cruce (*p3*)	20 %
Número de reglas a mutar (*mutar_reglas*)	5
Número de elementos a mutar por regla (*mutar_elementos*)	5

Tabla 8.1. Parámetros del algoritmo de aprendizaje del sistema Pittsburgh para el problema de la clasificación de núcleos en imágenes de tejido de mama.

Los individuos de la población son máquinas finitas de estados borrosas cuyos antecedentes y consecuentes vienen definidos por funciones de pertenencia gaussianas. El modo en que se codifican estas máquinas ha sido descrito en el capítulo 6. Las características de estos individuos se recogen en la tabla 8.2.

Número de reglas (*num_reglas*)	6
Número de estados (*num_estados*)	4
Número del estado de detección (*num_detec*)	4
Centro de las funciones de pertenencia (variable)	[0.1, 0.3, 0.5, 0.7, 0.9]
Desviación de las funciones de pertenencia (fijo)	0.2

Tabla 8.2. Características de las máquinas finitas de estados borrosas que componen la población en el sistema tipo Pittsburgh para el problema de la clasificación de núcleos en imágenes de tejido de mama.

En la figura 8.17 se presenta la curva de entrenamiento del proceso de aprendizaje Pittsburgh desarrollado. Como se puede observar en la gráfica, en aproximadamente 23 iteraciones, el sistema alcanza el umbral de eficiencia deseado (el umbral de parada es 0.1).

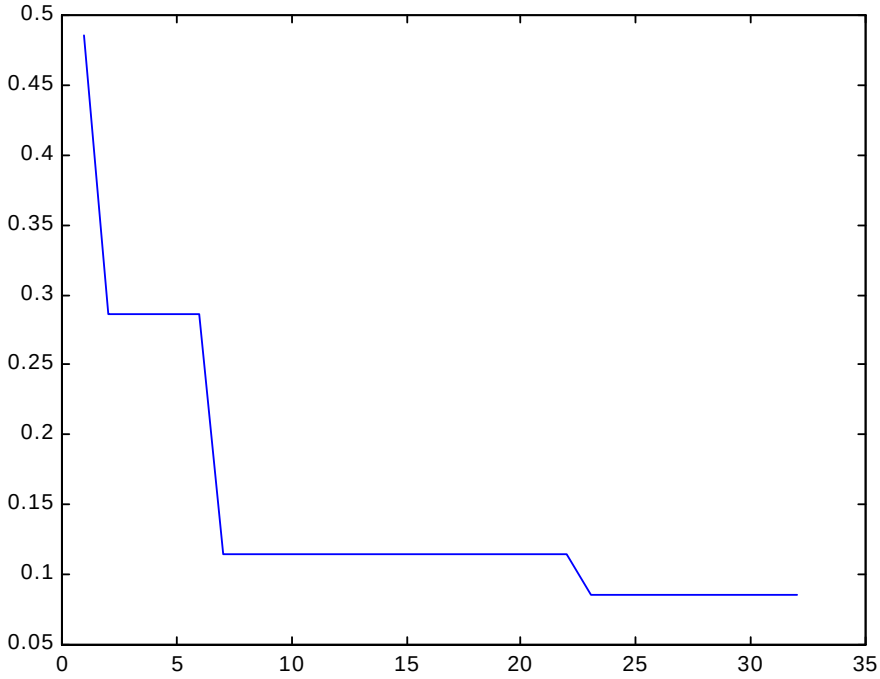


Figura 8.17. Curva de entrenamiento que refleja la evolución de la aptitud. Eje X: número de iteraciones. Eje Y: valor de aptitud.

El clasificador final está constituido por la FFSM ganadora del proceso de aprendizaje junto con los centros del algoritmo de clustering. En este caso, estos centros son [0.0002, 0.1036]. En la tabla 8.3 se presentan los resultados de la clasificación sobre el conjunto de entrenamiento.

Resultados del test.

Para estudiar la calidad del aprendizaje y evaluar la eficiencia de la FFSM obtenida como clasificador, se realiza un procedimiento de validación simple, consistente en evaluar la clasificación que realiza el sistema obtenido sobre las trazas que componen el conjunto de test. Posteriormente, se realizará una validación cruzada.

La clasificación se lleva a cabo analizando la reactividad del estado de detección de la FFSM ante cada traza del conjunto de test, del mismo modo en que la aptitud es calculada (ver capítulo 6), y asignando la respuesta a alguno de los dos clusters.

El conjunto de test se compuso de 31 núcleos catalogados como benignos y 41 núcleos catalogados como malignos. Los resultados de la clasificación del conjunto de test se muestran en la tabla 8.3.

	Conjunto de entrenamiento (17 trazas beignas y 18 trazas malignas)	Conjunto de test (31 trazas beingnas y 41 trazas malignas)	Conjunto total (48 trazas benignas y 59 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	15 trazas (88.23% aciertos)	31 trazas (100% aciertos)	46 trazas (95.83% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	2 trazas (11.77% error)	0 trazas (0% error)	2 trazas (4.17% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	17 trazas (94.44% aciertos)	27 trazas (65.85% aciertos)	44 trazas (74.57% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	1 traza (5.56% error)	14 trazas (34.15% error)	15 trazas (25.43% error)
Error global	8.57%	19.44%	15.88%

Tabla 8.3. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) para el problema de la clasificación de núcleos en tejido de mama. En la tercera columna se presenta un resumen de resultados.

Estos resultados reflejan la bondad de los árboles de complejidad como indicadores de la naturaleza de los núcleos. Se debe destacar que en el conjunto de test todos los núcleos benignos fueron clasificados correctamente y que los errores ocurrieron en la clasificación de núcleos malignos. Los resultados son coherentes con el hecho de que todas las células en las preparaciones benignas son benignas, mientras que en las preparaciones diagnosticadas como malignas pueden existir células no cancerígenas. Al estar considerando todas las células de estas últimas preparaciones como malignas estamos cometiendo un error en la información que le suministramos al algoritmo de aprendizaje supervisado.

A modo de resumen, se ha encontrado una FFSM que opera con un 8.6% error en el conjunto de entrenamiento y con un 19.4% error en el conjunto de test. Es importante destacar que la medida de textura de los núcleos utilizada es novedosa, sencilla y rápida. Con esta medida se observa que se consiguen resultados de clasificación aceptables, por lo tanto, el modo en que la cromatina del núcleo se distribuye entre las distintas escalas de los mapas de contornos utilizados proporciona bastante información de la naturaleza benigna o maligna de los núcleos.

En la validación simple que se ha presentado se agrupan aleatoriamente las trazas en dos conjuntos distintos: uno se usa como conjunto de entrenamiento para ajustar los parámetros del modelo en el clasificador y el otro (conjunto de test o de validación) se utiliza para estimar el error de generalización.

Para realizar una validación más exhaustiva de nuestro clasificador FFSM, se analizan los resultados de una validación cruzada (en inglés, *cross-validation*). Construimos aleatoriamente tres grupos de trazas diferentes (A, B y C) con las trazas disponibles de estas imágenes. Cada grupo contiene 16 trazas de células benignas y 16 trazas de células malignas. Combinando estos grupos para formar conjuntos de entrenamiento y test diferentes, se ejecutan distintas búsquedas con sistemas Pittsburgh.

En la tabla 8.4 se representan los resultados de la clasificación global (sobre todas las trazas benignas y malignas de las que se dispone) de las pruebas realizadas con los distintos conjuntos de entrenamiento y test. Además, se añade una comparativa entre la media del error de las FFSM de estas pruebas y el error cometido por la FFSM que se encontró inicialmente y que se desea validar.

	Error de clasificación sobre el conjunto de entrenamiento	Error de clasificación sobre el conjunto de test
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Primera prueba**. Conjunto de entrenamiento: grupos A-B. Conjunto de test : grupo C.	9.38 %	15.63 %
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Segunda prueba**. Conjunto de entrenamiento: grupos A-C. Conjunto de test : grupo B.	6.25 %	21.88 %
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Tercera prueba**. Conjunto de entrenamiento: grupos B-C. Conjunto de test : grupo A.	7.81 %	34.38 %
Media de los resultados de las tres pruebas	7.81 %	23.96 %
Clasificación global de la FFSM inicial que se desea validar.	8.57 %	19.44 %

Tabla 8.4. Resultados de clasificación con validación cruzada para el problema de clasificación de núcleos en imágenes de tejido de mama.

Como se puede apreciar, los resultados son aceptables y la validación cruzada confirma la eficiencia de la FFSM encontrada.

8.3.1.3 Conclusiones.

En esta sección se ha presentado el estudio preliminar realizado con los sistemas Pittsburgh sobre series de datos reales. Se diseñó un clasificador basado en sistemas borrosos recurrentes para clasificar núcleos sanos y cancerígenos en imágenes de

aspiraciones por aguja fina (FNA) de tejido de mama. El sistema borroso recurrente aplicado ha sido una máquina finita de estados borrosa (FFSM).

Las imágenes utilizadas corresponden a una base de imágenes catalogadas por un experto en el dominio. Después del proceso de segmentación, construimos series de datos que contienen información sobre la distribución de cromatina en los núcleos, es decir, hacemos una medida de la textura de los mismos. Estas series de datos son las trazas de entrada al sistema clasificador. Como el sistema de aprendizaje Pittsburgh es un proceso de aprendizaje supervisado, se construyen con estas trazas catalogadas conjuntos de entrenamiento y test.

Los principales objetivos perseguidos en esta investigación preliminar son tres:

- Comprobar si la medida de textura empleada basada en los mapas de contornos puede ser válida para la clasificación de núcleos, es decir, si contiene información suficiente para caracterizarlos.

- Comprobar si una FFSM puede constituir un sistema clasificador para series de datos reales, basando su clasificación en el comportamiento del estado de detección.

- Comprobar si un sistema evolutivo tipo Pittsburgh puede encontrar una FFSM con una eficiencia de clasificación aceptable para este problema en concreto.

Como los resultados muestran, se ha encontrado una FFSM que clasifica el conjunto de entrenamiento con un error de 8.6% y el conjunto de test con un error de 19.4%. Además, es un resultado comprobado con validación cruzada. Este resultado es bueno, pero hay que tener en cuenta que existen muchos factores en el proceso seguido que podrían mejorarlo: cambiar las especificaciones de la FFSM, cambiar el modo en que se extrae la textura de los núcleos, ajustar los parámetros del algoritmo, ...

Es importante destacar que la medida de textura utilizada es novedosa, sencilla y rápida. Permite construir trazas (series de datos) de longitud aceptable para la capacidad computacional del algoritmo y los resultados indican que estas trazas contienen suficiente información para caracterizar los núcleos adecuadamente. Con esta medida, basada en árboles de complejidad, se observa que no sólo la complejidad absoluta de la

distribución de cromatina en el núcleo es relevante, ya que además, el modo en que se distribuye esta complejidad en diferentes escalas es indicativo de la naturaleza benigna o maligna del núcleo.

Los resultados obtenidos son mejores en la clasificación de núcleos sanos que en la clasificación de núcleos patológicos. Esto puede estar influenciado por la suposición hecha de que todas las células pertenecientes a preparaciones catalogadas como "malignas" son malignas. Esta suposición implica un error a la hora de construir los conjuntos de entrenamiento y test, ya que pueden existir trazas que no están bien catalogadas en ellos, debido a que pueden existir células benignas localizadas en preparaciones donde existen células malignas.

Para evitar esta limitación y realizar un estudio más cuidadoso, se empezó a trabajar con expertos del dominio. En los siguientes experimentos realizados se cuenta con esta colaboración, y por tanto, con las células catalogadas individualmente, con lo que se disminuye esta fuente de error.

Una vez comprobada la validez de este procesamiento, en los siguientes experimentos se intenta profundizar en la metodología aplicada.

8.3.2 Clasificación de núcleos en imágenes de citologías de fluidos peritoneales.

8.3.2.1 Descripción del problema.

Como ya se comentó en la introducción, la inspección visual de imágenes de citologías es uno de los métodos más comunes para diagnosticar cáncer. Es una prueba que permite un diagnóstico de exactitud aceptable sin necesidad de intervenciones quirúrgicas, como es el caso de la biopsia.

En los experimentos que se presentan a continuación, nuestro objetivo es clasificar correctamente núcleos de células sanas y núcleos de células patológicas en imágenes digitalizadas de citologías peritoneales con las FFSMs resultantes de búsquedas realizadas por sistemas Pittsburgh, igual que en los experimentos anteriores.

Normalmente, este tipo de citologías se realiza extrayendo líquido de la zona por medio de una punción, centrifugándolo y aplicándole una tinción. Esta preparación es

la que se deposita en el porta-objetos. Las imágenes estudiadas son imágenes digitalizadas de estas preparaciones.

Con este tipo de citologías, el especialista puede diagnosticar tumores primarios intraabdominales (originarios de cualquier órgano intraabdominal) o tumores secundarios (metastásicos).

Para tal fin, a partir de este momento contamos con la colaboración del Dr. D. Lucio Díaz Flores y su equipo de investigación, del Departamento de Anatomía Patológica del Hospital Universitario de Canarias, que nos ceden las imágenes y nos ayudan en la clasificación, desde su conocimiento experto del dominio.

El Dr. D. Lucio Díaz Flores se ha dedicado al campo de la Anatomía Patológica durante 35 años. Los otros dos miembros del equipo son especialistas en el campo de la Histología, fundamentalmente en lo relacionado con la reparación tisular y diferenciación celular. El Dr. D. Lucio Díaz Flores ha publicado varios libros [Díaz-Flores et al., 1982], [Díaz-Flores et al., 1979], [Díaz-Flores et al., 1978], [Díaz-Flores et al., 1977], [Díaz-Flores et al., 1974], y cuenta con más de 400 publicaciones.

Este hospital cuenta con una Unidad de Investigación, a la que pertenece este equipo médico. Esta Unidad intenta estimular, apoyar y realizar una Investigación Orientada a Pacientes. Dicha investigación recurre a distintas estrategias, generalmente complementarias, con el fin de contribuir a resolver los problemas sanitarios más importantes. Una de las principales metas de esta unidad y, por tanto, del equipo de médicos con el que colaboramos, es automatizar muchas técnicas, en especial las relativas al diagnóstico, con el objetivo de realizar diagnósticos de manera más rápida y segura.

Este equipo de investigación constituye una referencia básica para esta investigación, debido a su gran experiencia en el campo de la detección de cáncer y del diagnóstico por medio de imágenes digitalizadas de pruebas médicas (citologías e histologías). Entre las técnicas histológicas que emplean se pueden citar las siguientes: la histología estándar en parafina, la inmunocitoquimia, la biopsia ósea sin decalcificar y la hibridación in situ.

En el problema que nos ocupa en esta sección, el equipo médico nos proporcionó una base de imágenes de citologías de fluidos peritoneales. En la figura 8.18 se muestran dos imágenes procedentes de esta base de imágenes de citologías, donde aparecen tejidos sanos y tejidos patológicos.

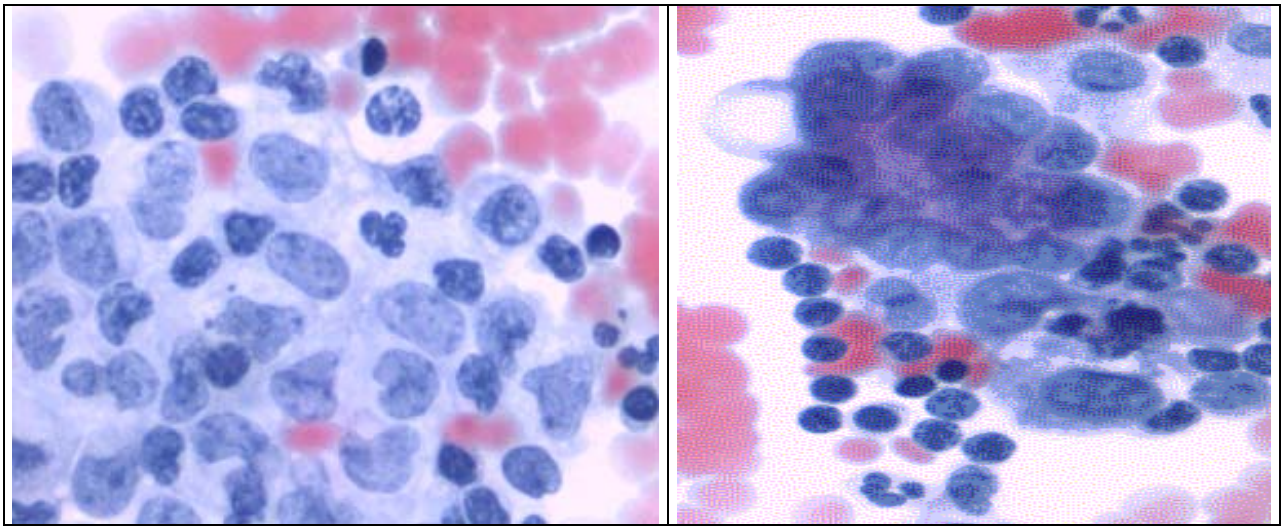


Figura 8.18. Citologías peritoneales. Derecha: tejido sano. Izquierda: tejido afectado por cáncer.

En primer lugar, se seleccionan núcleos de la imagen que no estén superpuestos y se extraen características de ellos, con el procedimiento descrito anteriormente. En el siguiente paso, se lleva a cabo la medida de la textura de cada núcleo propuesta en este trabajo. Esta medida dará lugar a la traza que representa a ese núcleo que será la entrada al sistema clasificador.

Para obtener dicha traza, se emplea la segunda aproximación de extracción de textura del núcleo, descrita en secciones anteriores. Este método está basado en contar el número de contornos que hay en cada uno de los 10 niveles con los que se construye el mapa de contornos del núcleo. Como ya se mencionó, esta nueva medida es más sencilla de obtener que las realizadas según el primer método (basado en la construcción de árboles jerárquicos o árboles homotópicos). Con este método, todas las trazas tienen la misma longitud. En este caso, las series de datos se componen de 10 valores. Se pretende comprobar que estas trazas de longitud corta son buenos descriptores de la naturaleza de los núcleos.

En la figura 8.19 se muestran los resultados de todo este procedimiento a un núcleo benigno (primera columna) y a un núcleo maligno (segunda columna), así como las trazas finales obtenidas.

En la siguiente sección se presentan los experimentos realizados para clasificar los núcleos en estas imágenes [Estévez et al., 2003].

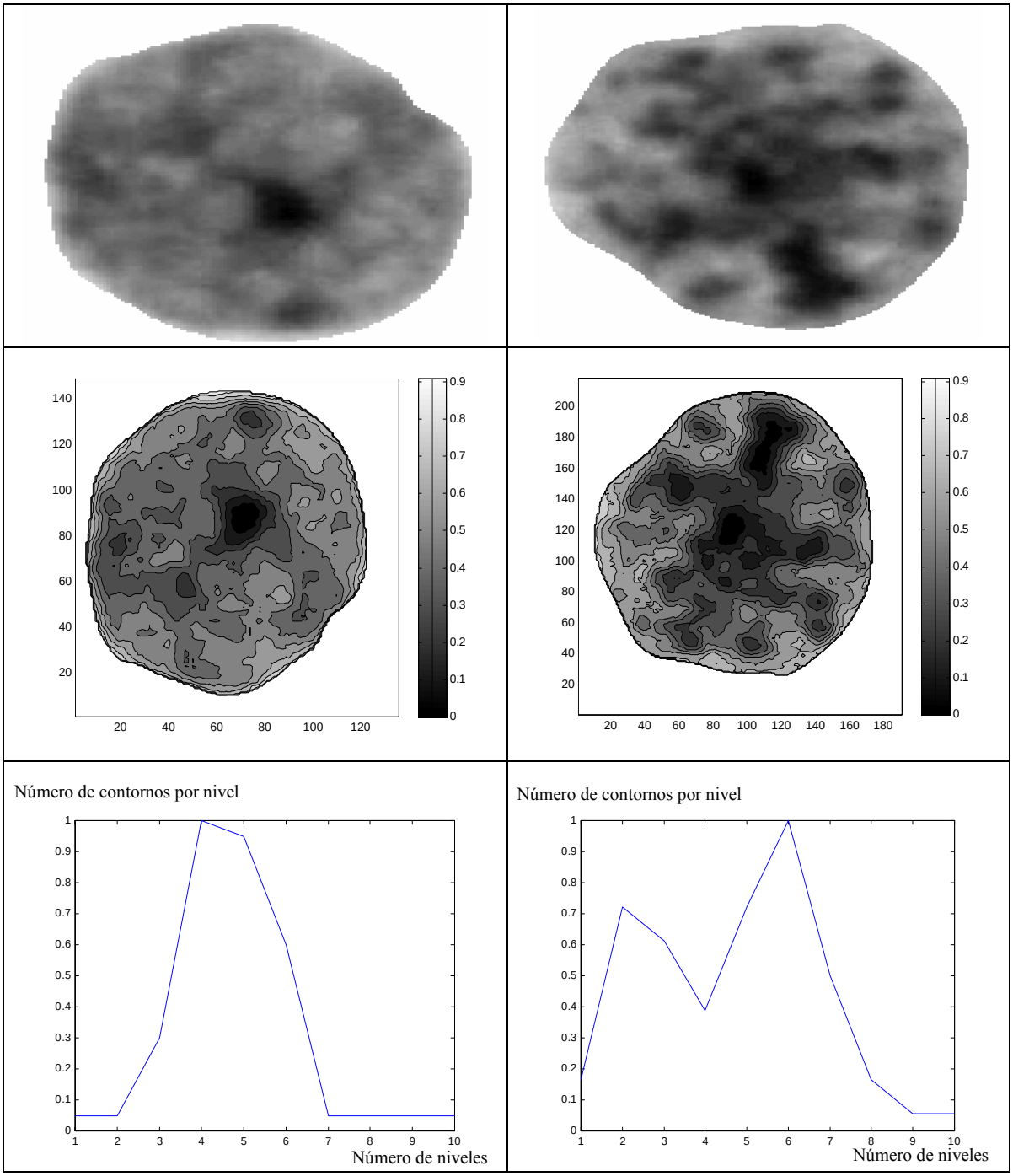


Figura 8.19. Núcleo aislado, mapa del núcleo y traza correspondiente para un núcleo benigno (primera columna) y otro maligno (segunda columna) para el problema de clasificación de núcleos en imágenes de citologías peritoneales.

8.3.2.2 Experimentos realizados y resultados.

Aplicaremos de nuevo sistemas tipo Pittsburgh para encontrar el sistema borroso recurrente capaz de clasificar los núcleos en dos clases. Para esto, previamente se construyen conjuntos de entrenamiento y test, ya que se trata de un algoritmo

supervisado. Recordemos que el sistema borroso buscado es una máquina finita de estados borrosa (FFSM). La descripción completa del sistema Pittsburgh se ha presentado en el capítulo 6.

El clasificador basado en un FFSM se entrena en el proceso de aprendizaje con el conjunto de entrenamiento. Para medir la calidad de este aprendizaje, es necesario comprobar la habilidad del clasificador FFSM para clasificar las trazas pertenecientes al conjunto de test. El clasificador se aplica a núcleos individuales, célula a célula, no imagen a imagen, con la ventaja sobre los anteriores experimentos (los relacionados con el tejido de mama) de que en esta ocasión contamos con la información exacta sobre la naturaleza de cada célula por separado.

Con las pruebas anteriores se comprobó la validez del procedimiento propuesto en esta investigación. El objetivo de estos nuevos experimentos es realizar un análisis más detallado del procedimiento, para estudiar los beneficios del empleo de FFSM en clasificación de series de datos reales. Debido a esto, se lleva a cabo una comparativa con otros métodos y una evaluación desde el punto de vista médico de los clasificadores obtenidos. A continuación, se presentan los resultados.

Resultados del entrenamiento.

El conjunto de entrenamiento se construyó con 15 trazas pertenecientes núcleos benignos y con 15 pertenecientes a núcleos malignos.

El procedimiento de aprendizaje se desarrolló bajo las condiciones que se muestran en la tabla 8.5. Se establece un umbral de parada del algoritmo más restrictivo.

Parámetro	Valor
Tamaño de la población (*num_maquinas*)	200
Umbral de parada del algoritmo (*umbral_fitness*)	0.05
Nivel que debe alcanzar el estado de detección para considerar que está activado a alta (*param_alta*)	0.7
Parámetro de selección (*alfa*)	0.5
Probabilidad para la reproducción (*p1*)	5 %
Probabilidad para la mutación (*p2*)	75 %
Probabilidad para el cruce (*p3*)	20 %
Número de reglas a mutar (*mutar_reglas*)	5
Número de elementos a mutar por regla (*mutar_elementos*)	5

Tabla 8.5. Parámetros del algoritmo de aprendizaje del sistema Pittsburgh para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Los individuos de la población son máquinas finitas de estados borrosas cuyos antecedentes y consecuentes vienen definidos por funciones de pertenencia gaussianas. El modo en que se codifican estas máquinas ha sido descrito en el capítulo 6. Las características de estos individuos se recogen en la tabla 8.6. Se ha aumentado el número de reglas de las máquinas a 10 (en las pruebas de tejido de mama las máquinas tenían 6 reglas). El motivo es que las máquinas encontradas con 6 reglas no tenían una eficiencia aceptable.

Número de reglas (*num_reglas*)	10
Número de estados (*num_estados*)	4
Número del estado de detección (*num_detec*)	4
Centro de las funciones de pertenencia (variable)	[0.1, 0.3, 0.5, 0.7, 0.9]
Desviación de las funciones de pertenencia (fijo)	0.2

Tabla 8.6. Características de las máquinas finitas de estados borrosas que componen la población en el sistema tipo Pittsburgh para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Tras 33 iteraciones, el algoritmo encontró una máquina cuyo valor de aptitud estaba por debajo del umbral (0.0333) y, por tanto, capaz de separar los dos tipos de trazas. En la figura 8.20 se muestra la curva de evolución de la aptitud.

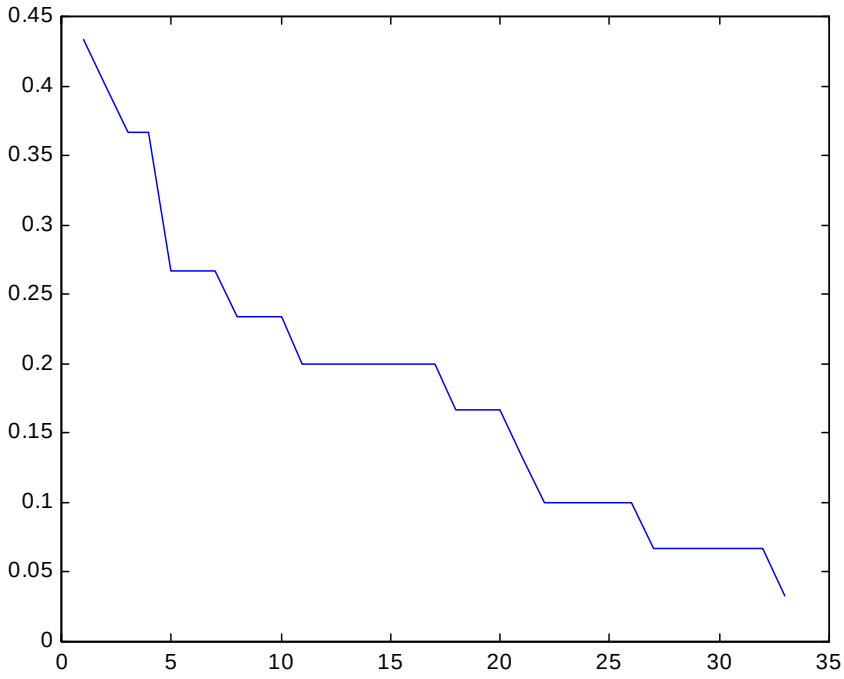


Figura 8.20. Evolución de la aptitud. Eje X: número de iteración. Eje y: valor de aptitud.

Esta máquina, junto con los centros del algoritmo de clustering, forma el clasificador final. Estos centros son [0.0495, 0.2503]. La clasificación de las trazas del conjunto de entrenamiento realizada por este sistema se muestra en la tabla 8.7.

Resultados del test.

El conjunto de test está formado por 72 trazas de células benignas y 7 trazas de células malignas, todas ellas clasificadas previamente por el experto. La clasificación de las trazas del conjunto de entrenamiento realizada por este sistema se muestra en la tabla 8.7. Se puede apreciar que el conjunto de test es bastante asimétrico. Esto es debido a las trazas de las que disponíamos en este problema en concreto. Este factor puede pesar en los resultados, al poder inducir al sistema clasificador a particularizar. En el siguiente experimento con sistemas Michigan se diseñarán conjuntos de entrenamiento y test más simétricos para evitar este inconveniente.

	Conjunto de entrenamiento (15 trazas benignas y 15 trazas malignas)	Conjunto de test (72 trazas benignas y 7 trazas malignas)	Conjunto total (87 trazas benignas y 22 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	14 trazas (93% aciertos)	55 trazas (76.38% aciertos)	69 trazas (79.31% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	1 traza (7% error)	17 trazas (23.62% error)	18 trazas (20.69% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	15 trazas (100% aciertos)	7 trazas (100% aciertos)	22 trazas (100% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	0 trazas (0% error)	0 trazas (0% error)	0 trazas (0% error)
Error global	3.33%	21.51%	16.51%

Tabla 8.7. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) para el problema de la clasificación de núcleos en imágenes de citologías peritoneales. En la tercera columna se presenta un resumen de resultados.

Estos resultados reflejan de nuevo la bondad de la medida escogida como indicadora de la naturaleza de los núcleos. Se debe destacar que, tanto en el conjunto de entrenamiento como en el conjunto de test, todos los núcleos malignos fueron clasificados correctamente y que los errores ocurrieron en la clasificación de núcleos benignos. Esto es importante desde el punto de vista médico, como comprobaremos más adelante con las curvas ROC.

En resumen, se ha conseguido encontrar mediante una búsqueda tipo Pittsburgh una FFSM que clasifica con un error de 3.33% las trazas del conjunto de entrenamiento y con un error de 21.51% las trazas del conjunto de test. De nuevo, se observa que se consiguen resultados de clasificación aceptables, y que, por lo tanto, el modo en que la cromatina del núcleo se distribuye entre las distintas escalas de los mapas de contornos utilizados proporciona bastante información de la naturaleza benigna o maligna de los núcleos.

Para validar el clasificador de un modo más exhaustivo, realizamos una validación cruzada. Construimos aleatoriamente tres grupos distintos (A, B y C) con las trazas disponibles del dominio. Los grupos A y B contienen cada uno 7 trazas de núcleos benignos y 7 de núcleos malignos. El grupo C contiene 8 trazas de cada clase. Combinando estos grupos para formar conjuntos de entrenamiento y test diferentes, se ejecutan distintas búsquedas con sistemas Pittsburgh.

El tamaño reducido de estos grupos se debe a la carencia de núcleos malignos en la base de imágenes utilizada. Por lo tanto, los resultados de esta validación cruzada deben considerarse solamente como una guía, ya que un número pequeño de muestras en el conjunto de entrenamiento afecta negativamente a la capacidad de generalización del clasificador.

	Error de clasificación sobre el conjunto de entrenamiento	Error de clasificación sobre el conjunto de test
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Primera prueba**. Conjunto de entrenamiento: grupos A-B. Conjunto de test : grupo C.	0 %	37.5 %
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Segunda prueba**. Conjunto de entrenamiento: grupos A-C. Conjunto de test : grupo B.	10 %	35.71 %
Clasificación global de la FFSM obtenida por el sistema Pittsburgh. **Tercera prueba**. Conjunto de entrenamiento: grupos B-C. Conjunto de test : grupo A.	3.33 %	28.57 %
Media de los resultados de las tres pruebas	4.44 %	33.92 %
Clasificación global de la FFSM inicial que se desea validar.	3.33 %	21.51 %

Tabla 8.8. Resultados de clasificación con validación cruzada para el problema de clasificación de núcleos en imágenes de citologías peritoneales.

En la tabla 8.8 se representan los resultados de la clasificación global (sobre todas las trazas benignas y malignas de las que se dispone) de las pruebas realizadas con los distintos conjuntos de entrenamiento y test. Además, se añade una comparativa entre la media del error de las FFSM de estas pruebas y el error cometido por la FFSM que se encontró inicialmente y que se desea validar.

Como se puede apreciar, los resultados son aceptables y la validación cruzada confirma la eficiencia de la FFSM encontrada. En los siguientes puntos, se analiza el mismo problema de clasificación con otras metodologías de reconocimiento de patrones, para comparar los resultados de clasificación con sistemas convencionales y la complejidad del problema de clasificación. Además, se evalúan los clasificadores desde el punto de vista del diagnóstico médico, con el análisis de curvas ROC.

Comparación con otros métodos de clasificación y reconocimiento de patrones.

El primer método de clasificación utilizado fue un método de clustering borroso no supervisado (fuzzy c-means clustering). Se aplicó sobre el conjunto total (conjunto de entrenamiento y conjunto de test: 87 trazas benignas y 22 trazas malignas). Los resultados de la clasificación según este algoritmo se recogen en la tabla 8.9 junto con una comparativa de estos resultados y los resultados de la FFSM bajo estudio encontrada con el sistema Pittsburgh.

	Clustering borroso (Fuzzy k-means clustering)	Clasificador con sistema borroso recurrente (FFSM)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	65 trazas (77.41% aciertos)	69 trazas (79.31% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	22 trazas (25.29% error)	18 trazas (20.69% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	17 trazas (77.21% aciertos)	22 trazas (100% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	5 trazas (22.73% error)	0 trazas (0% error)

Tabla 8.9. Resultados en la clasificación del conjunto total de trazas por el algoritmo de clustering borroso (primera columna) y por el sistema clasificador con FFSM bajo estudio (segunda columna) para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Estos resultados sirven para obtener una impresión previa de la dificultad de la clasificación sobre el espacio de vectores de características definido por las trazas

extraídas. Es evidente que estos resultados aunque indicativos de la existencia de dos regiones diferenciadas no son buenos, lo que nos lleva al uso de técnicas más elaboradas.

Los siguientes clasificadores que se aplicaron estaban basados en redes neuronales con propagación hacia adelante y supervisadas. Se han probado tres redes: con una neurona (red 1), dos neuronas (red 2) y tres neuronas (red 3) en la capa interna. Se entrenaron con el mismo conjunto de entrenamiento con que entrenamos el sistema Pittsburgh que buscaba la máquina de estados borrosa y se realizó el test con el mismo conjunto de test. A modo de resumen, en la tabla 8.10 se muestran los resultados sobre el conjunto de test de las distintas redes neuronales probadas. Además, se presenta la comparativa entre estos resultados y los obtenidos con la FFSM considerada.

	Red Neuronal 1	Red Neuronal 2	Red Neuronal 3	Clasificador con sistema borroso recurrente (FFSM)
Aciertos en la clasificación de núcleos benignos.	31 trazas (43.05% aciertos)	55 trazas (76.38% aciertos)	48 trazas (66.66% aciertos)	55 trazas (76.38% aciertos)
Errores en la clasificación de núcleos benignos.	41 trazas (56.95% error)	17 trazas (23.62% error)	24 trazas (33.34% error)	17 trazas (23.62% error)
Aciertos en la clasificación de núcleos malignos.	5 trazas (71.42% aciertos)	5 trazas (71.42% aciertos)	5 trazas (71.42% aciertos)	7 trazas (100% aciertos)
Errores en la clasificación de núcleos malignos.	2 trazas (28.58% error)	2 trazas (28.58% error)	2 trazas (28.58% error)	0 trazas (0% error)

Tabla 8.10. Resultados de las redes neuronales (columnas 1, 2 y 3). Comparación con el clasificador FFSM obtenido (última columna).

Comparándolas entre ellas, se observa que las tres tienen el mismo porcentaje de aciertos y fallos en la clasificación de las trazas malignas, pero la red neuronal que tiene dos neuronas en la primera capa se comporta mejor en la clasificación de las trazas benignas, ya que tiene el mayor porcentaje de aciertos.

Comparando estos resultados con los resultados del clasificador basado en la máquina de estados borrosa, se aprecia claramente que la máquina tiene mejores resultados (100% aciertos en las trazas malignas, 76.38% aciertos en las trazas benignas), lo que puede ser un indicador de que su capacidad de generalización es

mayor. Lo más importante en este tipo de clasificadores es que sean capaces de clasificar las trazas malignas sin error ya que es lo que más ayuda al diagnóstico de este tipo de patologías. Desde esta perspectiva, se observa que es mejor el clasificador constituido por el sistema borroso.

Evaluación con curvas ROC.

Es importante evaluar estos clasificadores desde el punto de vista de ayuda al diagnóstico. Para ello, utilizamos el análisis ROC, una metodología desarrollada en el seno de la Teoría de la Decisión en los años 50 y que ha sido muy aplicada en el ámbito de la biomedicina. Esta técnica de evaluación de las prestaciones de los clasificadores en medicina ha sido ampliamente detallada en el capítulo 3.

En la toma de decisiones clínicas es necesario valorar la utilidad de cualquier prueba diagnóstica, es decir, conocer su exactitud o capacidad de clasificar correctamente a los pacientes en distintas categorías. Las categorías típicas son: estar enfermo /no estar enfermo, respuesta positiva /negativa a la terapia, etc. En nuestro caso, los núcleos se clasifican según la categoría núcleo benigno /maligno.

Recordemos que la exactitud diagnóstica se mide en términos de *sensibilidad* y *especificidad*. La sensibilidad representa la probabilidad de clasificar correctamente a un individuo cuyo estado real sea definido como "positivo" respecto a la condición que estudia la prueba y la especificidad es la probabilidad de clasificar correctamente a un individuo cuyo estado real sea definido como "negativo" por la prueba. La curva ROC se construye mediante esta representación de los pares (1-especificidad, sensibilidad) obtenidos al considerar todos los posibles valores de corte de la prueba. Esta gráfica nos proporciona una representación global de la exactitud diagnóstica.

En el clasificador obtenido con el sistema tipo Pittsburgh basado en maquinas de estado borrosas, el valor de corte es el umbral que debe superar el nivel de activación del estado de detección para considerar que este estado está activado a alta (parámetro *param_alta*). Haciendo un barrido de este parámetro y analizando los resultados de la clasificación (aciertos y errores) se calculan los pares (sensibilidad, 1-especificidad) y la curva ROC representada en la figura 8.21.

Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	0.9771	1
0.4	0.8851	1
0.5	0.7702	1
0.6	0.6897	1
0.65	0.4828	1
0.68	0.3679	1
0.7	**0.2069**	**1**
0.72	0.1035	0.7272
0.74	0	0
0.8	0	0
0.9	0	0
1	0	0

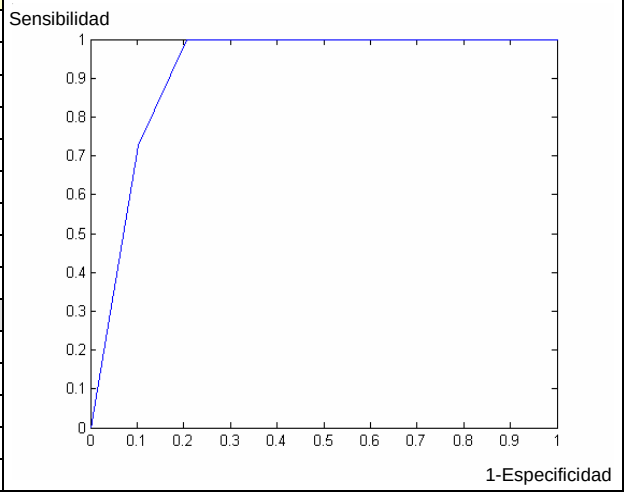


Figura 8.21. Curva ROC del clasificador basado en la máquina de estados borrosa encontrado con el sistema Pittsburgh para el problema de clasificación de núcleos en imágenes de citologías peritoneales.

El mejor clasificador se obtiene con un valor de corte igual a 0.7, ya que con ese valor se obtiene una sensibilidad igual a 1 (100% aciertos en núcleos malignos) y la mayor especificidad posible (mayor porcentaje de aciertos en núcleos benignos). Este valor de corte coincide con el valor en que hemos fijado el parámetro *param_alta*. Esta coincidencia indica la adecuación de la medida de aptitud utilizada en el proceso evolutivo.

Repitiendo estos mismos cálculos para los clasificadores basados en redes neuronales, donde los valores de corte se corresponden a distintos umbrales de discriminación, se obtienen las curvas ROC de las figuras 8.22, 8.23 y 8.24.

Valor de corte	1-Especificidad	Sensibilidad
0	0	0.3181
0.1	0	0.4545
0.2	0	0.5000
0.3	0	0.6818
0.4	0	0.8181
0.5	0.0690	0.8181
0.6	0.1380	0.8181
0.7	0.2759	0.8636
0.8	**0.5058**	**0.9090**
0.9	0.7932	0.9090
1	1	1

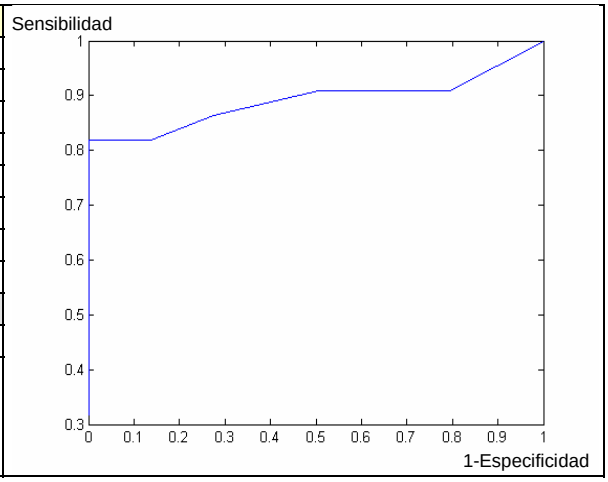


Figura 8.22. Curva ROC obtenida para el clasificador formado por la red neuronal 1 para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Valor de corte	1-Especificidad	Sensibilidad
0	0.0345	0.2727
0.1	**0.1150**	**0.9090**
0.2	0.1150	0.9090
0.3	0.1265	0.9090
0.4	0.1380	0.9090
0.5	0.1495	0.9090
0.6	0.1610	0.9090
0.7	0.1610	0.9090
0.8	0.1610	0.9090
0.9	0.1610	0.9090
1	1	1

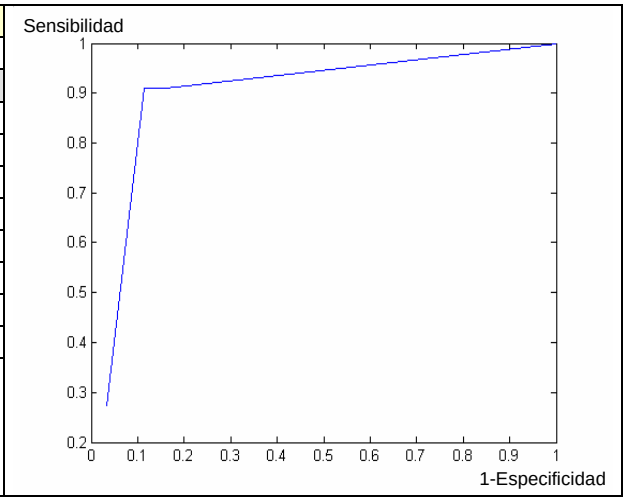


Figura 8.23. Curva ROC obtenida para el clasificador formado por la red neuronal 2 para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Valor de corte	1-Especificidad	Sensibilidad
0	0.1955	0.7727
0.1	**0.2989**	**0.8636**
0.2	0.3104	0.8636
0.3	0.3219	0.8636
0.4	0.3219	0.8636
0.5	0.3219	0.8636
0.6	0.3219	0.8636
0.7	0.3219	0.8636
0.8	0.3219	0.8636
0.9	0.3449	0.8636
1	1	1

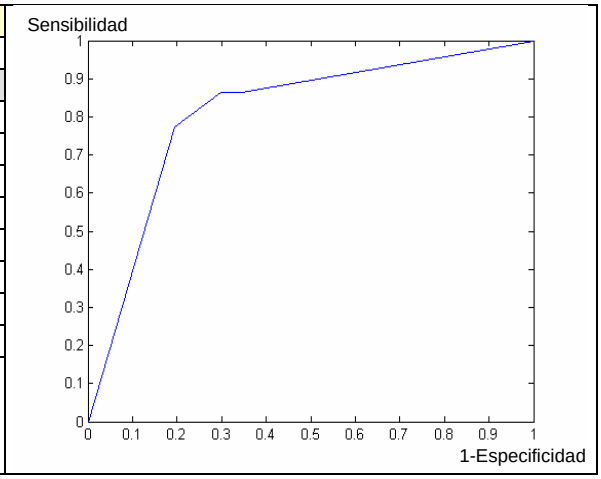


Figura 8.24. Curva ROC obtenida para el clasificador formado por la red neuronal 3 para el problema de la clasificación de núcleos en imágenes de citologías peritoneales.

Se puede observar que la red neuronal 1 clasifica mejor cuando su valor de corte está establecido en 0.8, punto donde la sensibilidad y la especificidad son las mayores posibles, mientras que para las redes neuronales 2 y 3 esto sucede con un valor de corte de 0.1. Esto corresponde a los siguientes pares (sensibilidad, 1-especificidad): (0.9090, 0.5058) para la primera red neuronal, (0.9090,0.1150) para la segunda y (0.8636,0.2989) para la tercera.

Como se mencionó en el capítulo 3, la exactitud de la prueba aumenta a medida que la curva se desplaza desde la diagonal hacia el vértice superior izquierdo. Un clasificador ideal (100% de sensibilidad y 100% de especificidad) pasaría por dicho vértice. Un modo de comparar los clasificadores, es comparar sus curvas ROC. A simple vista se aprecia que la curva correspondiente al clasificador basado en la

máquina de estados borrosa es la que más se aproxima a este vértice, por lo tanto, desde el punto de vista médico, es el método con mejores resultados de clasificación y el que más podría ayudar al diagnóstico.

8.3.2.3 Conclusiones.

Estos experimentos constituyen un estudio más detallado de la aplicación de máquinas de estado borrosas obtenidas mediante sistemas tipo Pittsburgh en la tarea de clasificación de patrones reales.

El objetivo de los experimentos presentados en esta sección es realizar la clasificación de núcleos benignos y malignos en imágenes de citologías de fluidos peritoneales. Para ello, se busca la FFSM capaz de clasificar estos núcleos mediante un sistema evolutivo de búsqueda tipo Pittsburgh.

Los resultados obtenidos confirman la capacidad de la FFSM de clasificar series de datos reales. La máquina encontrada clasifica con un error de 3.33% las trazas del conjunto de entrenamiento y con un error de 21.51% las trazas del conjunto de test. Estos resultados de clasificación son aceptables, y han sido comparados con los obtenidos en la validación cruzada. Los resultados de la validación cruzada no son tan buenos como se esperaba, debido seguramente a la limitación que nos supone tener pocas trazas en los conjuntos de entrenamiento y test en estas pruebas. En los siguientes experimentos se incrementará el número de trazas en cada conjunto.

Aún así, estos resultados son indicativos de que la medida que realizamos sobre cada núcleo proporciona información suficiente para caracterizarlos. Esta medida implica la construcción de una traza que recoge el modo en que la cromatina del núcleo se distribuye entre las distintas escalas de los mapas de contornos utilizados, con lo que se pone de manifiesto que la textura de los núcleos es muy buen indicador de la naturaleza benigna o maligna de los mismos. Existen otras características que en trabajos futuros se considerarán, para completar la información y poder realizar una clasificación más eficiente. Dentro de estas características se podrían citar las siguientes: el tamaño y forma de las células, relación núcleo-citoplasma, irregularidades de la frontera, etcétera.

Tras estudiar los resultados de clasificación de la FFSM sobre las series de datos reales y compararlos con los resultados de clasificación de los otros métodos probados

(clustering borroso y redes neuronales), se puede llegar a la conclusión de que el sistema clasificador constituido por la FFSM presenta una mayor capacidad de generalización en el caso concreto estudiado. El motivo puede ser atribuible al mayor número de parámetros libres en la FSSM en las pruebas realizadas unido a la utilización de un método de optimización global para su síntesis, pero este extremo debe ser analizado más cuidadosamente en futuras investigaciones.

Otro punto destacable y relacionado con el problema de la determinación del número de reglas para la FSSM es la observación de que un número insuficiente de reglas provoca un "estancamiento" en el proceso evolutivo pudiendo ser debido no necesariamente a una incapacidad estructural de la máquina para clasificar los patrones, sino a la carencia de intrones [Banzhaf, 1998] que protejan a las reglas importantes de su destrucción en el proceso evolutivo. De hecho, estos intrones se han observado en las máquinas sintetizadas al encontrarse reglas con activaciones despreciables y que, por lo tanto, no juegan un papel en el proceso de clasificación. Este es el motivo de que en estas pruebas se trabaje con máquinas con mayor número de reglas.

Nos parece reseñable la característica mencionada acerca del umbral de corte predefinido para el parámetro extraído del estado de detección, cuyo valor coincide con el que según las curvas ROC produce una sensibilidad igual a 1 (máxima) y la mayor especificidad posible, lo que indica la adecuación de la expresión elegida para la aptitud. También es importante destacar la comparativa de curvas ROC que es ventajosa para la máquina de estados borrosa frente al resto de algoritmos estudiados. El trabajo futuro debe llevarnos a una comparativa más extensa y a un análisis más profundo de los resultados obtenidos.

En todos los experimentos presentados hasta el momento se ha utilizado un sistema de búsqueda tipo Pittsburgh para encontrar FFSMs con eficiencia de clasificación aceptable. Los resultados demuestran la capacidad de los sistemas tipo Pittsburgh para encontrar FFSMs con estas características. En los experimentos presentados en la siguiente sección, se utilizarán sistemas tipo Michigan para encontrar estas FFSMs. El objetivo será verificar si este método de búsqueda es también apropiado para el problema que nos ocupa.

8.3.3 Clasificación de núcleos en imágenes de citologías pleurales.

8.3.3.1 Descripción del problema.

En los experimentos que se presentan a continuación, nuestro objetivo es clasificar correctamente núcleos de células sanas y núcleos de células patológicas en imágenes digitalizadas de citologías pleurales. La principal diferencia entre estas pruebas y las anteriormente descritas es el método de búsqueda seguido para encontrar la FFSM capaz de realizar esta clasificación. En vez de trabajar con sistemas tipo Pittsburgh, utilizaremos sistemas tipo Michigan.

Este tipo de citologías también se realiza por medio de punción y extracción de líquido, seguido de un proceso de centrifugado y tinción. Con estas pruebas, el especialista puede diagnosticar tumores primarios fundamentalmente de pulmón y pleura (originarios de esas zonas) y tumores secundarios (metastásicos).

Para tal fin contamos con la colaboración del equipo médico mencionado anteriormente, que nos cede las imágenes y nos ayuda en la clasificación, desde su conocimiento experto del dominio.

En el problema que nos ocupa en esta sección, el equipo médico nos proporcionó una base de imágenes de citologías pleurales. En la figura 8.25 se muestran dos imágenes procedentes de esta base de imágenes de citologías, donde aparecen tejidos sanos y tejidos patológicos.

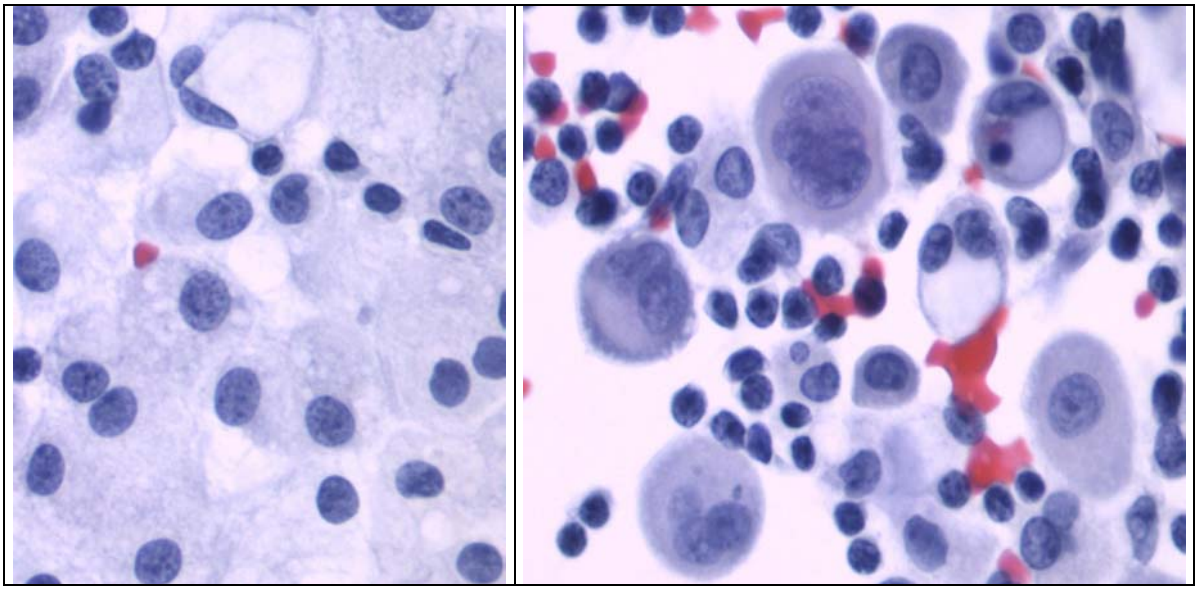


Figura 8.25. Izquierda: tejido sano. Derecha: tejido afectado por cáncer.

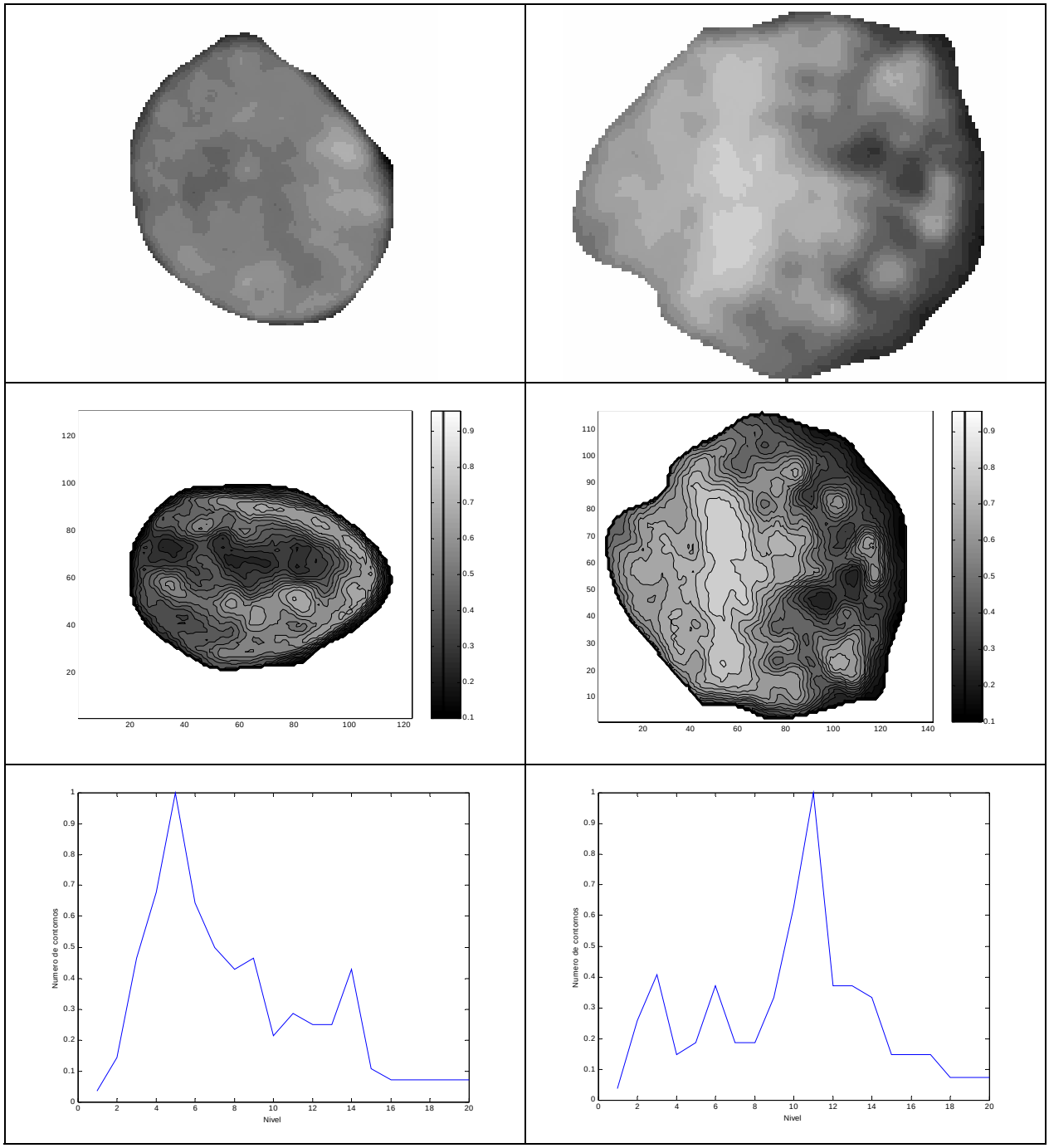


Figura 8.26. Núcleo aislado, mapa del núcleo y traza correspondiente para un núcleo benigno (primera columna) y otro maligno (segunda columna) para el problema de clasificación de núcleos en imágenes de citologías pleurales.

Una vez seleccionados los núcleos que se desean clasificar, el procedimiento a seguir para la construcción de las trazas es similar al realizado en los anteriores experimentos. La medida de textura aplicada en estas pruebas es la segunda medida propuesta, consistente en contar el número de contornos que hay en cada nivel del mapa topográfico del núcleo. Los mapas realizados para los núcleos de estas imágenes se

distribuyen en 20 niveles. Se sigue utilizando esta medida porque es más sencilla que la construcción de árboles homotópicos. Además, en los experimentos de la sección anterior posibilitó unos buenos resultados de clasificación.

En la figura 8.26, se muestran los resultados de todo este procedimiento a un núcleo benigno (primera columna) y a un núcleo maligno (segunda columna), así como las trazas finales obtenidas.

En la siguiente sección se presentan los experimentos realizados para clasificar los núcleos en estas imágenes.

8.3.3.2 Experimentos realizados y resultados.

En estos experimentos se han aplicado sistemas tipo Michigan para encontrar un sistema borroso recurrente capaz de clasificar los núcleos en dos clases. Para esto, previamente se construyen conjuntos de entrenamiento y test, ya que también se trata de un algoritmo supervisado. Recordemos que el sistema borroso buscado es una máquina finita de estados borrosa (FFSM). La descripción completa del sistema Michigan se ha presentado en el capítulo 6.

De igual modo que en los sistemas tipo Pittsburgh, en los sistemas tipo Michigan el clasificador basado en un FFSM se entrena con el conjunto de entrenamiento. Para medir la calidad de este aprendizaje, es necesario comprobar la habilidad del clasificador FFSM para discriminar correctamente las trazas pertenecientes al conjunto de test. El clasificador se aplica a núcleos individuales.

Con las pruebas anteriores, se comprobó la validez de la metodología Pittsburgh en este tipo de problemas. El objetivo de estos nuevos experimentos es comprobar la validez de la metodología Michigan en el mismo tipo de problema de clasificación de series de datos reales.

En los sistemas Michigan la etapa de entrenamiento es computacionalmente menos costosa que en los sistemas tipo Pittsburgh. Recordemos que en Pittsburgh se trabaja con una población de FFSMs (200 FFSMs en las pruebas ya presentadas) mientras que en Michigan sólo se considera en cada iteración una FFSM. Por lo tanto, en Michigan sólo se realiza una evaluación por iteración, mientras que en Pittsburgh se realizan 200 evaluaciones. El objetivo de aplicar los sistemas Michigan sobre el mismo problema de clasificación es comprobar si se pueden obtener FFSM que clasifiquen con eficiencia similar a las obtenidas por el sistema Pittsburgh, en menos tiempo.

Se pretende realizar un análisis detallado del procedimiento. Por este motivo, se han desarrollado pruebas con diferentes conjuntos de entrenamiento y de test obtenidos de una muestra de células común. Además, se ha realizado una comparativa con otros métodos de clasificación (clustering borroso, redes neuronales con propagación hacia delante e identificación del HMM) y una evaluación desde el punto de vista médico de los clasificadores obtenidos con el análisis de las curvas ROC. A continuación, se presentan los resultados.

Descripción de los experimentos.

Se ha dividido el conjunto total de trazas disponibles en tres subconjuntos (A, B y C). Se han realizado seis experimentos, en cada uno de ellos los subconjuntos A, B y C juegan el papel de conjunto de entrenamiento o conjunto de test, según se muestra en la tabla 8.11. El grupo A está formado por 20 trazas correspondientes a núcleos benignos y 20 trazas de núcleos malignos. Los grupos B y C están compuestos cada uno por 21 trazas de núcleos benignos y 21 trazas de núcleos malignos.

	Conjunto de entrenamiento	Conjunto de test
Experimento 1	A	B + C
Experimento 2	B	A+C
Experimento 3	C	A+B
Experimento 4	A+B	C
Experimento 5	A+C	B
Experimento 6	B+C	A

Tabla 8.11. Combinación entre los distintos grupos de trazas A, B y C para constituir los distintos conjuntos de entrenamiento y de test en los experimentos con sistemas Michigan desarrollados para el problema de clasificación de núcleos en imágenes de citologías pleurales.

Se han realizado tres búsquedas con sistemas tipo Michigan en cada experimento obteniéndose, por lo tanto, tres máquinas finitas de estados borrosas distintas para clasificar las trazas de los núcleos de las imágenes de citologías pleurales por experimento. Todas las pruebas se hacen en el mismo número de iteraciones (1000 iteraciones).

Todos los procedimientos de aprendizaje se desarrollaron bajo las mismas condiciones. Estas condiciones se muestran en la tabla 8.12. En el capítulo 6 se ha descrito en detalle el algoritmo implementado en el sistema Michigan.

Parámetro	Valor
Nivel que debe alcanzar el estado de detección para considerar que está activado a alta (*param_alta*)	0.6
Factor utilizado en el proceso de recompensa (negativa y positiva) de las meta-reglas (*factor_recompensa*)	50
Factor que interviene en el proceso de penalización de las meta-reglas (*penalizacion*)	0.01
Factor que se utiliza en la comparación con la media de fuerzas para realizar el borrado (*factor_borrado*)	0.001
Límite de experiencia exigido para realizar el borrado (*umbral_experiencia*)	3
Número de encajes mínimo exigido (*num_encajes*)	50
Valor de fuerza que se desea que tenga una meta-regla al ser creada por primera vez (*fuerza_inicial*)	5
Valor de experiencia que se desea que tenga una meta-regla al ser creada por primera vez (*experiencia_inicial*)	0
Probabilidad de introducir símbolos *don´t care* en la parte de comparación (primer campo) de la meta-regla que se genera en el proceso de recubrimiento (*p_cov*)	40%
Antigüedad requerida en las meta-reglas que encajan para disparar el algoritmo genético (*umbral_antigüedad*)	20
Porcentaje de elementos de *M* cuya antigüedad debe superar el umbral de antigüedad *umbral_antiguedad* para que se dispare un algoritmo genético (*porción*)	0
Número mínimo de iteraciones que deben pasar desde el último algoritmo genético que se ha disparado antes de disparar uno nuevo (*min_iter*)	15
Parámetro de selección (*alfa*)	0.5
Probabilidad para la reproducción (*p1*)	5%
Probabilidad para la mutación (*p2*)	60%
Probabilidad para el cruce (*p3*)	35%
Probabilidad de realizar una mutación sobre un elemento de una meta-regla (*p_mut*)	60%
Probabilidad de mutación de la parte de comparación de una meta-regla (*p_mut2*)	50%

Tabla 8.12. Parámetros del algoritmo de aprendizaje del sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales.

El parámetro *porción* tiene asignado un valor 0, lo que quiere decir que para disparar el algoritmo genético sobre el conjunto de meta-reglas que encajan en la iteración considerada, *M*, no se toma en consideración el porcentaje de elementos que superan un cierto umbral de antigüedad. En el capítulo 7 se comprobó, en base a los resultados obtenidos con los experimentos realizados sobre datos simulados (series de datos de modelos ocultos de Markov), que este criterio no era muy bueno. Por lo tanto, se sigue el segundo criterio propuesto, ya que dio mejores resultados en las pruebas citadas: el algoritmo genético se dispara sobre *M* con una frecuencia constante, es decir,

cada *min_iter* iteraciones. El valor de *min_iter* se ha fijado en un valor de 15, por ser este valor el que daba lugar a las FFSM con mejores resultados de clasificación en las pruebas sobre datos simulados. El valor del resto de los parámetros también se ha fijado teniendo en cuenta los valores que mejor funcionaban en los experimentos sobre series de datos simuladas.

Como ya se ha mencionado antes, en el sistema Michigan se trabaja solamente con una FFSM en vez de con una población de FFSMs. Los antecedentes y consecuentes de esta FFSM vienen definidos por funciones de pertenencia gaussianas. El modo en que se codifica esta máquina ha sido descrito en el capítulo 6. Las características de la FFSM utilizada se recoge en la tabla 8.13.

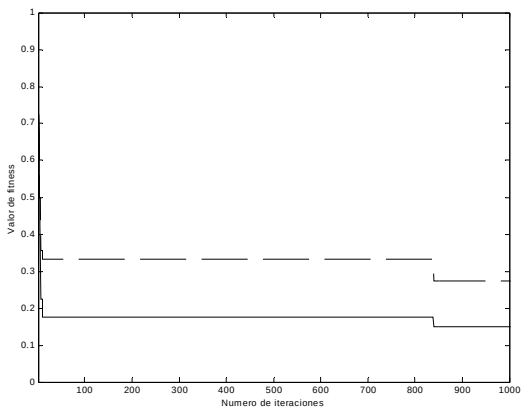
Número de reglas (*num_reglas*)	10
Número de estados (*num_estados*)	4
Configuración para la generación de consecuentes (*num_puntos*)	6
Número del estado de detección (*num_detec*)	4
Centro de las funciones de pertenencia (variable)	[0.1, 0.3, 0.5, 0.7, 0.9]
Desviación de las funciones de pertenencia (fijo)	0.2

Tabla 8.13. Características de la máquina finita de estados borrosa utilizada en el sistema tipo Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales.

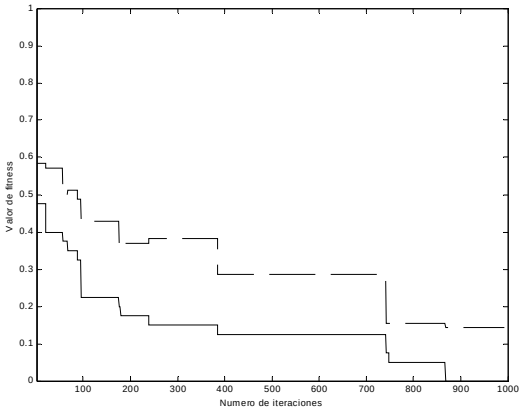
Primer experimento.

En la figura 8.27 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.

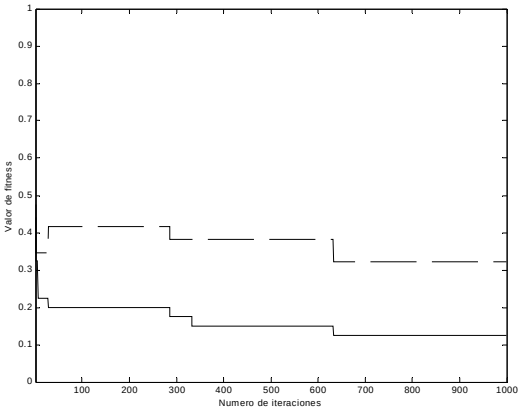
Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.14, 8.15 y 8.16.



Error en entrenamiento: 15%
Error en test: 27.38%



Error en entrenamiento: 0%
Error en test: 14.29%



Error en entrenamiento: 12.5%
Error en test: 32.14%

Figura 8.27. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 1 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

	Conjunto de entrenamiento (20 trazas benignas y 20 trazas malignas)	Conjunto de test (42 trazas benignas y 42 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	14 trazas (70% aciertos)	19 trazas (45.24% aciertos)	33 trazas (53.22% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	6 trazas (30% error)	23 trazas (54.76% error)	29 trazas (46.77% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	20 trazas (100% aciertos)	42 trazas (100% aciertos)	62 trazas (100% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	0 trazas (0% error)	0 trazas (0% error)	0 trazas (0% error)

Tabla 8.14. Primer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (20 trazas benignas y 20 trazas malignas)	Conjunto de test (42 trazas benignas y 42 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	20 trazas (100% aciertos)	36 trazas (85.71% aciertos)	56 trazas (90.32% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	0 trazas (0% error)	6 trazas (14.29% error)	6 trazas (9.68% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	20 trazas (100% aciertos)	40 trazas (95.23% aciertos)	60 trazas (96.77% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	0 trazas (0% error)	2 trazas (4.77% error)	2 trazas (3.23% error)

Tabla 8.15. Primer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (20 trazas benignas y 20 trazas malignas)	Conjunto de test (42 trazas benignas y 42 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	17 trazas (85% aciertos)	25 trazas (59.52% aciertos)	42 trazas (67.75% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	3 trazas (15% error)	17 trazas (40.48% error)	20 trazas (32.25% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	18 trazas (90% aciertos)	32 trazas (76.2% aciertos)	50 trazas (80.65% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	2 trazas (10% error)	10 trazas (23.8% error)	12 trazas (19.35% error)

Tabla 8.16. Primer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.17, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.18 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	FSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	33 trazas (53.22% aciertos)	56 trazas (90.32% aciertos)	42 trazas (67.75% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	29 trazas (46.77% error)	6 trazas (9.68% error)	20 trazas (32.25% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	62 trazas (100% aciertos)	60 trazas (96.77% aciertos)	50 trazas (80.65% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	0 trazas (0% error)	2 trazas (3.23% error)	12 trazas (19.35% error)

Tabla 8.17. Experimento 1. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	15%	27.38%
FFSM 2	0%	14.29%
FFSM 3	12.5%	32.14%
Media	9.1667%	24.6033%
Varianza	8.0364	9.2433

Tabla 8.18. Comparativa de resultados de las tres FFSMs obtenidas en el primer experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

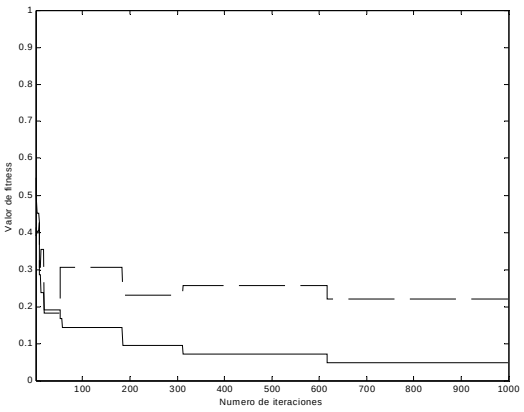
A la vista de estos resultados se puede afirmar que hemos encontrado FFSMs por el procedimiento de Michigan de eficiencia aceptable, como las encontradas por los sistemas Pittsburgh en los experimentos presentados en secciones anteriores.

Se debe resaltar que la segunda máquina encontrada tiene unos resultados de clasificación especialmente buenos, tanto en la clasificación de núcleos benignos (9.68% error) como en la de núcleos malignos (3.23% error). Es la que presenta una clasificación ideal sobre el conjunto de entrenamiento (0% error) y el menor error en la clasificación del conjunto de test (14.29%).

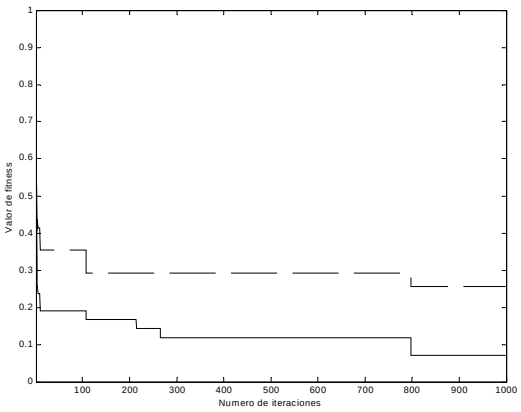
Las restantes máquinas clasifican mejor las trazas de núcleos malignos (0% error para la primera y 19.35% error para la tercera), y sus resultados de clasificación sobre el conjunto de entrenamiento no son malos, pero los del test reflejan la dificultad que tienen para aumentar su capacidad de generalización.

Segundo experimento.

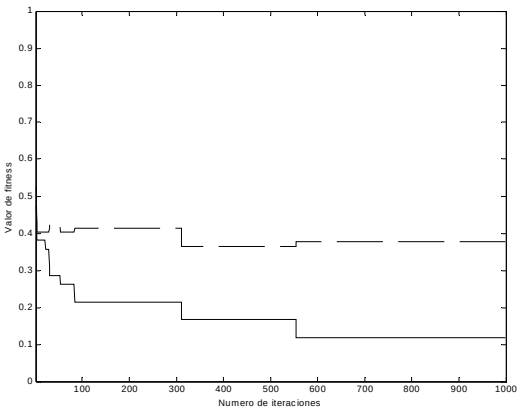
En la figura 8.28 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.



Error en entrenamiento: 4.76%
Error en test: 21.95%



Error en entrenamiento: 7.14%
Error en test: 25.61%



Error en entrenamiento: 11.9%
Error en test: 37.8%

Figura 8.28. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 2 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.19, 8.20 y 8.21.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	21 trazas (100% aciertos)	36 trazas (87.80% aciertos)	57 trazas (91.93% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	0 trazas (0% error)	5 trazas (12.19% error)	5 trazas (8.07% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	19 trazas (90.47% aciertos)	29 trazas (70.73% aciertos)	48 trazas (77.42% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	2 trazas (9.53% error)	12 trazas (29.27% error)	14 trazas (22.58% error)

Tabla 8.19. Segundo experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	21 trazas (100% aciertos)	36 trazas (87.8% aciertos)	57 trazas (91.94% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	0 trazas (0% error)	5 trazas (12.2% error)	5 trazas (8.06% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	18 trazas (85.71% aciertos)	24 trazas (58.54% aciertos)	42 trazas (67.75% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	3 trazas (14.28% error)	17 trazas (41.46% error)	20 trazas (32.25% error)

Tabla 8.20. Segundo experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.22, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.23 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	18 trazas (85.72% aciertos)	27 trazas (65.85% aciertos)	45 trazas (72.58% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	3 trazas (14.28% error)	14 trazas (34.15% error)	17 trazas (27.41% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	19 trazas (90.48% aciertos)	29 trazas (70.73% aciertos)	48 trazas (77.42% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	2 trazas (9.52% error)	12 trazas (29.27% error)	14 trazas (22.58% error)

Tabla 8.21. Segundo experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	FFSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	57 trazas (91.93% aciertos)	57 trazas (91.94% aciertos)	45 trazas (72.58% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	5 trazas (8.07% error)	5 trazas (8.06% error)	17 trazas (27.41% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	48 trazas (77.42% aciertos)	42 trazas (67.75% aciertos)	48 trazas (77.42% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	14 trazas (22.58% error)	20 trazas (32.25% error)	14 trazas (22.58% error)

Tabla 8.22. Experimento 2. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	4.76%	21.95%
FFSM 2	7.14%	25.61%
FFSM 3	11.9%	37.80%
Media	7.9333%	28.4533%
Varianza	3.6355	8.2987

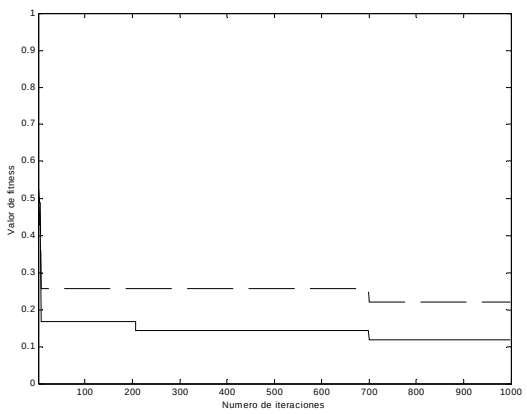
Tabla 8.23. Comparativa de resultados de las tres FFSMs obtenidas en el segundo experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Las FFSMs encontradas en este experimento presentan una eficiencia de clasificación peor que las encontradas en el primer experimento.

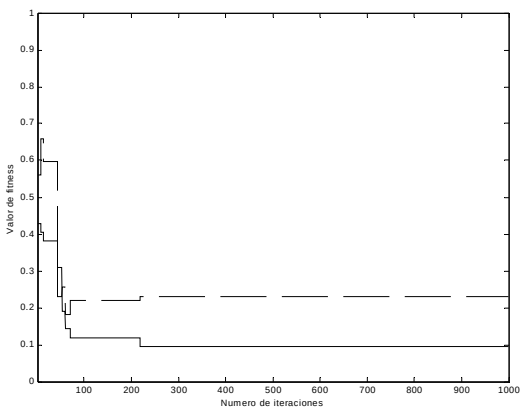
La primera máquina es la que tiene mejores resultados en la clasificación de núcleos benignos (8.07% error) y de núcleos malignos (22.58% error), y la que presenta la mejor clasificación del conjunto de entrenamiento y del conjunto de test.

Tercer experimento.

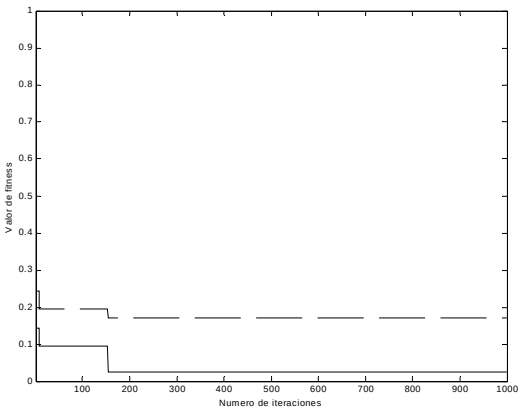
En la figura 8.29 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.



Error en entrenamiento: 11.9%
Error en test: 21.95%



Error en entrenamiento: 9.52%
Error en test: 23.17%



Error en entrenamiento: 2.38%
Error en test: 17.07%

Figura 8.29. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 3 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.24, 8.25 y 8.26.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	18 trazas (85.72% aciertos)	31 trazas (75.6% aciertos)	49 trazas (79.04% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	3 trazas (14.28% error)	10 trazas (24.4% error)	13 trazas (20.96% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	16 trazas (76.2% aciertos)	36 trazas (87.8% aciertos)	52 trazas (83.87% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	5 trazas (23.8% error)	5 trazas (12.2% error)	10 trazas (16.13% error)

Tabla 8.24. Tercer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	19 trazas (90.48% aciertos)	37 trazas (90.24% aciertos)	56 trazas (90.32% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	2 trazas (9.52% error)	4 trazas (9.76% error)	6 trazas (14.28% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	17 trazas (85.71% aciertos)	29 trazas (70.73% aciertos)	46 trazas (74.2% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	4 trazas (14.28% error)	12 trazas (29.27% error)	16 trazas (25.8% error)

Tabla 8.25. Tercer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (21 trazas benignas y 21 trazas malignas)	Conjunto de test (41 trazas benignas y 41 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	21 trazas (100% aciertos)	41 trazas (100% aciertos)	62 trazas (100% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	0 trazas (0% error)	0 trazas (0% error)	0 trazas (0% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	15 trazas (71.42% aciertos)	34 trazas (82.93% aciertos)	49 trazas (79.03% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	6 trazas (28.58% error)	7 trazas (17.07% error)	13 trazas (20.97% error)

Tabla 8.26. Tercer experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.27, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.28 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	FFSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	49 trazas (79.04% aciertos)	56 trazas (90.32% aciertos)	62 trazas (100% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	13 trazas (20.96% error)	6 trazas (14.28% error)	0 trazas (0% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	52 trazas (83.87% aciertos)	46 trazas (74.2% aciertos)	49 trazas (79.03% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	10 trazas (16.13% error)	16 trazas (25.8% error)	13 trazas (20.97% error)

Tabla 8.27. Experimento 3. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	11.90%	21.95%
FFSM 2	9.52%	23.17%
FFSM 3	2.38%	17.07%
Media	7.9333%	20.7300%
Varianza	4.9544	3.2278

Tabla 8.28. Comparativa de resultados de las tres FFSMs obtenidas en el tercer experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Las FFSMs encontradas en este experimento presentan una eficiencia de clasificación mejor que las encontradas en el segundo experimento. Comparando los resultados del segundo experimento (tabla 8.23) con los de éste (tabla 8.28), se observa que la media del error en el entrenamiento es igual pero la media del error cometido en el test es menor en las máquinas de este experimento, lo que indica una mayor capacidad de generalización en estas últimas.

Las tres máquinas presentan resultados de clasificación aceptables sobre trazas benignas y malignas. Se debe resaltar el comportamiento de la tercera máquina, capaz de clasificar las trazas benignas del conjunto de entrenamiento y del conjunto de test sin error (100% aciertos).

Observando las curvas de entrenamiento y test, se pude apreciar que la curva de test sigue a la de entrenamiento, sin llegar a la situación de sobreajuste, que es el comportamiento deseado en todo proceso de aprendizaje, pero que también los procesos desarrollados presentan un gran estancamiento.

Cuarto experimento.

En la figura 8.30 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.

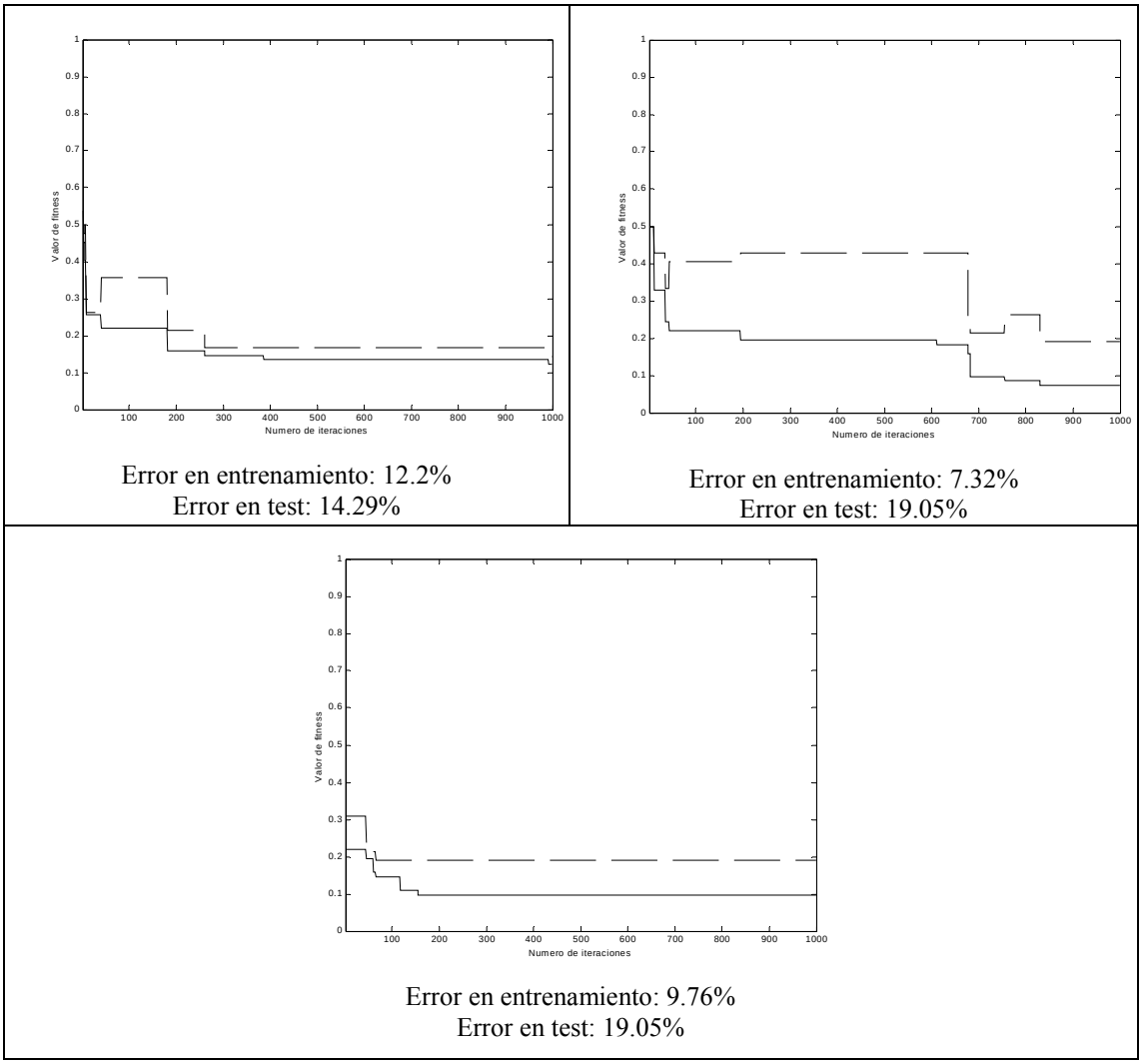


Figura 8.30. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 4 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.29, 8.30 y 8.31.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	36 trazas (87.8% aciertos)	17 trazas (80.95% aciertos)	53 trazas (85.48% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	5 trazas (12.2% error)	4 trazas (19.05% error)	9 trazas (14.52% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	36 trazas (87.8% aciertos)	19 trazas (90.48% aciertos)	55 trazas (88.7% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	5 trazas (12.2% error)	2 trazas (9.52% error)	7 trazas (11.3% error)

Tabla 8.29. Cuarto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	36 trazas (87.8% aciertos)	14 trazas (66.67% aciertos)	50 trazas (80.65% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	5 trazas (12.2% error)	7 trazas (33.33% error)	12 trazas (19.35% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	40 trazas (97.56% aciertos)	21 trazas (100% aciertos)	61 trazas (98.39% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	1 traza (2.44% error)	0 trazas (0% error)	1 trazas (1.61% error)

Tabla 8.30. Cuarto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.32, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.33 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	35 trazas (85.37% aciertos)	15 trazas (71.43% aciertos)	50 trazas (80.65% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	6 trazas (14.63% error)	6 trazas (28.57% error)	12 trazas (19.35% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	39 trazas (95.12% aciertos)	19 trazas (90.48% aciertos)	58 trazas (93.55% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	2 trazas (4.88% error)	2 trazas (9.52% error)	4 trazas (6.45% error)

Tabla 8.31. Cuarto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	FFSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	53 trazas (85.48% aciertos)	50 trazas (80.65% aciertos)	50 trazas (80.65% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	9 trazas (14.52% error)	12 trazas (19.35% error)	12 trazas (19.35% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	55 trazas (88.7% aciertos)	61 trazas (98.39% aciertos)	58 trazas (93.55% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	7 trazas (11.3% error)	1 trazas (1.61% error)	4 trazas (6.45% error)

Tabla 8.32. Experimento 4. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	12.2%	14.29%
FFSM 2	7.32%	19.05%
FFSM 3	9.76%	19.05%
Media	9.7600%	17.4633%
Varianza	2.4400	2.7482

Tabla 8.33. Comparativa de resultados de las tres FFSMs obtenidas en el cuarto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Las FFSMs encontradas en este experimento presentan una eficiencia de clasificación mejor en el test que las encontradas en el experimento anterior, como se

aprecia de comparar los resultados (tabla 8.28 y tabla 8.33), lo que indica una mayor capacidad de generalización en estas últimas.

Las tres máquinas presentan resultados de clasificación aceptables sobre trazas benignas y malignas. Se debe resaltar que las tres máquinas son especialmente sensibles a las trazas malignas, dando para ellas unos valores de clasificación muy buenos (11.3% error para FFSM 1, 1.61% error para FFSM 2 y 6.45% error para FFSM 3, como se puede observar en la tabla 8.32). Esto es muy importante desde el punto de vista del diagnóstico médico, y se verá reflejado en el posterior análisis con curvas ROC. En este sentido, la mejor es la segunda máquina.

Las curvas de entrenamiento y test de la tercera máquina reflejan que ha sido el proceso de aprendizaje con mayor estancamiento de los tres desarrollados en este experimento.

Quinto experimento.

En la figura 8.31 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.

Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.34, 8.35 y 8.36.

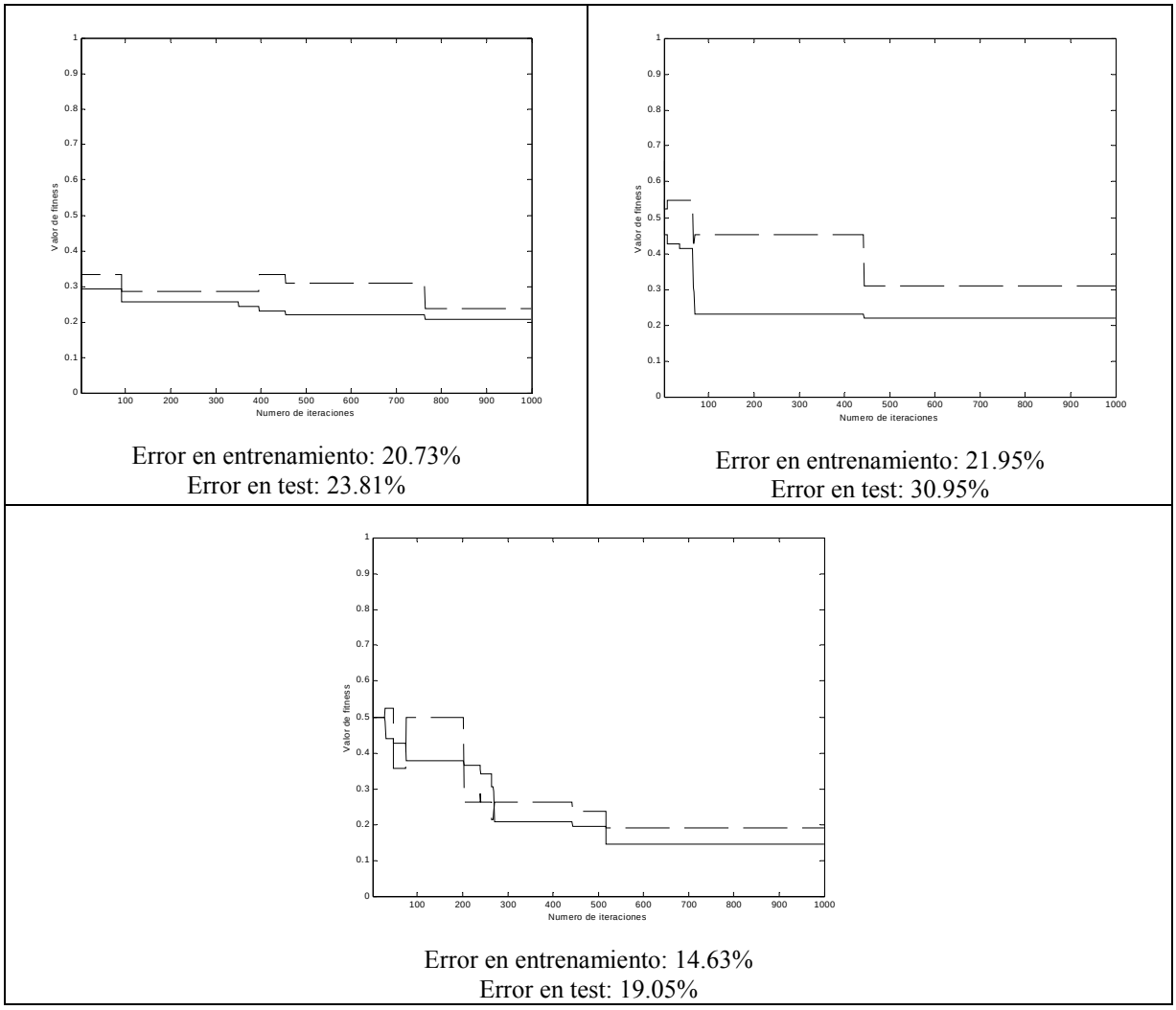


Figura 8.31. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 5 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	40 trazas (97.56% aciertos)	19 trazas (90.48% aciertos)	59 trazas (95.16% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	1 traza (2.44% error)	2 trazas (9.52% error)	3 trazas (4.84% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	25 trazas (60.98% aciertos)	13 trazas (61.9% aciertos)	38 trazas (61.3% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	16 trazas (39.02% error)	8 trazas (38.1% error)	24 trazas (38.7% error)

Tabla 8.34. Quinto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	40 trazas (97.56% aciertos)	17 trazas (80.65% aciertos)	57 trazas (91.94% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	1 traza (2.44% error)	4 trazas (19.05% error)	5 trazas (8.06% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	25 trazas (60.98% aciertos)	6 trazas (28.57% aciertos)	31 trazas (50% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	16 trazas (39.02% error)	15 trazas (71.43% error)	31 trazas (50% error)

Tabla 8.35. Quinto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (41 trazas benignas y 41 trazas malignas)	Conjunto de test (21 trazas benignas y 21 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	36 trazas (87.8% aciertos)	19 trazas (90.48% aciertos)	55 trazas (88.71% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	5 trazas (12.2% error)	2 trazas (9.52% error)	7 trazas (11.29% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	34 trazas (82.93% aciertos)	15 trazas (71.43% aciertos)	49 trazas (79.03% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	7 trazas (17.07% error)	6 trazas (28.57% error)	13 trazas (20.97% error)

Tabla 8.36. Quinto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.37, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.38 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	FFSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	59 trazas (95.16% aciertos)	57 trazas (91.94% aciertos)	55 trazas (88.71% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	3 trazas (4.84% error)	5 trazas (8.06% error)	7 trazas (11.29% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	38 trazas (61.3% aciertos)	31 trazas (50% aciertos)	49 trazas (79.03% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	24 trazas (38.7% error)	31 trazas (50% error)	13 trazas (20.97% error)

Tabla 8.37. Experimento 5. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	20.73%	23.81%
FFSM 2	21.95%	30.95%
FFSM 3	14.63%	19.05%
Media	19.1033%	24.6033%
Varianza	3.9218	5.9895

Tabla 8.38. Comparativa de resultados de las tres FFSMs obtenidas en el quinto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Las FFSMs encontradas en este experimento presentan la peor eficiencia de clasificación sobre trazas malignas de todas las estudiadas hasta este momento en los experimentos expuestos. Esto no es un factor positivo desde el punto de vista del diagnóstico médico.

En cambio, las máquinas encontradas en este experimento son muy eficientes en la clasificación de núcleos benignos (4.84% error para FFSM 1, 8.06% error para FFSM 2 y 11.29% error para FFSM 3, como se puede observar en la tabla 8.37).

A pesar de que la segunda máquina presenta muy buena capacidad de clasificación de núcleos benignos, es indudablemente la peor en la clasificación de núcleos malignos (50% error). Este hecho hace que no sea válida en la ayuda al diagnóstico médico.

En general, los valores medios del error en el entrenamiento y en el test han empeorado en este experimento, como se puede comprobar al comparar los resultados

de la tabla 8.33 y tabla 8.38, a pesar que las curvas de entrenamiento y test de estas máquinas presentan un comportamiento normal, sin grandes estancamientos.

Sexto experimento.

En la figura 8.32 se presentan la curvas de entrenamiento y test para los tres procesos de aprendizaje desarrollados con sistemas Michigan en este experimento. Además, se presentan los valores finales del error en la clasificación del conjunto de entrenamiento y del conjunto de test de la FFSM obtenida en cada proceso.

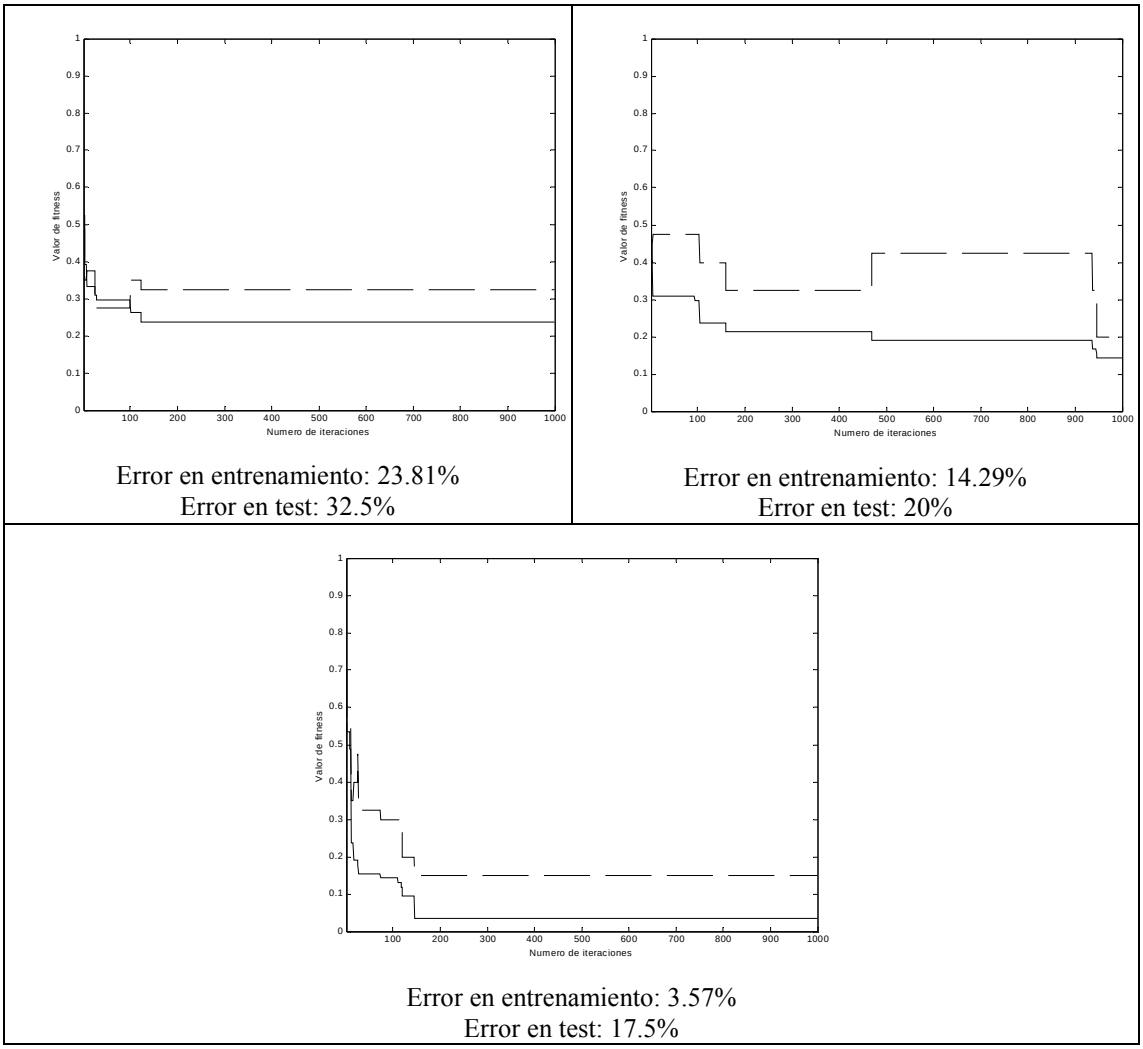


Figura 8.32. Curvas de entrenamiento y test para las tres pruebas con sistemas Michigan realizadas en el experimento 6 en el problema de clasificación de núcleos de imágenes de citologías pleurales.

Estas máquinas, junto con sus correspondientes centros del algoritmo de clustering, forman tres clasificadores finales distintos. La clasificación de las trazas del

conjunto de entrenamiento y del conjunto de test realizada por cada una de las FFSM se muestra en las tablas 8.39, 8.40 y 8.41.

	Conjunto de entrenamiento (42 trazas benignas y 42 trazas malignas)	Conjunto de test (20 trazas benignas y 20 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	29 trazas (69.05% aciertos)	14 trazas (70% aciertos)	43 trazas (69.35% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	13 trazas (30.95% error)	6 trazas (30% error)	19 trazas (30.65% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	35 trazas (83.33% aciertos)	13 trazas (65% aciertos)	48 trazas (77.42% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	7 trazas (16.67% error)	7 trazas (35% error)	14 trazas (22.58% error)

Tabla 8.39. Sexto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del primer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	Conjunto de entrenamiento (42 trazas benignas y 42 trazas malignas)	Conjunto de test (20 trazas benignas y 20 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	31 trazas (73.81% aciertos)	15 trazas (75% aciertos)	46 trazas (74.2% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	11 trazas (26.19% error)	5 trazas (25% error)	16 trazas (25.8% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	41 trazas (97.62% aciertos)	17 trazas (85% aciertos)	58 trazas (93.55% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	1 traza (2.38% error)	3 trazas (15% error)	4 trazas (6.45% error)

Tabla 8.40. Sexto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del segundo sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

A modo de resumen, en la tabla 8.42, se presenta una comparativa del comportamiento de estos tres sistemas clasificadores en la clasificación del conjunto total de trazas, y en la tabla 8.43 se presentan los errores que comete cada clasificador en la clasificación del conjunto de entrenamiento y de test, junto con la media y la varianza de estos resultados.

	Conjunto de entrenamiento (42 trazas benignas y 42 trazas malignas)	Conjunto de test (20 trazas benignas y 20 trazas malignas)	Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	40 trazas (95.24% aciertos)	19 trazas (95% aciertos)	59 trazas (95.16% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	2 trazas (4.72% error)	1 trazas (5% error)	3 trazas (4.84% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	41 trazas (97.62% aciertos)	14 trazas (70% aciertos)	55 trazas (88.71% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	1 trazas (2.38% error)	6 trazas (30% error)	7 trazas (11.29% error)

Tabla 8.41. Sexto experimento. Resultados en la clasificación del conjunto de entrenamiento (primera columna) y en la clasificación del conjunto de test (segunda columna) del tercer sistema clasificador obtenido por el sistema Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales. En la tercera columna se presenta un resumen de resultados.

	FFSM 1 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 2 Conjunto total (62 trazas benignas y 62 trazas malignas)	FFSM 3 Conjunto total (62 trazas benignas y 62 trazas malignas)
Aciertos en la clasificación de trazas correspondientes a núcleos benignos.	43 trazas (69.35% aciertos)	46 trazas (74.2% aciertos)	59 trazas (95.16% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos benignos.	19 trazas (30.65% error)	16 trazas (25.8% error)	3 trazas (4.84% error)
Aciertos en la clasificación de trazas correspondientes a núcleos malignos.	48 trazas (77.42% aciertos)	58 trazas (93.55% aciertos)	55 trazas (88.71% aciertos)
Errores en la clasificación de trazas correspondientes a núcleos malignos.	14 trazas (22.58% error)	4 trazas (6.45% error)	7 trazas (11.29% error)

Tabla 8.42. Experimento 6. Resultados de clasificación de los tres sistemas clasificadores obtenidos con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

	Error en entrenamiento	Error en test
FFSM 1	23.81%	32.5%
FFSM 2	14.29%	20%
FFSM 3	3.57%	17.5%
Media	13.8900%	23.3333%
Varianza	10.1259	8.0364

Tabla 8.43. Comparativa de resultados de las tres FFSMs obtenidas en el sexto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

La primera FFSM encontrada en este experimento es la que presenta una peor eficiencia de clasificación de trazas benignas y malignas. En cambio, la última máquina

encontrada en este experimento es muy eficiente, tanto en la clasificación de núcleos benignos (4.84% error) como malignos (11.29%), tal y como se puede apreciar en la tabla 8.42.

Conviene destacar que la segunda máquina presenta buena capacidad de clasificación de núcleos benignos y una capacidad de clasificación de núcleos malignos aún mejor, lo cual es recomendable en el dominio del problema.

Los valores medios del error en el entrenamiento y en el test han mejorado ligeramente en este experimento, como se puede comprobar al comparar los resultados de la tabla 8.38 y tabla 8.43.

Comparación con otros métodos de clasificación y reconocimiento de patrones.

El primer método de clasificación utilizado fue un método de clustering borroso no supervisado (*fuzzy c-means clustering*). Se aplicó sobre el conjunto total (conjunto de entrenamiento y conjunto de test: 62 trazas benignas y 62 trazas malignas). Los resultados de la clasificación según este algoritmo se recogen en la tabla 8.44, junto con una comparativa con los resultados de las FFSM encontradas con los sistema Michigan en cada uno de los seis experimentos. En la tabla 8.44 se muestra el error total, es decir, el número total de fallos (fallos en trazas benignas + fallos en trazas malignas) dividido entre el número total de trazas (62 trazas benignas + 62 trazas malignas).

Los resultados de la clasificación realizada por el algoritmo de clustering borroso, al igual que en las pruebas presentadas en la sección anterior relativas a citologías peritoneales, sirven para obtener una impresión previa de la dificultad de la clasificación sobre el espacio de vectores de características definido por las trazas extraídas. Es evidente que estos resultados aunque indicativos de la existencia de dos regiones diferenciadas no son buenos, y hasta las peores máquinas encontradas en los experimentos de Michigan presentan mejores resultados globales de clasificación.

Por todo esto, aplicamos otras técnicas mejores de clasificación para realizar esta comparativa. En este caso, utilizaremos dos redes neuronales con propagación hacia delante, la primera (red neuronal 1) con tres neuronas en la capa intermedia, y la segunda (red neuronal 2) con cuatro neuronas en la capa intermedia.

		Error total de clasificación		
Clustering borroso (Fuzzy k-means clustering)		**37.0900%**		
		Error total de clasificación	Media	Desviación
Experimento Michigan 1	FFSM 1	23.38%	**18.5433%**	10.5428
	FFSM 2	6.45%		
	FFSM 3	25.8%		
Experimento Michigan 2	FFSM 1	15.32%	**18.8133%**	5.3727
	FFSM 2	16.12%		
	FFSM 3	25%		
Experimento Michigan 3	FFSM 1	18.54%	**15.5867%**	4.4406
	FFSM 2	17.74%		
	FFSM 3	10.48%		
Experimento Michigan 4	FFSM 1	12.9%	**12.0933%**	1.3972
	FFSM 2	10.48%		
	FFSM 3	12.9%		
Experimento Michigan 5	FFSM 1	21.77%	**22.5733%**	6.8902
	FFSM 2	29.83%		
	FFSM 3	16.12%		
Experimento Michigan 6	FFSM 1	26.61%	**16.9300%**	9.3015
	FFSM 2	16.12%		
	FFSM 3	8.06%		

Tabla 8.44: Resultados en la clasificación del conjunto total de trazas por el algoritmo de clustering borroso y por los sistemas clasificadores con FFSM bajo estudio obtenidos en los seis experimentos de Michigan para el problema de la clasificación de núcleos en imágenes de citologías pleurales.

Se realizan seis entrenamientos distintos con las redes neuronales, cada uno de ellos con el mismo conjunto de entrenamiento empleado en los entrenamientos de los sistemas Michigan en los experimentos anteriormente presentados. El test se realizó con los correspondientes conjuntos de test utilizados en los citados experimentos.

A modo de resumen, en la tabla 8.45 se muestran los resultados de clasificación del conjunto de entrenamiento y del conjunto de test de las distintas redes neuronales probadas. Además, se presenta la comparativa entre estos resultados y los obtenidos con las FFSMs consideradas en cada uno de los seis experimentos.

Experimento 1						
	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FFSM 1	15%	**9.1667%**	8.0364	27.38%	**24.6033%**	9.2433
FFSM 2	0%			14.29%		
FFSM 3	12.5%			32.14%		
Red neuronal 1	**0%**			**27.38%**		
Red neuronal 2	**0%**			**32.14%**		
Experimento 2						
Experimento 2	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FFSM 1	4.76%	**7.9333%**	3.6355	21.95%	**28.4533%**	8.2987
FFSM 2	7.14%			25.61%		
FFSM 3	11.9%			37.80%		
Red neuronal 1	**0%**			**23.80%**		
Red neuronal 2	**0%**			**35.37%**		
Experimento 3						
	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FSM 1	11.90%	**7.9333%**	4.9544	21.95%	**20.7300%**	3.2278
FSM 2	9.52%			23.17%		
FSM 3	2.38%			17.07%		
Red neuronal 1	**14.28%**			**34.14%**		
Red neuronal 2	**23.81%**			**52.44%**		
Experimento 4						
	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FFSM 1	12.2%	**9.7600%**	2.4400	14.29%	**17.4633%**	2.7482
FFSM 2	7.32%			19.05%		
FFSM 3	9.76%			19.05%		
Red neuronal 1	**2.43%**			**23.80%**		
Red neuronal 2	**0%**			**16.67%**		
Experimento 5						
	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FFSM 1	20.73%	**19.1033%**	3.9218	23.81%	**24.6033%**	5.9895
FFSM 2	21.95%			30.95%		
FFSM 3	14.63%			19.05%		
Red neuronal 1	**0%**			**7.14%**		
Red neuronal 2	**0%**			**14.29%**		
Experimento 6						
	Error en entrenamiento	Media	Varianza	Error en test	Media	Varianza
FFSM 1	23.81%	**13.8900%**	10.1259	32.5%	**23.3333%**	8.0364
FFSM 2	14.29%			20%		
FFSM 3	3.57%			17.5%		
Red neuronal 1	**0%**			**25%**		
Red neuronal 2	**0%**			**25%**		

Tabla 8.45. Resultados de las redes neuronales en el entrenamiento y en el test bajo las mismas condiciones de los seis experimentos realizados con sistemas Michigan. Comparación con los resultados en el entrenamiento y en el test de las FFSMs obtenidas con los sistemas Michigan.

A la vista de estos resultados, comparando el comportamiento de las dos redes neuronales, se puede apreciar que la red neuronal 2 (constituida por 4 neuronas en la capa intermedia) no mejora los resultados de clasificación en el test obtenidos con la red neuronal 1 (constituida por 3 neuronas en la capa intermedia), salvo en el experimento número cuatro. Es decir, la red neuronal con menos neuronas en la capa intermedia tiene una mayor capacidad de generalización.

En cada experimento con sistemas Michigan se han obtenido tres clasificadores basados en sistemas borrosos. Se puede observar en la tabla 8.45 que en cinco de los seis experimentos se ha podido encontrar alguna FFSM con mejor eficiencia de clasificación que las redes neuronales. Estos resultados están destacados en la tabla.

De especial interés son los resultados correspondientes al tercer experimento, donde se puede apreciar que las tres FFSMs obtenidas por sistemas Michigan son mejores en la clasificación de los núcleos que cualquiera de las dos redes probadas. En cambio, en el quinto experimento no se encontró ninguna FFSM capaz de superar en eficiencia a las redes neuronales.

Si comparamos las medias del error en el test de las FFSMs de cada experimento con el error en el test de las correspondientes redes neuronales, se puede apreciar que en los experimentos 1, 3 y 6 estas medias son menores que los errores de las dos redes y que en los experimentos 2 y 4 son menores que los de la segunda red.

Para descartar que los resultados de las redes estén influenciados por algún fenómeno de sobre-ajuste en su entrenamiento, y que esto favorezca la bondad de los resultados de las FFSMs en esta comparativa, realizamos un experimento adicional con la primera red (constituida por tres neuronas en la primera capa), por ser ésta la que presenta el mejor comportamiento de las dos probadas.

En este experimento, se vuelve a entrenar la red seis veces, bajo las mismas condiciones de cada experimento de Michigan (mismos conjuntos de entrenamiento y test), pero esta vez fijando un umbral de error en el entrenamiento igual a 11%. Este umbral se escoge así porque en media, el error de entrenamiento de las FFSMs de todos los experimentos es 11.2977%. Se intenta obtener una red que presente el mismo error en el entrenamiento para ver cómo afecta este factor a su capacidad de generalización en el test. En la tabla 8.46 se pueden observar los resultados de estos entrenamientos adicionales con la red neuronal 1 en comparación con los valores obtenidos en los entrenamiento anteriores.

	Resultado anterior Error en entrenamiento	Nuevo resultado Error en entrenamiento	Resultado anterior Error en test	Nuevo resultado Error en test
Experimento 1	0%	5%	27.38%	33.33%
Experimento 2	0%	14.28%	23.80%	34.14%
Experimento 3	14.28%	14.28%	34.14%	42.68%
Experimento 4	2.43%	10.97%	23.80%	21.14%
Experimento 5	0%	9.75%	7.14%	23.80%
Experimento 6	0%	9.52%	25%	32.5%

Tabla 8.46. Comparativa entre los resultados de clasificación en entrenamiento y test anteriores (columnas 1 y 3) de la red neuronal 1 en cada experimento y los nuevos resultados (columnas 2 y 4) limitando la eficiencia en su entrenamiento.

Se puede descartar que se produjera un fenómeno de sobre-ajuste en las pruebas anteriores, ya que la capacidad de generalización en estas últimas pruebas es notablemente peor en todos los experimentos.

Por último, se ha realizado una clasificación a partir de la identificación del modelo oculto de Markov. La hipótesis realizada es que las series de datos reales obtenidas son el resultado de un proceso Markoviano. La clasificación se hace a partir de un conjunto de entrenamiento, de forma supervisada. En este proceso se identifica el modelo oculto de Markov utilizando el algoritmo de Baum-Welch. Una vez obtenido un modelo para cada clase de serie de datos (núcleos benignos y malignos) en el proceso de entrenamiento, se podrán asignar las series de datos a las clases a partir del cálculo de la probabilidad condicionada al modelo $P(O|\lambda)$. En la práctica existe la dificultad del desconocimiento de las probabilidades asociadas a las clases $P(\lambda_1)$ y $P(\lambda_2)$, con lo que a priori no tendemos a favorecer ninguno de los modelos. En este experimento se ha diseñado el conjunto de test de forma que $P(\lambda_1)=P(\lambda_2)$ (asignando el mismo número de patrones a las dos clases), lo que mantiene las proporciones usadas en el entrenamiento.

Este procedimiento se realiza seis veces, utilizando los mismos conjuntos de entrenamiento y test que en los seis experimentos desarrollados con sistemas Michigan anteriormente expuestos, con el fin de realizar una comparativa entre los resultados. En la tabla 8.47 se muestran los resultados de clasificación de entrenamiento y de test obtenidos al aplicar el algoritmo de Baum-Welch como se ha explicado previamente.

	Error en entrenamiento	Error en test
Experimento 1	30%	37.5%
Experimento 2	25%	30%
Experimento 3	30%	28.47%
Experimento 4	30%	20%
Experimento 5	26.25%	37.5%
Experimento 6	26.25%	37.5%

Tabla 8.47. Resultados de clasificación a partir de la identificación de modelos ocultos de Markov para cada clase bajo las mismas condiciones de los seis experimentos realizados con sistemas Michigan en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Tanto en el entrenamiento como en el test, se observa que los resultados no han sido buenos, lo que puede indicar que la hipótesis realizada sobre el proceso que genera los datos es incorrecta, o también, que la longitud de las series de datos es insuficiente para distinguir los modelos a partir del procedimiento de identificación-evaluación de $P(O|\lambda)$.

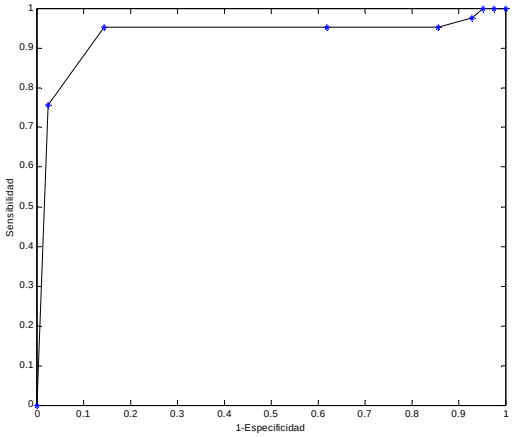
Evaluación con curvas ROC.

Es importante evaluar estos clasificadores desde el punto de vista de ayuda al diagnóstico. Para ello, utilizamos de nuevo el análisis ROC, técnica de evaluación de clasificadores en medicina comentada en detalle en el capítulo 3.

Evaluaremos con esta metodología la mejor máquina obtenida con sistemas Michigan de cada uno de los seis experimentos elegidos, y la mejor red neuronal en cada caso.

En los clasificadores obtenidos con sistemas tipo Michigan basados en maquinas de estado borrosas, el valor de corte es el umbral que debe superar el nivel de activación del estado de detección para considerar que este estado está activado a alta (parámetro *param_alta*). Haciendo un barrido de este parámetro y analizando los resultados de la clasificación sobre el conjunto de test (aciertos y errores) se calculan los pares (sensibilidad, 1-especificidad) y las curvas ROC representadas en las figuras 8.33, 8.34, 8.35, 8.36, 8.37 y 8.38 para cada una de las máquinas seleccionadas.

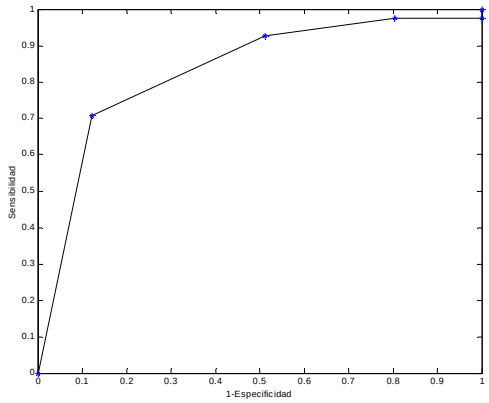
Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	0.9762	1
0.5	0.9524	1
0.52	0.9286	0.9762
0.55	0.8571	0.9524
0.58	0.6190	0.9524
0.6	**0.1429**	**0.9524**
0.62	0.0238	0.7561
0.65	0	0
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 1: FFSM 2
(0% error en entrenamiento, 14.29% error en test)

Figura 8.33. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el primer experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

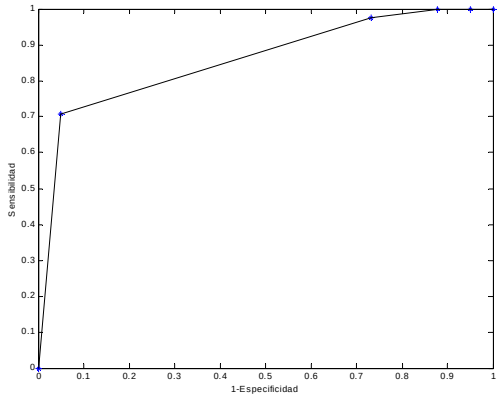
Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	1	1
0.5	1	1
0.55	1	0.9756
0.58	0.8049	0.9756
0.59	0.5122	0.9268
0.6	**0.1220**	**0.7073**
0.62	0	0
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 2: FFSM 1
(4.76% error en entrenamiento, 21.95% error en test)

Figura 8.34. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el segundo experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

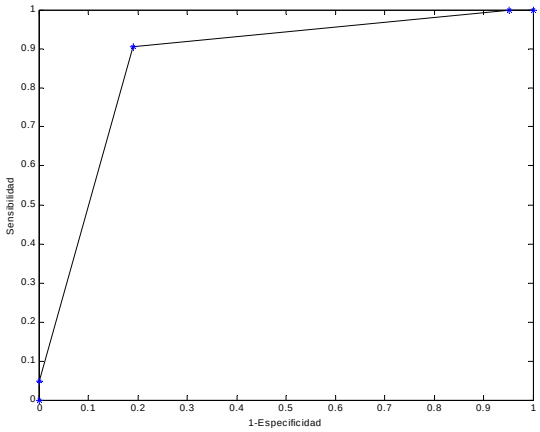
Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	1	1
0.5	1	1
0.54	0.9512	1
0.56	0.8780	1
0.58	0.7317	0.9756
0.6	**0.0488**	**0.7073**
0.61	0	0
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 3: FFSM 3
(2.38% error en entrenamiento, 17.07% error en test)

Figura 8.35. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el tercer experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

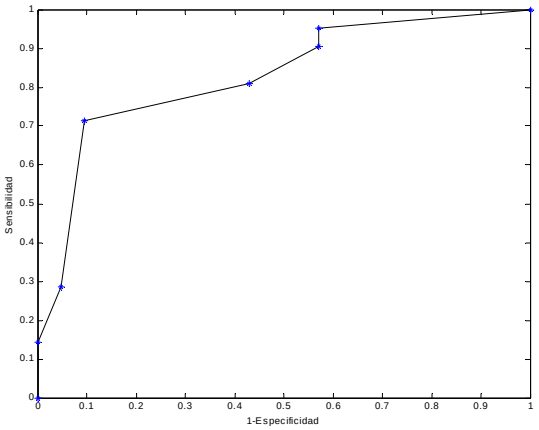
Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	1	1
0.5	1	1
0.58	0.9524	1
0.6	**0.1905**	**0.9048**
0.61	0	0.0476
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 4: FFSM 1
(12.2% error en entrenamiento, 14.29% error en test)

Figura 8.36. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el cuarto experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

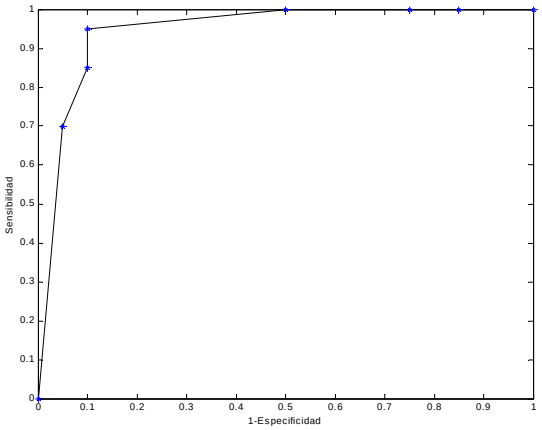
Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	1	1
0.5	1	1
0.57	0.5714	0.9524
0.58	0.5714	0.9048
0.59	0.4286	0.8095
0.6	**0.0952**	**0.7143**
0.61	0.0476	0.2857
0.62	0	0.1429
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 5: FFSM 3
(14.63% error en entrenamiento, 19.05% error en test)

Figura 8.37. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el quinto experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

Valor de corte	1-Especificidad	Sensibilidad
0	1	1
0.1	1	1
0.2	1	1
0.3	1	1
0.4	1	1
0.5	0.85	1
0.52	0.75	1
0.55	0.5	1
0.58	0.1	0.95
0.59	0.1	0.85
0.6	**0.05**	**0.7**
0.61	0	0
0.7	0	0
0.8	0	0
0.9	0	0
1	0	0



Mejor máquina del experimento 6: FFSM 3
(3.57% error en entrenamiento, 17.5% error en test)

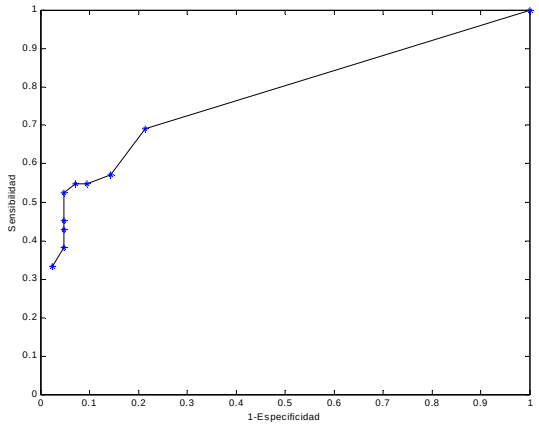
Figura 8.38. Curva ROC del mejor clasificador basado en la máquina de estados borrosa encontrado con un sistema Michigan en el sexto experimento para el problema de clasificación de núcleos en imágenes de citologías pleurales.

Los mejores clasificadores se obtienen con un valor de corte igual a 0.6, ya que con ese valor se obtienen los mayores valores de sensibilidad (porcentaje de aciertos en la clasificación de núcleos malignos) con la mayor especificidad posible (mayor porcentaje de aciertos en núcleos benignos). En las tablas se ha destacado este dato por ser el valor de corte óptimo. Este valor de corte coincide con el valor en que hemos

fijado el parámetro *param_alta*. Esta coincidencia indica la adecuación de la medida de aptitud utilizada en el proceso evolutivo.

Repitiendo estos mismos cálculos para los mejores clasificadores basados en redes neuronales bajo las mismas condiciones de cada experimento (mismo conjunto de entrenamiento y de test) se obtienen las curvas ROC de las figuras 8.39, 8.40, 8.41, 8.42, 8.43 y 8.44.

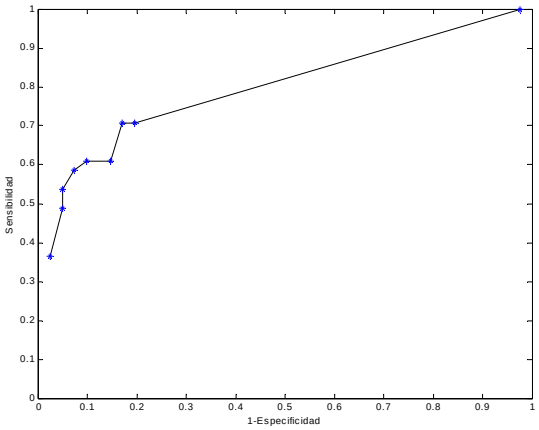
Valor de corte	1-Especificidad	Sensibilidad
0	0.0238	0.3333
0.1	0.0476	0.3810
0.2	0.0476	0.3810
0.3	0.0476	0.4286
0.4	0.0476	0.4524
0.5	0.0476	0.5238
0.6	0.0714	0.5476
0.7	0.0952	0.5476
0.8	0.1429	0.5714
0.9	0.2143	0.6905
1	1	1



Mejor red neuronal: 1 (tres neuronas / 1 neurona)
(0% error en entrenamiento, 27.38% error en test)

Figura 8.39. Curva ROC de la mejor red neuronal para las condiciones del primer experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

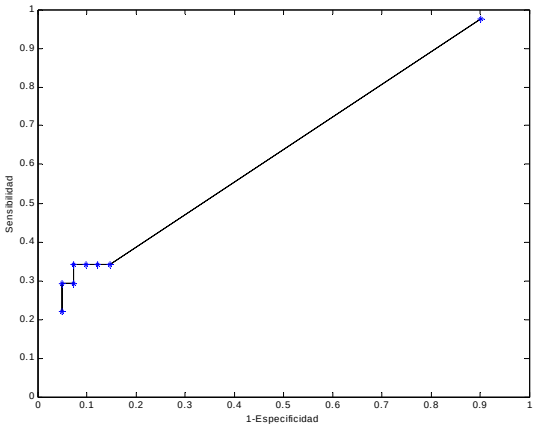
Valor de corte	1-Especificidad	Sensibilidad
0	0.0244	0.3659
0.1	0.0488	0.4878
0.2	0.0488	0.5366
0.3	0.0732	0.5854
0.4	0.0976	0.6098
0.5	0.1463	0.6098
0.6	0.1463	0.6098
0.7	0.1463	0.6098
0.8	0.1463	0.6098
0.9	0.1463	0.6098
0.95	0.1707	0.7073
0.98	0.1951	0.7073
1	0.9756	1



Mejor red neuronal: 1 (tres neuronas / 1 neurona)
(0% error en entrenamiento, 23.80% error en test)

Figura 8.40. Curva ROC de la mejor red neuronal para las condiciones del segundo experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

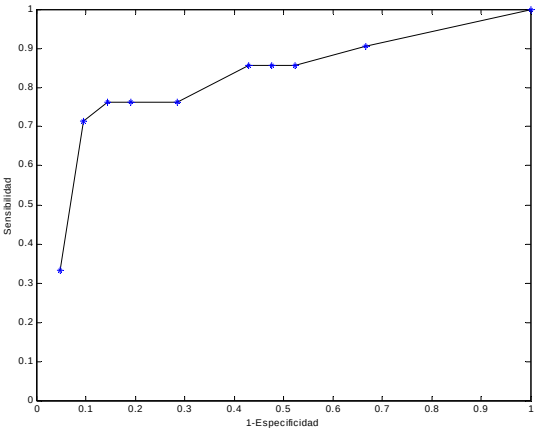
Valor de corte	1-Especificidad	Sensibilidad
0	0.0488	0.2195
0.1	0.0488	0.2927
0.2	0.0488	0.2927
0.3	0.0488	0.2927
0.4	0.0488	0.2927
0.5	0.0732	0.2927
0.6	0.0732	0.2927
0.7	0.0732	0.2927
0.8	0.0732	0.3415
0.9	0.0976	0.3415
0.96	0.1220	0.3415
0.98	0.1463	0.3415
1	0.9024	0.9756



Mejor red neuronal: 1 (tres neuronas / 1 neurona)
(14.28% error en entrenamiento, 34.14% error en test)

Figura 8.41. Curva ROC de la mejor red neuronal para las condiciones del tercer experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

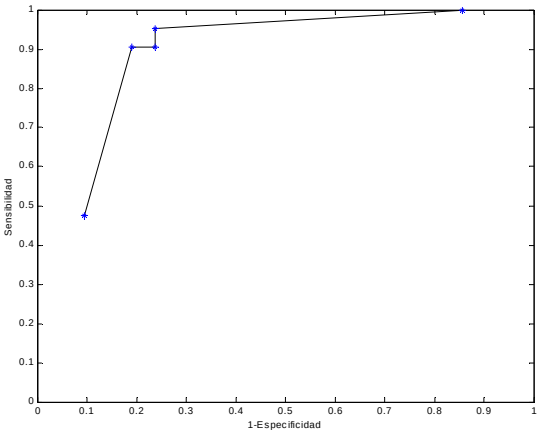
Valor de corte	1-Especificidad	Sensibilidad
0	0.0476	0.3333
0.1	0.0952	0.7143
0.2	0.1429	0.7619
0.3	0.1905	0.7619
0.4	0.2857	0.7619
0.5	0.4286	0.8571
0.6	0.4762	0.8571
0.7	0.4762	0.8571
0.8	0.5238	0.8571
0.9	0.6667	0.9048
1	1	1



Mejor red neuronal: 2 (cuatro neuronas / 1 neurona)
(0% error en entrenamiento, 16.67% error en test)

Figura 8.42. Curva ROC de la mejor red neuronal para las condiciones del cuarto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

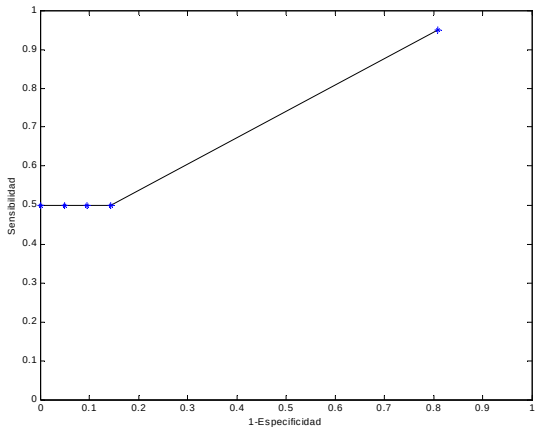
Valor de corte	1-Especificidad	Sensibilidad
0	0.0952	0.4762
0.1	0.1905	0.9048
0.2	0.1905	0.9048
0.3	0.1905	0.9048
0.4	0.1905	0.9048
0.5	0.1905	0.9048
0.6	0.2381	0.9048
0.7	0.2381	0.9048
0.8	0.2381	0.9048
0.9	0.2381	0.9048
0.98	0.2381	0.9524
1	0.8571	1



Mejor red neuronal: 1 (tres neuronas / 1 neurona)
(0% error en entrenamiento, 7.14% error en test)

Figura 8.43. Curva ROC de la mejor red neuronal para las condiciones del quinto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Valor de corte	1-Especificidad	Sensibilidad
0	0	0.5
0.1	0	0.5
0.2	0	0.5
0.3	0	0.5
0.4	0	0.5
0.5	0.05	0.5
0.6	0.05	0.5
0.7	0.05	0.5
0.8	0.05	0.5
0.9	0.05	0.5
0.94	0.0952	0.5
0.95	0.1429	0.5
1	0.8095	0.95



Mejor red neuronal: 1 (tres neuronas / 1 neurona)
(0% error en entrenamiento, 7.14% error en test)

Figura 8.44. Curva ROC de la mejor red neuronal para las condiciones del sexto experimento en el problema de clasificación de núcleos en imágenes de citologías pleurales.

Se puede observar que los valores de corte óptimos para las redes neuronales varían de una curva a otra. Las redes clasifican mejor el conjunto de test cuando sus valores de corte están establecidos en los valores destacados en las correspondientes tablas. En estos puntos es donde la sensibilidad y la especificidad son las mayores posibles.

Como se mencionó en el capítulo 3, la exactitud de la prueba aumenta a medida que la curva se desplaza desde la diagonal hacia el vértice superior izquierdo. Un

clasificador ideal (100% de sensibilidad y 100% de especificidad) pasaría por dicho vértice. Un modo de comparar los clasificadores, es comparar sus curvas ROC.

A simple vista se aprecia que, igual que en el caso estudiado de la clasificación de núcleos en imágenes de citologías de fluidos peritoneales, las curvas correspondientes a los clasificadores basados en máquinas de estados borrosas son las que más se aproximan a este vértice, por lo tanto, desde el punto de vista médico, serían los método con mejores resultados de clasificación y los que más podría ayudar al diagnóstico.

Destacan las curvas correspondientes a las FFSMs de las figuras 8.33 y 8.38, por ser curvas ROC que reflejan un comportamiento especialmente bueno del clasificador. En cambio, en la figura 8.41, se muestra la peor curva ROC de las presentadas, correspondiente a una red neuronal.

8.3.3.3 Conclusiones.

Estos experimentos se han realizado con el objetivo de comprobar la validez de la aplicación de los sistemas Michigan en el problema de clasificación de series de datos reales. Mediante sistemas tipo Michigan se obtienen FFSMs capaces de clasificar núcleos sanos y patológicos en imágenes de citologías. Para realizar estas pruebas, en esta ocasión, se han utilizado imágenes de citologías pleurales, previamente catalogadas por un experto del dominio.

Se han presentado seis experimentos, con distintos conjuntos de entrenamiento y test en cada uno, construidos con las trazas disponibles en este problema. En cada experimento se han buscado tres FFSMs distintas con sistemas Michigan, para estudiar la tendencia de estos clasificadores de un modo más global.

Los resultados obtenidos confirman de nuevo la capacidad de la FFSM de clasificar series de datos reales. Se han encontrado máquinas con resultados aceptables de clasificación, e incluso, algunas con mejores resultados que las encontradas por sistemas Pittsburgh en los experimentos relacionados con imágenes de citologías peritoneales presentados en el apartado anterior. En los experimentos con sistemas Pittsburgh se había encontrado una máquina con un error de 3.33% en el entrenamiento y con un error de 21.51% en el test. En los seis experimentos con sistemas Michigan se han encontrado máquinas que mejoran esta eficiencia, o al menos, la mantienen. La

principal ventaja que presentan los sistemas Michigan frente a los sistemas Pittsburgh es que los procesos de entrenamiento son mucho más rápidos. Sin embargo, hay que recordar que en las pruebas citadas se usan conjuntos de entrenamiento diferentes.

De nuevo, estos buenos resultados ponen de manifiesto que la textura de los núcleos es muy buen indicador de la naturaleza benigna o maligna de los mismos y confirman que el modo en que extraemos esta información es válida. Como ya se ha mencionado antes, existen otras características que en trabajos futuros se considerarán, para completar la información y poder realizar una clasificación más eficiente. Dentro de estas características se podrían citar las siguientes: el tamaño y forma de las células, relación núcleo-citoplasma, irregularidades de la frontera, etc.

Tras estudiar los resultados de clasificación de las FFSMs sobre las series de datos reales y compararlos con los resultados de clasificación de los otros métodos probados (clustering borroso y redes neuronales), se observa que, en la mayoría de los experimentos realizados, se ha encontrado una FFSM que presenta una mayor capacidad de generalización que la presentada por estos métodos.

En los experimentos desarrollados con sistemas Pittsburgh, sobre las imágenes de citologías peritoneales, y con sistemas Michigan, sobre las imágenes de citologías pleurales, las FFSMs buscadas están constituidas por 10 reglas. Esto se ha elegido así tras desarrollar una serie de experimentos previos en los que se buscaban máquinas con distintos tipos de reglas. Las FFSMs que presentaban mejores eficiencias de clasificación eran las constituidas por 10 reglas. Recordemos que, en las conclusiones de los experimentos anteriores, mencionábamos la observación de que un número insuficiente de reglas provoca un "estancamiento" en el proceso evolutivo y lo relacionábamos con la carencia de intrones que protejan a las reglas importantes de su destrucción en el proceso evolutivo.

Un análisis aparte merece también el hecho de haber tomado como constante el ancho de las funciones de pertenencia quedando fuera este factor del conjunto de parámetros de diseño. Desde el punto de vista de los sistemas borrosos que constituyen la máquina esto supone el cubrimiento regular del espacio de entradas con parches gaussianos del mismo tamaño aproximadamente. Esto simplifica el espacio de búsqueda del algoritmo evolutivo, pero conlleva el problema de la elección adecuada del ancho prefijado para las funciones de pertenencia. Si este ancho es excesivamente grande se pierde precisión y si es excesivamente pequeño se incrementa notablemente el número

de reglas necesarias para un recubrimiento correcto, lo que además va en detrimento del objetivo inicial de simplificar el espacio de búsqueda y hace mucho menos comprensible la base de reglas. Nuestro propósito en futuras investigaciones es combinar el método de diseño presentado con la técnica de simplificación de la base de reglas por recombinación de antecedentes [Yam et al., 1999], idea que se basa en producir máquinas con particiones en el espacio de entrada regulares y suficientemente finas para luego simplificarlas con la técnica mencionada que equivale a una reconfiguración de la partición.

En los estudios realizados de los distintos clasificadores con curvas ROC sobre los conjuntos de test, es importante destacar que, en el caso de las FFSMs, los valores de corte óptimos coinciden con el valor del parámetro correspondiente al umbral que debe superar el estado de detección para considerar su activación a alta (parámetro *param_alta*). Esto indica que para el conjunto de test el mejor valor para el umbral es precisamente el establecido en el entrenamiento. Al menos se puede afirmar que una parte del modelo subyacente a los datos, y que concierne al papel que juega el umbral de activación del estado de detección en el procesamiento de las trazas, ha sido aprendido por las máquinas, ya que sigue obteniéndose la mejor clasificación desde el punto de vista de las curvas ROC para este valor del parámetro.

En contraste, en las redes neuronales, estos valores óptimos de corte cambian de una prueba a otra. Esto no es lo más conveniente ya que indica que el umbral de discriminación utilizado en el entrenamiento (0.5) no es el mejor para el test. Esta situación se debe a que la red neuronal no está aprendiendo en su totalidad el modelo común a los datos de entrenamiento y test, sino que aprende características particulares de los datos de entrenamiento.

Analizando las curvas de los clasificadores evaluados, se aprecia que la comparativa de curvas ROC es claramente ventajosa para las máquinas de estados borrosas frente a las redes neuronales. Las mejores máquinas de cada experimento presentan una tendencia muy pronunciada a clasificar correctamente los núcleos malignos. Esta faceta en un calsificador es un aspecto muy importante en los sistemas de ayuda al diagnóstico médico.

Por último, es importante resaltar que los algoritmos propuestos en esta tesis dependen de factores experimentales que se deben controlar, como por ejemplo, el enfoque de las imágenes o la resolución de las mismas.

Conclusiones, aportaciones y líneas abiertas.

Conclusiones.

En este trabajo de investigación se ha estudiado un sistema borroso recurrente, denominado máquina de estados borrosa, como parte fundamental de un sistema de clasificación de series de datos.

El objetivo principal de este estudio fue buscar una vía de solución para uno de los principales problemas de las máquinas de estados borrosas: el diseño automático de las mismas. Para ello, se exploró en el campo de los algoritmos genéticos, utilizándose los enfoques de Pittsburgh y Michigan.

Tras elaborar dos algoritmos adecuados para diseñar la máquina de estados borrosa bajo estas estrategias, se pasó a realizar un estudio de validación, utilizando un modelo de referencia como es el modelo oculto de Markov. Además, se realizó un estudio de aplicabilidad sobre datos reales, en el campo de la clasificación de núcleos celulares en citologías.

La principal conclusión de este trabajo de investigación es la viabilidad de los sistemas basados en algoritmos genéticos como herramientas de diseño para la construcción de clasificadores basados en el concepto de máquinas de estado borrosas.

Esto se comprobó tanto en el estudio de validación como en el de aplicabilidad. El método propuesto se basa en utilizar el grado de reactividad de la máquina de estados borrosa para obtener medidas globales sobre las características de las series de datos analizadas y así integrar dicho algoritmo en un clasificador. Esta medida global permitió clasificar series temporales de longitudes medias obtenidas a partir de modelos ocultos de Markov, con eficiencia similar a un algoritmo basado en la identificación del HMM y el cálculo de la probabilidad condicionada al modelo. De esta forma se comprobó que los sistemas propuestos pueden clasificar series temporales provenientes de modelos markovianos.

El trabajo con datos reales se desarrolló sobre series de datos obtenidas de imágenes de citologías médicas. Las series de datos pretenden describir el aspecto de la distribución de cromatina en el núcleo celular.

Los objetivos para utilizar esta técnica de clasificación sobre estas series fueron dos: por una parte, comprobar si el método utilizado para extraer esta medida de la textura de la cromatina nuclear, que es novedoso, sencillo y rápido, genera parámetros útiles con información acerca de la naturaleza benigna o maligna del núcleo. Por otra, aplicar los clasificadores basados en máquinas de estados borrosas diseñados mediante sistemas Pittsburgh y Michigan sobre este problema, para estudiar la validez de la metodología en la clasificación de series de datos reales.

Se aplicó la técnica de clasificación propuesta para diferentes pruebas médicas observándose que efectivamente las series de datos extraídas de los núcleos celulares contienen información sobre la naturaleza del núcleo.

Un análisis más detallado fue realizado en el caso de las citologías pleurales, donde se estudió de forma comparativa la eficiencia del sistema Michigan, una clasificación no supervisada, redes neuronales con propagación hacia adelante y el método basado en la identificación del modelo oculto de Markov junto con la evaluación de la probabilidad de la observación condicionada al modelo. Los resultados de la clasificación no supervisada, que se realizó con el algoritmo *fuzzy c-means clustering,* sirvieron para obtener una medida relativa de la dificultad del problema de clasificación. Tanto este método como el basado en la hipótesis del modelo oculto de Markov subyacente, fueron bastante peores en las pruebas realizadas que las redes neuronales y el sistema Michigan.

En las pruebas realizadas con redes neuronales, si bien el error de entrenamiento era disminuido considerablemente, se observaron problemas a la hora de la generalización. La aplicación de una técnica de parada temprana para evitar el sobre-ajuste a los datos de entrenamiento tampoco mejoró el rendimiento de las mismas en los conjuntos de test. Sin embargo, en las pruebas con el sistema Michigan, a pesar de producirse errores en el entrenamiento superiores a los de las redes neuronales, se obtenían errores menores en el conjunto de validación.

Además, se presentó una evaluación de los clasificadores borrosos diseñados y de las redes neuronales con curvas ROC, análisis que estudia la eficiencia de un clasificador desde el punto de vista del diagnóstico médico. Esto es importante, ya que nos permite conocer la viabilidad de estos sistemas como clasificadores en problemas reales de ayuda al diagnóstico. Este análisis fue bastante favorable para los clasificadores basados en la máquina de estados borrosa.

Una conclusión positiva, que se desprende de la experimentación realizada, es que las máquinas obtenidas por el sistema Michigan alcanzan una eficiencia en cuanto a la clasificación similar a las máquinas obtenidas por el sistema Pittsburgh, tanto en el estudio de simulación como en el estudio con datos reales. Recordemos que el sistema Michigan ofrece una mejor eficiencia computacional, obteniendo estos resultados con un menor tiempo de cálculo.

Estos resultados positivos, tanto en simulación como con datos reales, aunque demuestran la capacidad de los sistemas propuestos para clasificar series de datos, no nos deben hacer olvidar las principales dificultades encontradas con la técnicas de diseño basadas en los sistemas Pittsburgh y Michigan, y que son comunes a cualquier sistema evolutivo. La parametrización de los algoritmos es una de las principales dificultades: encontrar valores óptimos para todos los parámetros, especialmente en el caso del sistema Michigan, no es una tarea sencilla y requiere de bastante experimentación. Por otra parte, nos encontramos también con la naturaleza estocástica del proceso de entrenamiento, que nos lleva a diferentes resultados para diferentes entrenamientos. Este problema dificulta la investigación sobre los propios algoritmos en gran medida, ya que son necesarias un gran número de pruebas para asegurar la calidad de los parámetros o llegar a una conclusión acerca del comportamiento del sistema.

Finalmente, tenemos el problema de la parada de los algoritmos, es decir en qué momento se decide detener el proceso evolutivo y escoger la solución con el mejor

valor de la función objetivo. Dado el desconocimiento sobre el valor característico de la función objetivo en el mínimo global, se deben establecer umbrales de eficiencia mínima, de forma que si la solución alcanza este umbral durante el entrenamiento el proceso iterativo del algoritmo es detenido.

Aportaciones.

Esta investigación se ha centrado en dos problemas asociados con el algoritmo denominado máquina finita de estados borrosa. Estos dos problemas son la aplicabilidad del algoritmo en el campo del reconocimiento de patrones y el diseño automático del sistema recurrente borroso.

- En cuanto al primer problema, se ha diseñado un clasificador basado en la FFSM para la clasificación de series temporales. La principal idea del método es usar una medida de la reactividad de un estado de la FFSM (cambios en el nivel de activación) como característica a usar en la clasificación. Se trata de una medida global de la serie temporal que alimenta a la FFSM, es decir no se detectan patrones puntuales en los datos, sino comportamientos generales capaces de inducir una reactividad específica en el sistema borroso recurrente. Ahora bien, el principal problema sigue siendo establecer la configuración adecuada para la máquina de estados borrosa.

- La primera aproximación al problema del diseño automático del sistema recurrente borroso es un esquema puramente competitivo, representado por el sistema tipo Pittsburgh. Cada individuo de la población que sufre el proceso evolutivo es una FFSM convenientemente codificada. Los operadores genéticos diseñados actúan mutando, cruzando y replicando las FFSM. La principal ventaja de este sistema es su convergencia a un mínimo global. Sin embargo, la complejidad computacional requerida para evolucionar la población es muy elevada ya que en cada iteración hay que evaluar un gran número de máquinas de estado borrosas con el conjunto de entrenamiento.

- Este problema se puede aliviar usando el solapamiento entre poblaciones. Esta técnica, sin embargo, puede derivar hacia una convergencia prematura del proceso evolutivo, aunque se han encontrado solapamientos para los que los resultados de entrenamiento y test no difieren demasiado de los obtenidos con solapamientos muy bajos, lo cual los hace ventajosos desde el punto de vista computacional.

- La segunda de las estrategias implementadas para resolver el problema del diseño automático trata sobre todo de reducir la carga computacional inherente a un sistema Pittsburgh. Se trata de un sistema Michigan, basado en meta-reglas. La idea consiste en que los individuos de la población que se hace evolucionar son meta-reglas que describen cambios a realizar sobre una máquina de estados borrosa. En función de cómo sea la estructura de la FFSM, la meta-regla describe los cambios a realizar para mejorar su eficiencia en la clasificación. Una población de meta-reglas bien evolucionada debe ser capaz de proponer los cambios adecuados dado el estado actual de la FFSM. En ese sentido, se ha contemplado este problema como un problema de un paso y no multi-paso, lo que entraña el riesgo de llevar a la FFSM a mínimos locales. Sin embargo, las pruebas realizadas, tanto en simulación como con datos reales, dan buenos resultados, aunque en el futuro se investigará la consideración del problema multi-paso para tratar de mejorar los resultados obtenidos.

- Se ha presentado además un estudio con datos simulados basado en un modelo de referencia, el modelo oculto de Markov. La utilización de este modelo tiene importancia desde el punto de vista de la validación de los algoritmos planteados, la investigación de sus parámetros y la comparación con clasificadores bien establecidos y estudiados para estos datos, como es el clasificador basado en la identificación del modelo mediante el algoritmo de Baum-Welch y el cálculo de la probabilidad condicionada a la clase $P(O \mid \lambda)$.

- La utilización de series de datos para caracterizar la distribución de cromatina en los núcleos celulares y el modo de extraerlas son otras de las aportaciones realizadas en esta tesis. Los experimentos presentados establecen una capacidad

discriminante en estas series que deberá ser investigada más a fondo en combinación con otras características más usuales. Esta capacidad se ha visto reflejada no sólo con los clasificadores basados en la máquina de estados borrosa sino con las redes neuronales con propagación hacia delante.

- El problema de clasificación de núcleos en imágenes de citologías es un problema complejo que depende en gran medida de la experiencia del especialista. Existen sistemas que automatizan en mayor o menor grado alguno de los protocolos médicos, sin embargo, en la gran mayoría de los casos el médico no cuenta con herramientas de análisis computerizado que le permitan mejorar su capacidad de diagnóstico. Este trabajo representa una contribución a este problema porque no se queda en el análisis de las características convencionales de los núcleos celulares, sino que trata de buscar nuevos aspectos que describan el estado de los mismos. En particular se ha investigado una medida global de la textura del núcleo frente a las medidas clásicas, que son de carácter más local y estadístico.

Líneas abiertas.

Estas son las principales líneas abiertas que se plantean en el futuro inmediato.

- Experimentación en espacios de búsqueda más complejos, introduciendo variedad en la definición del ancho de las funciones de pertenencia que representan los niveles de activación de los estados y de la entrada externa, así como la utilización de otras funciones de pertenencia como las triangulares y trapezoidales.

- Consideración del problema multi-paso. Esto lleva a una redefinición del sistema de asignación de créditos en el sistema de tipo Michigan, en donde debería considerarse una estimación de la mejora del clasificador por la aplicación de las meta-reglas activadas como consecuencia de la aplicación de la meta-regla a la que se está recompensando.

- Investigación sobre la influencia de parámetros críticos en el algoritmo, como aquellos que afectan a la inclusión de términos comodín en las meta-reglas del sistema tipo Michigan y que determinan su capacidad de generalización.

- Realización de un estudio comparativo de los algoritmos presentados con redes neuronales recurrentes, estudiando las ventajas y desventajas de ambas aproximaciones al problema.

- Aplicación de los algoritmos empleados para el análisis de núcleos celulares a la estructura sintáctica de tejidos (distribución de núcleos en un tejido), lo que supone un cambio de escala espacial en el análisis de las muestras.

- Obtención de descriptores globales de la textura en núcleos celulares adicionales a los usados en esta investigación. En la selección de estos descriptores será necesario evaluar y tener en cuenta la dependencia de los mismos con las condiciones experimentales.

- Construcción de un clasificador que incorpore características clásicas de los núcleos con los nuevos aspectos investigados, con el fin de completar la información suministrada al sistema clasificador y aumentar su eficiencia de clasificación.

- Aplicación de la técnica de diseño de máquinas de estado borrosas aquí expuesta a otros campos, como por ejemplo, en el control de procesos.

- Avanzar en el campo de la implementación de los algoritmos. Actualmente, se ha investigando sobre la plataforma Matlab. Sin embargo, se aumentaría mucho la eficiencia computacional y se acortarían los tiempos requeridos para las pruebas si se pasase a una implementación paralela.

Referencias.

[Abe y Thawonmas, 1997] Abe, S. Y Thawonmas, R. (1997). A fuzzy classifier with ellipsoidal regions. *IEEE Transactions on Fuzzy Systems*, 5, pp. 358-368.

[Adams y Bischof, 1994] Adams, R., y Bischof, L. (1994). Seeded Region Growing, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 16, No. 6, pp. 641-647.

[Aguilar, 1994] Aguilar, R. (1994). *Sistema expero para la extracción automática de características del potencial evocado visual.* Tesina, Centro Superior de Informática, Universidad de La Laguna, Tenerife.

[Aho et al., 1986] Aho, A., Sethi, R. y J.D., U. (1986). *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Publishing Company, Reading, Massachusetts.

[Akaike, 1973] Akaike, H. (1973). Information Theory and an Extension of the Masimum Likelihood Principle. En *Second International Symposium on Information Theory*, eds. Petrov, B.N. y Cáski, F., pp. 267-281. Budapest: Akademiai Kaidó.

[Akaike, 1974] Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19, pp. 716-723.

[Antonisse, 1989] Antonisse, J. (1989). A new interpretation of schema notation that overturns the binary encoding constraint. *Proceedings of the Third International Conference on Genetic Algorithms*.

[Bagley, 1967] Bagley, J.D. (1967). The behaviour of adaptative systems which employ genetic and correlation algorithms. Tesis. Universidad de Michigan.

[Baglio et al., 1993] Baglio, S., Fortuna, L., Graziana, S., y Muscato, G. (1993). Membership function shape and the dynamical behaviour of a fuzzy system. En *Proceedings First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, Aachen, Alemania, pp. 645-650.

[Bamber, 1975] Bamber, D. (1975). The area above the ordinal dominance graph and the area below the receiver operating graph. *J Math Psych*, 12, pp. 387-415.

[Banzhaf, 1998] Banzhaf W., Nordin P., Keller R.E., y Francone F.D. (1998). *Genetic Programming*, Morgan Kaufmann Publishers, Inc.

[Bardossy y Duckstein, 1995] Bardossy, A. y Duckstein, L. (1995). *Fuzzy Rule-Based Modeling with Application to Geophysical, Biological and Engineering Systems*. CRC Press.

[Barro et al., 2001] Barro, S., Marín, R., Palacios, F. y Ruíz, R. (2001). Fuzzy logic in a patient supervision system. *Artificial Intelligent in Medicine*, 21, pp. 193-199.

[Beck y Schultz, 1986] Beck, J.R. y Shultz, E.K. (1986). The use of relative operating characteristic (ROC) curves in test performance evaluation. *Arch Pathol Lab Med*, 110, pp. 13-20.

[Berenji, 1992] Berenji, H. (1992). Fuzzy logic controllers. En *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Eds. Yager, R.R. y Zadeh, L.A., pp. 69-96. Kluwer Academic Press.

[Bersini y Varela, 1994] Bersini, H. Y Varela, F. (1994). The immune learning mechanisms: Recruitment reinforcement and their applications. En R. Patton (editor) *Computing with Biological Metaphors*. Chapman and Hall.

[Bersini, 1993] Bersini, H. (1993). Immune network and adaptive control. *Toward a Practica of Autonoumous Systems – Proceedings of the First ECAL,* pp. 217-225, MIT Press.

[Bezdek y Pal, 1992] Bezdek, J.C. y Pal, S.K. (1992). *Fuzzy Models for Pattern Recognition. Methods that Search for Structures in Data.* IEEE Press.

[Bibbo et al., 1981] Bibbo, M., Bartels, P.H., Sychra, J.J., y Weid, G.L. (1981). Chromatin appearance in intermediate cells from patients with uterince cancer, *Acta Cytologica,* vol. 25, p. 23-28.

[Bibbo et al., 1990] Bibbo, M., Miche, F., Bartels, P.H., Dytch, H., Bania, C., Lerma, E., y Montag, A.G. (1990). Karyometric maker features in normal-appearing glands adjacent to human colonic adenocarcinoma, *Cancer Research,* vol. 50, p. 147-151.

[Binaghi et al., 1996] Binaghi, E., Brivio, P.A. y Rampini, A. (1996). *Soft Computing in Remote Sensing Data Análisis*. World Scientific.

[Bishop, 1995] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Booker, 1982] Booker, L.B. (1982). *Intelligent Behavior as an Adaptation to the Task Environment.* Tesis. Department of Computer and Communication Science. University of Michigan.

[Borst et al., 1979]. Borst, H., Abmayr, W., y Gais, P. (1979). A thresholding method for automatic cell image segmentation, *J. Histochem. Cytochem,* 27(1):180-7.

[Breiman et al., 1984] Breiman, L., Friedman, J.H., Losen, R.A. y Stone, C.J. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks/Cole.

[Bridle, 1990] Bridle, J.S. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. En *Neural Computing: Algorithms, Architectures and Applications*. Eds. F. Fogleman Soulié y J. Hérault, pp. 227-236. Berlin.

[Burger et al., 1981] Burger, G., Jütting, U., y Rodenacker,K. (1981). Changes in benign cell populations in cases of cervical cancer and its precursors. *Analytical and Quantitative Cytology*, 3(4):261–271.

[Burger et al., 1986] Burger, G., Jütting, U., Aubele, M., y Auer, G. (1986). The role of chromatin pattern in automated cancer cytopathology, *Quantitative image analysis in cancer cytology and histology,*Mary J., Rigaut J editores, 91-102, Elsevier Science Publisher Amsterdam.

[Burgueño et al., 1995] Burgueño, M.J., García-Bastos, J.L. y González-Buitrago, J.M. (1995). Las curvas ROC en la evaluación de las pruebas diagnósticas. *Med Clin* (Barc), 104, pp. 661-670.

[Canny, 1986] Canny, J.A. (1986). Computational Approach to Edge Detection, *IEEE Trans. Pattern Analysis and Machine Intelligence* Vol. 8, No. 6, pp. 679-698.

[Carse et al., 1996] Carse, B., Fogarty, T.C., y Munro, A. (1996). Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems*, 80, pp. 273-294.

[Cavicchio, 1970] Cavicchio, D.J. (1970). *Adaptative search using simulated evolution*. Tesis. Universidad de Michigan, Ann Arbor.

[Chang et al., 1991] Chang, T.C., Hasegawa, K, y Ibbs, C.W. (1991). The effects of membership functions in fuzzy reasoning. *Fuzzy Sets and Systems*, 44, pp. 169-186.

[Chi et al., 1996] Chi, Z., Yan H. y Pham, T. (1996). *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. World Scientific.

[Chou et al., 1992] Chou, J.S., Chen, C.T., Chen, S.Y., y Lin, W.C. (1992). Parallel Implementation of an Adaptive Split-and-Merge Method for Medical Image Segmentation *SPIE Medical Imaging VI: Image Processing* Vol. 1652, pp. 62-67.

[Chow et al., 1999a] chow, M.Y., Altug, S. y Trussel, H. (1999). Heuristic constraints enforcement for training of and knowledge extraction form a fuzzy/neural architecture – part I: Foundation. *IEEE Transactions on Fuzzy Systmes*, 7(2), pp. 143-150.

[Chow et al., 1999b] chow, M.Y., Altug, S. y Trussel, H. (1999). Heuristic constraints enforcement for training of and knowledge extraction form a fuzzy/neural architecture – part II: Implementation and application. *IEEE Transactions on Fuzzy Systmes*, 7(2), pp. 151-159.

[Chow y Kaneko, 1972] Chow, C.K., y Kaneko, T. (1972). Automatic Boundary Detection of the Left Ventricle from Cineangiograms, *Computers and Biomedical Research* Vol. 5, pp. 388-410.

[Cordón et al., 1997] Cordón, O., Herrera, F., y Peregrín, A. (1997). Applicability of the fuzzy operators in the design of fuzzy logic controllers. *Fuzzy Sets and Systems*, 86, pp. 15-41.

[Cordón et al., 1998a] Cordón, O., del Jesús, M.J. y Herrera, F. (1998). Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods. *International Journal of Intelligent Systems,* 13(10-11), pp. 1025-1053.

[Cordón et al., 1998b] Cordón, O., Herrera F. y Sánchez, L. (1998). Computing the spanish médium electrical line maintenance costs by means of evolution–based learning processes. *Proceedings Eleventh International Conference on Industrial & Engineering Applications of Artificial Intelligence*. Eds. Ali, M. del Pobil, A.P. y Mira, J. pp. 478-486. Berlin: Springer-Verlag.

[Cordón et al., 2001] Cordón, O., Herrera, F., Hoffmann, F., y Magdalena, L. (2001). *Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Kwonledge Bases*. Advances in Fuzzy Systems – Applications and Theory. Vol. 19. World Scientific, Singapore.

[Dasarathy, 1991] Dasarathy, B.V. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press.

[Dash y Liu, 1997] Dash, M. y Liu, H. (1997). *Feature Selection for Classification*. Intelligent Data Analysis, Elsevier Science Inc.

[Davis y Smith, 1985] Davis, L. y Smith, D. (1985). *Adaptative design of layout synthesis* (Texas Instruments internal report). Dallas: Texas Instruments.

[Davis, 1991] Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

[De Groot, 1970] De Groot, M.H. (1970). *Optimal Statistical Decisions.* McGraw Hill. New York.

[De Jong et al., 1993] De Jong, K.A., Spears, W.M. y Gordon, D.F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13, 161-188.

[De Jong, 1975] De Jong, K.A. (1975). *An análisis of the behaviour of a class of genetic adaptive systems*. Tesis. Universidad de Michigan.

[De La Maza y Tidor, 1991] de la Maza, M. y Tidor, B. (1991). Increased flexibility in genetic algorithms: the use of variable Boltzmann selective pressure to control propagation. *Proceedings of the ORSA CSTS Conference: Computer Science and Operations Research: New Developments in Their Interfaces*.

[Delgado et al., 1997] Delgado, M., Gómez-Skarmeta, A.F. y Martín, F. (1997). A fuzzy clustering based rapid-prototyping for fuzzy rule-based modeling. *IEEE Transactions on Fuzzy Systems*, 5(2), pp. 223-233.

[Delgado et al., 1998] Delgado, M., Vila, M.A. y Voxman, W. (1998). A fuziness measure for fuzzy numbers: applications. *Fuzzy Sets and Systems*, 94(2), pp. 205-216.

[Devijver y Kittler, 1982] Devijver, P.A. y Kittler, J.V. (1982). *Pattern Recognition. A Statistical Approach*. Englewood Cliffs, NJ: Prentice Hall.

[Díaz-Flores et al., 1974] Díaz-Flores L y cols. (1974). *Anatomía Patológica*, Edit. J. Juberías, 1ª Ed., Granada.

[Díaz-Flores et al., 1977] Díaz-Flores L. (1977). *Lecciones de Citología*, Edit. J. Juberías, 1ª Ed., Granada.

[Díaz-Flores et al., 1982] Díaz-Flores L. y cols. (1982). *Compendio de Histología I*, Edit. P. Moreno, 1ª Ed., Granada.

[Díaz-Flores, et al., 1978] Díaz-Flores L. y cols. (1978). *Lecciones de Histología*, Edit. J. Juberías, 1ª Ed., Granada.

[Díaz-Flores, et al., 1979] Díaz-Flores L. y cols. (1979). *Anatomía Patológica General. Vol. I y II*, Edit. J. Juberías, 1ª Ed., Granada.

[Dorigo y Bersini, 1994] Dorigo, M y Bersini, H. (1994) *Proceedings of From Animals to Animats,3rd International Conference on Simulation of Adaptive Behaviour (SAB94)*, Brighton, UK.

[Dreyer et al., 1999] Dreyer, T., Garner, D., Doudkine, A., Knolbauch, I., y Popella, C. (1999). Malignancy associated changes: A new diagnostic tool in early laryngeal cancer detection?, (Abstract A051), 6th ESACP Congress, Heidelberg.

[Driankov et al., 1993] Driankov, D., Hellendoorn, H. y Reinfrank, M. (1993). *An introduction to Fuzzy Control*. Springer-Verlag.

[Duda et al., 2001] Duda, R.O., Hart, P.E., y Stork, D.G. (2001). *Pattern Classification*, Wiley Interscience.

[Duda y Hart, 1973] Duda, R.O., y Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, Wiley.

[Einstein et al., 1998] Einstein, A.J., Wu, H., Sanchez, M., y Gil, J. (1998). Fractal Characterization of Chromatin Appearance for Diagnosis in Breast Cytology, *Journal of Pathology,* 185:366-381.

[Englander, 1985] Englander, A.C. (1985). Machine learning of visual recognition using genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms and ther Applications*, pp. 197-201.

[Eshelman y Schaffer, 1993] Eshelman, L.J. y Schaffer, J.D. (1993). Real-coded genetic algorithms and interval-schemata. En *Foundations of Genetic Algorithms-2*, Whitley, D. (ed.), Morgan Kaufmann, San Mateo, pp. 187-202.

[Estévez et al., 2003] Estévez., J., Alayón, S., Moreno, L., Sigut, J., y Aguilar, R. (2003). Cytological Images Análisis with a Genetic Fuzzy Finite State Machine. *International Journal of Medical Informatics*. En prensa.

[Estévez et al., 2002a] Estévez, J., Alayón, S., Moreno, L., Aguilar, R. y Sigut, J. (2002). Cytological Breast Fine Needle Aspirate Image Análisis with a Genetic Fuzzy Finite State Machine, *Proceedings of the 15th IEEE Symposim on Computer-Based Medical Systems,* pp. 21-26, Maribor, Slovenia.

[Estévez et al., 2002b] Estévez, J., Alayón, S., Moreno, L., Aguilar, R., y Sigut, J. (2002). Diseño de una Clase de Sistemas Borrosos Recurrentes Mediante Algoritmos Genéticos Para la Tarea de Clasificación de Patrones, *XXIII Jornadas de Automática,* La Laguna, S/C de Tenerife.

[Estévez et al., 2002c] Estévez, J.I., Moreno, L., Alayón, S., Aguilar, R. y Sigut. J. (2002). Two problems in Fuzzy State Machine design: structure definition and rule base simplification. *Proceedings International Federation of Automatic Control (IFAC 2002)*, Barcelona, España.

[Estévez et al., 2001] Estévez, J., Hamilton, A., y Moreno, L. (2001). Un modelo para el razonamiento aproximado: la lógica borrosa. *Las Matemáticas del Siglo XX. Una mirada en 101 artículos*. Ed. Antonio Martinón. Editorial Nívola.

[Estévez, 2001] Estévez, J. (2001). *Metodología Basada en Lógica Difusa para el Análisis de Señales Continuas con Contenido Simbólico*, Centro Superior de Informática, Universidad de La Laguna, Tenerife.

[Fisher, 1936] Fisher, R.A. (1936). The use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7, pp. 179-188.

[Fitzpatrick et al., 1984] Fitzpatrick, J.M., Grefenstette, J.J. y Van Gucht, D. (1984). Image registration by genetic search. *Proceedings of IEEE Southeast Conference*, pp. 460-464.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press.

[Gauch, 1999] Gauch, J.M. (1999). Image Segmentation and Analysis via Multiscale Gradient Watershed Hierarchies, *IEEE Trans. Image Processing* Vol. 8, No. 1, pp. 69-79.

[Geman et al., 1992] German, S., Bienenstock, E. y Dousat, R. (1992). Neural Networks and the Bias/Variance dilema. *Neural Computation*, 4(1), pp. 1-58.

[Gharahmani y Jordan, 1994] Gharahmani, Z. Y Jordan, M.I. (1994). Supervised learning from incomplete data via EM approach. En Cowan, J.D., Tesauro, G.T. y Alspector, J. (Eds.), *Advances in Neural Information Processing Systems*, 6, pp. 120-127. San Mateo, CA: Morgan Kaufmann.

[Gillies, 1985] Gillies, A.M. (1985). *Machine learning procedures for generating image domain feature detectors*. Tesis. Universidad de Michigan, Ann Arbor.

[Goldberg, 1983] Goldberg, D.E. (1983). *Computer-aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning,* Tesis. University of Michigan Press, Ann Arbor, MI.

[Goldberg, 1989] Goldberg, D.B. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[Golston et al., 1990] Golston, J.E., Moss, R.H., y Stoecker, W.V. (1990). Boundary Detection in Skin Tumour Images: an Overall Approach and a Radial Search Algorithm, *Pattern Recognition* Vol. 23, No. 11, pp. 1235-1247.

[González et al., 1993] González, A., Pérez R. y Verdegay, J.L. (1993). Learning the structure of a fuzzy rule: a genetic approach. *Proceedings First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, Aachem, Alemania, pp. 814-819.

[González et al., 1995] González, A., Pérez R. y Valenzuela, A. (1995). Diagnosis of myocardial infarction through fuzzy learning techniques. *Proceedings Sixth International Fuzzy Systems Association World Congress (IFSA'95)*, Sao Paulo, Brasil, pp. 273-276.

[González y Pérez, 1998] González, A. y Pérez, R. (1998). Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems,* 96, pp. 37-51.

[Gorrini y Bersini, 1994] Gorrini, V. y Bersini, H. (1994). Recurrent fuzzy systems. *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, pp. 193-198.

[Grefenstette y Baker, 1989] Grefenstette, J.J. y Baker, J.E. (1989). How genetic algorithms work: a critical look at implicit parallelism. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 20-27, San Mateo, CA. Morgan Kauffman, San Francisco, CA.

[Grefenstette y Fitzpatrick, 1985] Grefenstette, J.J. y Fitzpatrick, J.M. (1985). Genetic search with approximate function evaluations. *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pp. 112-120.

[Grefenstette, 1991] Grefenstette, J.J. (1991). Lamarckian learning in multi-agent environments. En Belew, R. Y Booker, L. (eds.). *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann.

[Grefenstette, 1998] Grefenstette, J.J. (1998). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning,* 3, 225-245.

[Hallinan, 1999] Hallinan, J.S. *Detection of Malignant Associated Changes in Cervical Cells Using Statistical and Evolutionary Techniques*, Thesis, Cytometrics Project, Center for Sensor Signal and Information Processing, University of Queensland, 1999.

[Hand, 1982] Hand, D.J. (1982). *Kernel Discriminat Analysis*. Chichester: Research Studies Press.

[Hanley y McNeil, 1982] Hanley, J.A. y McNeil, B.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143, pp. 29-36.

[Hanley y McNeil, 1983] Hanley, J.A. y McNeil, B.J. (1983). A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148, pp. 839-843.

[Hanley, 1988] Hanley, J.A. (1988). The robustness of the binormal model used to fit ROC curves. *Med Decision Making*, 8, pp. 197-203.

[Hanley, 1996] Hanley, J.A. (1996). The use of the "binormal" model for parametric ROC analysis of quantitative diagnostics tests. *Statist Med*, 15, pp. 1575-1585.

[Haralick et al., 1973]. Haralick, R.M., Shanmugan, K., y Dinstein, I. (1973). Textural features for image classification. *IEEE Trans. On Systems, Man, and Cybernetics,* SMC-3:610-621.

[Haris et al., 1998] Haris, K., Efstratiadis, S.N., Maglaveras, N., y Katsaggelos, A.K. (1998). Hybrid Image Segmentation Using Watersheds and Fast Region Merging, *IEEE Trans. Image Processing* Vol. 7, No. 12, pp. 1684-1699.

[Harris et al., 1993] Harris, C.J., Moore, C.G. y Brown, M. (1993). *Intelligent Control. Aspects of Fuzzy Logic and Neural Networks*. World Scientific.

[Hedstrom et al., 1996] Hedstrom, J., Sainio, V., Kemppainen, E., Haapianen, R., Kivilaaksо, E., Schroder, T., Leinonen, J. y Setnman, U.H. (1996). Serum complex of trypsin 2 and (alpha)(sub 1) antitrypsin as diagnostic and prognostic marker of acute pancreatitis: clinical study in consecutive patients. *Br Med* J 313, pp. 333-337.

[Hirota, 1993] Hirota, K. (1993). *Industrial Applications of Fuzzy Technology.* Springer-Verlag.

[Holland y Reitman, 1978] Holland, J.H., Reitman, J.S. (1978). Cognitive systems based on adaptive algorithms. En D.A. Waterman y F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems.* Academic Press.

[Holland, 1976] Holland, J.H. (1976). Adaptation. En R. Rosen y F.M. Snell (Eds.), *Progress in Theoretical Biology IV,* pp. 263-293. Academic Press.

[Holland, 1975] Holland, J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press.

[Holland, 1981] Holland, J.H. (1981). *Genetic Algorithms and Adaptation,* Technical Report 34. University of Michigan, Ann Arbor, Dept. of Computer and Communication Sciences.

[Holland, 1986] Holland, J.H. (1986). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-base systems. En Michalski, R., Carbonell, J. y Mitchell, T. (eds.), *Machine Learning: An artificial intelligence approach* (vol. 2). San Mateo, CA: Morgan Kaufmann.

[Hsieh y Turnbull, 1996] Hsieh, F., Turnbull, B.W. (1996). Nonparametric and semiparametric estimation of the receiver operating characteristic curve. *Ann Statist*, 24, pp. 25-40.

[Hu, 1962] Hu, M.K. (1962). Visual pattern recognition by moment invariants. *IRE Trans.* IT, 8:179-187.

[Hugues, 1968] Hugues, G.F. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions Info. Theory,* IT-14, pp. 55-63.

[Hunstein et al., 1986] Hunstein, W., Pilcher, J., Klink, J. y Schulz, H. (1986). Automatic analysis overcomes limitations of sleep stage scoring. *Electroencephalography and clinical neurophysiology*, 64, pp. 364-374.

[Ishibuchi et al., 1992] Ishibuchi, H., Nozaki, K. y Tanaka, H. (1992). Distributed representation of fuzzy rules and its application to pattern classification. *Fuzzy Sets and Systems*, 52, pp. 21-32.

[Jang et al., 1997] Jang, J.S., Sun, C.T., y Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*. Matlab Curriculum Series. Prentice Hall, Upper Saddle River, NJ 07458.

[Jang y Sun, 1993] Jang, J.S., y Sun, C.T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1), pp. 156-159.

[Jang, 1993] Jang, J.S. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man, and Cybernetics*, 23(3), pp. 665-684.

[Janikow, 1993] Janikow, C.Z. (1993). A knowledge-intensive GA for supervised learning. *Machine Learning*, 13, 189-228.

[Jarkans et al., 1980] Jarkrans, T., Bengtsson, E., Eriksson, O., Nordin, B., y Stenkvist, B. (1980). Using Tracking Algorithms for Cell Segmentation, *Proc. 1st Scandinavian Conference on Image Analysis*, pp. 325-330.

[Jiang et al., 1996] Jiang, Y., Metz, C.E. y Nishikawa, R.M. (1996). A receiver operating characteristic partial area index for highly sensitive diagnostics tests. *Radiology*, 201, pp. 745-750.

[Jiménez y Landgrebe, 1998] Jiménez, L. y Landgrebe, D. (1998). Supervised Classification in high dimensional space: geometrical, statistical ans asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics*.

[Jütting et al. 1999] Jütting, U., Gais, P., Rodenacker, K., Böhm, J., Koch, S., Praner, H.W, y Höfler, H. (1999). Diagnosis and prognosis of neuroendocrine tumours of the lung by means of high resolution image analysis. *Anal Cell Pathol,* 18(2):109-19.

[Kang et al., 2000] Kang, S.-J., Woo, C.-H., Hwang, H.-S. y Woo, K.B. (2000). Evolutionary design of fuzzy rule base for nonlinear system modelling and control. *IEEE Transactions on Fuzzy Systems 8(1),* 37-45.

[Kass et al., 1987] Kass, M., Witkin, A., y Terzopoulos, D. (1987). Snakes: Active Contour Models, *Proc. 1st International Conference on Computer Vision*.

[Kittler y Illingworth, 1985] Kittler, J., y Illingworth, J. (1985). Minimum Error Thresholding, *Pattern Recognition*, Vol. 19, No. 1, pp. 41-47.

[Kosko y Dickerson, 1995] Kosko, B. y Dickerson, J. (1995). Function approximation with additive fuzzy systems. En Nguyen, H., M., S., R., T., y R.R., Y., editores, *Theoretical Aspects of Fuzzy Control*, pp. 313-347. John Wiley & Sons, Inc.

[Kosko, 1994] Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11), pp. 1329-1332.

[Kovacs, 2002] Kovacs, T (2002). Two Views of Classifier Systems. En P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editores, *Advances in Learning Classifier Systems*, Vol. 2321, pages 74-87. Springer-Verlag.

[Lazzerini et al., 1999] Lazzerini, B., Reyneri, L.M. y Chiaberge, M.. (1999). A neuro-fuzzy approach to hybrid intelligent control. *IEEE Transactions on Industry Applications*, 35(2), pp. 413-425.

[Lee y Takagi, 1996] Lee, C.C. y Takagi, H. (1996). Hybrid genetic-fuzzy systems for intelligent systems design. En Herrera, F. y Verdegay, J.L. (Eds.), *Genetic Algorithms and Soft Computing,* Número 8 en Studies in Fuziness and Soft Computing, pp. 226-250. Physica-Verlag.

[Lee, 1990] Lee, C.C. (1990). Fuzzy logic in control systems: fuzzy logic controller – Parts I and II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2), pp. 404-418, pp. 419-435.

[Leow y Miikkulaien, 1994] Leow, W. y Miikkulainen, R. (1994). Visor: schema-based scene análisis with a structured neural networks. *Neural Processing Letters*, 1(2), pp. 18-23.

[Leow, 1994] Leow, W. (1994). Visor: learning visual schemas in neural networks for object recognition in scene analysis. Technical Report AI94-219.

[Lerma-Puertas et al., 1989] Lerma-Puertas, E., Galera-Davidson, H., Bartels, P.H., Kim, D.H., Dytch, H.E., y Bibbo, M. (1989). Karyometric marker features in fine leedle aspirates of follicular adenoma of the thyroid, *Analytical and Quantitative Cytology and Histology,* vol. 12, p. 223-228.

[Little y Rubin, 1987] Little, R.J. y Rubin, D.B. (1987). *Statistical Analysis with Missing Data*. New York: Wiley.

[Magdalena y Monasterio, 1995] Magdalena, L. y Monasterio, F. (1995). Evolutionary-based learning applied to fuzzy controllers. En *Proc. 4th IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'95,* Vol. 3, pp. 1111-1118. Yojohama, Japón.

[Magdalena, 1997] Magdalena, L. (1997). Adapting the gain of a FLC with genetic algorithms. *International Journal of Approximate Reasoning*, 14(4), pp. 327-349.

[Magdalena, 1998] Magdalena, L. (1998). Crossing unordered sets of rules in evolutionary controllers. *International Journal of Intelligent Systems 13(10/11)*, 993-1010.

[Mamdani y Assilian, 1975] Mamdani, E. y Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Machine Studies*, 7, pp. 1-13.

[Mamdani, 1974] Mamdani, E. (1974). Applications of fuzzy algorithm for control a simple dynamic plant. *Proceedings of the IEE*, 121(12), pp. 1585-1588.

[Mandal et al., 1992] Mandal, D.P., Murthy, C.A. y Pal, S.K. (1992). Formulation of a multivalued recognition system. *IEEE Transactions on Systems, Man, and Cybernetics,* 22, pp. 607-620.

[Mangasarian et al., 1995] Mangasarian, O., Street, N., y Wolberg, W. (1995). Breast Cancer Diagnosis and Prognosis via Linear Programming. *Operations Research*, 43(4), pp. 570-577.

[Marichal, 2003] Marichal, R. (2003).*Estudio de la Dinámica de una Red Neuronal Recurrente Discreta y su Aplicación a las Señales Biomédicas e Identificación de Sistemas*. Tesis Doctoral. Centro Superior de Informática. Universidad de La Laguna.

[Marr y Hildreth, 1980] Marr, D., y Hildreth, E. (1980). Theory of Edge Detection, *Proc. Royal Society* Vol. B207, pp. 187-217

[McClish, 1989] McClish, D.K. (1989). Analyzing a portion of the ROC curve. *Med Decision Making*, 9, pp. 190-195.

[McNeil et al., 1975] McNeil, B.J., Keeler, E. y Adelstein, S.J. (1975). Primer on certain elements of medical decision making. *N Engl J Med*, 293, pp. 211-215.

[Metz et al., 1998] Metz, C.E., Herman, B.A. y Shen, J. (1998). Maximum likelihood estimation of receiver operating characteristic (ROC) curves from continuously distributed data. *Statist Med*, 17, pp. 1033-1053.

[Michalewicz, 1992] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlín.

[Minkus et al., 1997] Minkus, G., Jütting, U., Aubele, M., Rodenacker, K., Gais., P., Breuner, W., y Hermanns, W. (1997). Canine neuroendocrine tumors of the pancreas: A study using image analysis techniques for the discriminaton of metastatic versus nometastatic tumors. *Vet Pathol,* 34:138-145.

[Moreno et al., 1995a] Moreno, L., Piñeiro, J.D., Sánchez, J.L., Mañas, S., Merino, J., Acosta, L. y Hamilton, A. (1995). Brain Maturation Estimation Using Neural Classifier, *IEEE Transactions on Biomedical Engineering.* Vol 42, No. 4, pp 428-432.

[Moreno et al., 1995b] Moreno, L., Piñeiro, J.D., Sánchez, J.L., Mañas, S. , Merino, J., Acosta, L. y Hamilton, A. (1995). Using Neural Networks to Improve Classification: Application to Brain Maturation, *Neural Networks*, Vol. 8, Nº 5, pp. 815-820.

[Moreno et al., 1997] Moreno, L., Estévez, J.I., Mañas, S., Merino, J. Sigut, J., Piñeiro, J. y Aguilar, R. (1997). Design of a system based on fuzzy logic for the detection of eeg morphologies. *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*. La Laguna, Tenerife.

[Moreno et al., 2000] Moreno, L., Estévez, J.I., Aguilar, R., Sánchez, J. Sigut, J., Piñeiro, J. y Marichal, R. (2000). Automatic análisis of signals with symbolic content. *Artificial Intelligence in Medicine*, 18, pp. 245-265.

[Moreno et al., 2001a] Moreno, L., Estévez, J.I., Aguilar, R., Sigut, J., y González, C. (1997). Exploiting the advantages of symbolically interpretable continuos biomedical parameters with a fuzzyfied symbolic model. *Artificial Intelligence in Medicine,* 21, pp. 253-262.

[Moreno et al., 2001b] Moreno, L., Sánchez, J.L., Mañas, S., Piñeiro, J.D., Merino, J.J., Sigut, J.F., Aguilar, R., Estévez, J. y Marichal, R. (2001). Tools for Acquisition, Processing and Knowledge-Based Diagnostic of the Electroencephalogram and Visual Evoked Potentials, *Journal of Medical Systems*, Vol. 25, pp. 177-194.

[Muders et al., 1997] Muders, F., Kromer, E.P., Griese, D.P., Pfeifer, M., Hense, H.W., Riegger, G.A.J. y Elsner, D. (1997). Evaluation of plasma natriuretic peptides as markers of left ventricular dysfunction. *Heart* J 134(3), pp. 442-450.

[Nomura et al.,1991] Nomura, H., Hayashi, H. Y Wakami, N. (1991). A self-tuning method of fuzzy control by descendent method. *Proceedings Fourth International Fuzzy Systems Association World Congress (IFSA'91)*, Bruselas, Bélgica, pp. 155-158.

[Ohta et al., 1980] Ohta, Y.U., Kanade, T., y Sakai, T. (1980). Color Information for Region Segmentation. *Computer Graphics and Image Processing* Vol. 13, pp. 222-241.

[Oliveira, 1995] Oliveira, J. (1995). A design methodology for fuzzy system interfaces. *IEEE Transactions on Fuzzy Systems*, 3(4), pp. 404-414.

[Ossen et al., 1994] Ossen, A., Zamzow, T., Oswald, H., y Fleck, E. (1994). Segmentation of Medical Images using Neural-Network Classifiers, *Proc. Int. Conf. on Neural Networks and Expert Systems in Medicine and Healthcare* NNESMED 94, Plymouth, UK, pp. 427-432.

[Palcic et al., 1993] Palcic, B., Susnik, B., Garner, D., y Olivotto, I. (1993). Quantitative evaluation of malignant potential of early breast cancer using high resolution image cytometry, *Journal of Cellular Biochemistry Supplement*, 17G, p.107-113.

[Palcic et al., 1998] Palcic, B., Payne, P., MacAulay, C., y Lam, S. (1998). Malignancy Associated Changes (MAC) in Lung Cancer, *Laboratory Testing of Solid Tumoirs*, Blackwell Scientific Publlications.

[Parkes et al., 1995] Parkes, C., Wald, N.J., Murphy, P., George, L., Watt, H.C., Kirby, R., Knekt, P., Helzlsouer y K.J.,Tuomilehto, J. (1995). Prospective observational study to assess value of prostate specific antigen as screening test for prostate cancer. *Br Med* J 311, pp. 1340-1343.

[Parodi y Bonelli, 1993] Parodi, A. y Bonelli, P. A new approach to fuzzy classifier systems. En *Proc. Fifth International Conference on Genetic Algorithms (ICGA'93),* pp. 223-230, Morgan Kaufmann.

[Pérez et al., 1996] Pérez, S.V.M., Ibarra, R.M., Ángeles, R.M. y Meneses, G.A. (1996). Biopsia por aspiración con aguja fina de glándula salival. *Revista del Instituto Nacional de Cancerología de México*, 42(1), pp. 9-15.

[Perona y Malik, 1990] Perona, P., y Malik, J. (1990). Scale-Space and Edge Detection Using Anisotropic Diffusion, *IEEE Trans. Pattern Analysis and Machine Intelligence* Vol. 12, No. 7, pp. 629-639.

[Piñeiro et al., 1998a] Piñeiro, J.D., Marichal, R.L., Moreno, L., Sigut, J.F., Estévez, J.I., Aguilar, R.M., Sánchez, J.L., y Merino, J.J. (1998). Evoked Potential feature detection with Recurrent Dynamic Neural Networks, *International Symposium ICSC/IFAC on Neural Computation - NC'98*.

[Piñeiro et al., 1998b] Piñeiro, J.D., Sigut, J.F., Moreno, L., Estévez, J.I., Aguilar, R.M., Sánchez, J.L., Mañas, S., y Merino, J.J. (1998). A Knowledge-Based System for Aiding in the Diagnosis in Neurophysiology. *International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*.

[Piñeiro et al., 2000] Piñeiro, J.D., Marichal, R.L., Moreno, L., Sigut, J.F., Estévez, J.I., y Aguilar, R.M. (2000). Dynamics of a Small Discrete Recurrent Neural Network, *International Symposium on Neural Computation - NC'2000*.

[Piñeiro et al., 2001] Piñeiro, J.D., Marichal, R.L., Moreno, L., Sigut, J.F., y González, E.J. (2001). Study of Chaos in a Simple Discrete Recurrence Neural Network en Connectionist Models of Neurons, *Learning Processes and Artificial Intelligence*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2084, pp 579-585.

[Piñeiro et al., 2002] Piñeiro, J.D., Marichal, R.L., Sigut, J.F., Estévez, J.I., González, E.J., y Aguilar, R.M. (2002). Analysis of Pseudo-Chaotic Orbits in a Simple Discrete Recurrent Neural Network, *International Symposium on Engineering of Intelligent Systems - EIS'2002*.

[Piñeiro, 1996] Piñeiro, J. (1996). *Hacia un Sistema Basado en el Conocimiento para el Diagnóstico de Patologías en Señales Cerebrales.* Tesis Doctoral, Centro Superior de Informática, Universidad de La Laguna, Tenerife.

[Rabiner y Juang, 1986] Rabiner, L.R. y Juang, B.H. (1986). An introduction to hidden Markov models, *IEEE ASSP Magazine*, 3(1), pp. 4-16.

[Rabiner y Juang, 1993] Rabiner, L.R. y Juang, B.H. (1993). *Fundamentals os Speech Recognition*, Prentice Hall Signal Processing Series.

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden markov models and selected application sin speech recognition. *Proceeding IEEE, 77(2):257-286*

[Radcliffe, 1991] Radcliffe, N.J. (1991). Formal analysis and random respectful recombination. *Proceedings 4th International Conference on Genetic Algorithms*, Morgan Kauffman, San Mateo, CA, pp. 222-229.

[Raghavan y Bichard, 1979] Raghavan, V.V. y Bichard, K. (1979). A clustering strategy based on a formalism of the reproductive processes in natural systems. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 241-246.

[Rechenberg, 1994] Rechenberg, I. (1994). *Evolutionsstrategie'93*. Frommann Verlag, Stuttgart, Alemania.

[Reyneri, 1997] Reyneri, L.M. (1997). An introduction to fuzzy state automata. *Proceedings International Work-Conference on Artificial and Natural Neural Networks, (IWANN '97)*, Lanzarote, España, pp. 273-283.

[Ripley, 1996] Ripley, B.D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

[Robertson y Zweig, 1981] Robertson, E.A. y Zweig, M.H. (1981). Use of receiver operating characteristic curves to evaluate the clinical performance of analytical systems. *Clin Chem,* 27, pp. 1569-1574.

[Rodenacker et al., 1981] Rodenacker, K., Gais, P., y Jütting, U. (1981). Segmentation and measurement of the texture in digitized images, *Stereol Iugosl,* 3(Suppl I):165-174.

[Rodenacker et al., 1992] Rodenacker, K., Aubele, M., Burger, G., Gais, P., Jütting, U., y Gössner, W. (1992). Cytometry in histological sections of colon carcinoma. *Pathology Research and Practice,* 188:556-560.

[Rodenacker, 2001] Rodenacker, K. (2001). A Feature set for Cytometry of Digitized Microscopic Images, *Technical Report*, http://citeseer.nj.nec.com/rodenacker01feature.html.

[Rosenberg, 1970a] Rosenberg, R.S. (1970). Simulation of genetic populations with biochemical properties: I. The model. *Mathematical Biosciences*, 7, 223-257.

[Rosenberg, 1970b] Rosenberg, R.S. (1970). Simulation of genetic populations with biochemical properties: II. Selection of crossover probabilities. *Mathematical Biosciences*, 8, 1-37.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of Neurodynamics*. Washington, DC: Spartan Books.

[Saffiotti et al., 1995] Saffioti, A., Konolige, K. y Ruspini, E.H. (1995). A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2), pp. 481-526.

[Sahoo et al., 1988] Sahoo, P.K., Soltani, S., Wong, A.K.C., y Chen, Y.C. (1988). A Survey of Thresholding Techniques, *Computer Vision, Graphics and Image Processing,* Vol. 41, pp. 233-260.

[Salzarulo et al., 1991] Salzarulo, P., Fagiolo, I., Perviano, P., Bes, F. y Schulz, H. (1991). Levels of eeg background activity and sep states in the first year of life. En Terzano, M., Halasz, P. Y Declerck, A., eds., *Phasic events and dynamic organization of sleep*, pp. 53-63. Raven, New York.

[Sánchez, 1993] Sánchez, J.L. (1993). *Métodos para el Procesamiento y Análisis Estadístico Multivariante de Señales Multicanal: Aplicación al Estudio del EEG,* Tesis Doctoral, Centro Superior de Informática, Universidad de La Laguna, Tenerife

[Schmid y Fischer, 1997] Schmid, P., y Fischer, S. (1997). Colour Segmentation for the Analysis of Pigmented Skin Lesions, *Proc. 5th International Conference on Image Processing and its Applications*, pp. 688-692.

[Schulerud, 1997] Schulerud, H. (1997). Evaluation of morphological and textural nuclear image analysis as a tool in tumour pathology. Thesis 238. Departmaent of Informatics. Universoty of Oslo.

[Schwefel, 1995] Schwefel, H.P. (1995). *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. John Wiley & Sons, New York.

[Setnes et al., 1998] Setnes, M., Babu$\hat{s}$ka, R., Kaymak, U. y van Nauta Lemke, H. (1998). Similarity measures in fuzzy rule base simplification. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics,* 29(2), pp. 227-236.

[Shaefer, 1985] Shaefer, C.G. (1985). Directed trees method for fitting a potential function. *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pp. 207-225.

[Sigut et al., 2001] Sigut, J.F., Piñeiro, J.D., Marichal, R.L., y Moreno, L. (2001). Integración de Técnicas de Análisis y Clasificación mediante un Sistema Basado en el Conocimiento para Problemas de Diagnóstico. *XXII Jornadas de Automática*.

[Sigut, 2001] Sigut, J. (2001). *Integración de Técnicas de Análisis y Clasificación de Datos mediante un sistema Basado en el Conocimiento para el Diagnóstico de Patologías Cerebrales y Dislexias*, Centro Superior de Informática, Universidad de La Laguna.

[Singh et al., 2000] Singh, H., Sekinger, E., y Gross, D. (2000). Chromatin and Cancer: Causes and Consequences, *Journal of Cellular Biochemistry Supplement* 35:61-68.

[Smith y De Jong, 1981] Smith, T. y De Jong, K.A. (1981). Genetic Algorithms applied to the calibration of information driven models of US migration patterns. *Proceedings of the 12th Annual Pittsburgh Conference on Modelling and Simulation*, pp. 995-959.

[Smith, 1980] Smith, S.F. (1980). *A learning system based on genetic adaptive algorithms.* Tesis. Department of Computer Science. Universidad de Pittsburgh.

[Smith, 1983] Smith, S.F. (1983). Flexible learning of problem solving heuristics through adaptive search. En *Proc. Eighth International Joint Conference on Artificial Intelligence,* pp. 422-425. Morgan Kaufmann.

[Smith, 1989] Smith, G.M. (1989). Image texture analysis using zero crossings information, Thesis, University of Queensland, St. Lucia Brisbane, Queensland 4072, Australia.

[Smith, 1992] Smith, R.E. (1992). A Report on The First International Workshop on Learning Classifier Systems (1992), *The First International Workshop on Learning Classifier Systems*. NASA Johnson Space Center, Houston, Texas.

[Stadnyk, 1987] Stadnik, I. (1987). Schema recombination in pattern recognition problems. *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 27-35.

[Steimann, 1996] Steimann, F. (1996). The interpretation of time-varying data with diamon-1. *Artificial Intelligence in Medicine*, 8, pp. 343-357.

[Steimann, 1997] Steimann, F. (1997). Fuzzy set theory in medicine. *Artificial Intelligence in Medicine*, 11, pp. 1-7.

[Steimann, 2001] Steimann, F. (2001). On the use and usefulness of fuzzy sets in medical ai. *Artificial Intelligence in Medicine*, 21, pp. 131-137.

[Street et al., 1993] Street, W.N., Wolberg, W.H. y Mangasarian, O.L. (1993). Nuclear Feature Extraction for breast tumor diagnosis. *International Symposium on Electronic Imaging: Science and Technology*, vol. 1905, pp. 861-870.

[Sugeno y Kang, 1988] Sugeno, M. y Kang, G. (1988). Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28, pp. 15-33.

[Sugeno y Yasukawa, 1993] Sugeno, M. y Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems,* 1(1), pp. 7-31.

[Sundaram, 2000] Sundaram, R.H. (2000). The Baum-Welch algorithm, *Technical Report*, Mississippi State University.

[Surmann et al., 1995] Surmann, H., Huser, J. y Peters, L. (1995). A fuzzy system for indoor mobile robot navigation. *Proceedings of the Fourth IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'95*, Yokohama, Japón, pp. 83-88.

[Surmann y Maniadakis, 2001] Surmann, H. y Maniadakis, M. (2001). Learning feed-forward and recurrent fuzzy systems: a genetic approach. *Journal of Systems Architecture*, 47 (7), pp. 649-662.

[Susnik et al., 1995] Susnik, B., Worth, A., LeRiche, J., y Palcic, B. (1995). Malignancy-associated changes in the breast: changes in chromatin distribution in epithelial cells in normal-appearing tissue adjacent to carcinoma, *Analytical and Quantitative Cytology and Histology*, vol. 17, no. 1, p.62-68.

[Sutton y Barto, 1998] Sutton, R.S. y Barto, A.G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

[Swets y Pickett, 1982] Swets, J.A. y Pickett, R.M. (1982). *Evaluation of diagnostic systems: methods from signal detection theory*. Nueva York: Academic Press.

[Tackett, 1994] Tackett, W.A. (1994). *Recombination, Selection and the Genetic Construction of Computer Programs*. Tesis. Universidad de California.

[Takagi et al., 1992] Takagi, T., Suzuki, N., Koda, T. y Kojima, Y. (1992). Neural networks designed on aproxímate reasoning architecture and their applications. *IEEE Transactions on Neural Networks,* 3(5), pp. 752-760.

[Takagi y Hayashi, 1991] Takagi, T. y Hayashi, I. (1991). NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 5(3), pp. 191-212.

[Takagi y Sugeno, 1985] Takagi, T. Y Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15, pp. 116-132.

[Teller, 1996] Teller, A. (1996). Evolving programmers: the coevolution of intelligent recombination operators. En Angeline, P.J. y Kinnear, K.E. (eds.), *Advances in Genetic Programming 2*, capítulo 3, pp. 45-68. MIT Press, Cambridge, MA.

[Trillas, 1980] Trillas, E. (1980). *Conjuntos Borrosos*. Vicens Vives.

[Turksen, 1991] Turksen, I. (1991). Measurement of membership functions and their acquisition. *Fuzzy Sets and Systems*, 40, pp. 5-38.

[Twardowski, 1993] Twardoski, K. (1993). Credit assignment for pole balancing with learning classifier system. *Proceedings of Fifth International Conference on Genetic Algorithms,* pp. 238-245, Morgan Kaufmann, San Mateo, CA.

[Umbaugh et al., 1993] Umbaugh, S.E., Moss, R.H., Stoecker, W.V., y Hance, G.A. (1993). Automatic Color Segmentation Algorithms, *IEEE Engineering in Medicine and Biology* September, pp. 75-82.

[Umbers y King,1980] Umbers, I.G. y King, P.J. (1980). An analysis of human-decision making in cement kiln control and the implications for automation. *International Journal of Man-Machine Studies*, 12, pp. 11-23.

[Usen et al., 1999] Usen, S., Weber, M., Mulholland, K., Jaffar, S., Oparaugo, A., Omosigho, C., Adegbola, R. y Greenwood, B. (1999). Clinical predictors of hypoxaemia in Gambian children with acute lower respiratory tract infection: Prospective Cohort Study. *Br Med* J 318, pp. 86-91.

[Valenzuela-Rendón, 1991] Valenzuela-Rendón, M. (1991). The fuzzy classifier system: Motivations and first results, En H.P. Schwefel y R. Männer, *Proc. First International Conference on Parallel Problem Solving from Nature – PPSN I,* pp. 330-334. Springer-Verlag, Berlín.

[Vapnik, 1982] Vapnik, V.N. (1982). *Estimation of Dependencies Based on Empirical Data.* New York: Springer.

[Velasco y Magdalena, 1995] Velasco, J.R. y Magdalena, L. (1995). Genetic Algorithms in Fuzzy Control Systems. En J. Periaux, G. Winter, M. Galán y P. Cuesta (Editores), *Genetic Algorithms in Engineering and Computer Science,* pp. 141-165. John Wiley and Sons.

[Velasco, 1998] Velasco, J.R. (1998). Genetic-based on-line learning for fuzzy process control. *International Journal of Intelligent Systems 13(10-11)*, 891-903.

[Virant y Zimic, 1995] Virant, J. y Zimic, N. (1995). Fuzzy automata with fuzzy relief. *IEEE Transactions On Fuzzy Systems*, 3(1), pp. 69-74.

[Watanabe, 1969] Watanabe, S. (1969). *Knowing and Guessing*. New York: Wiley.

[Watkins, 1989] Watkins, C.J.C.H. (1989). Learning with delayed rewards. Tesis. Psychology Department. University of Cambridge, England.

[Wee y Fu, 1969] Wee, W. y Fu, K. (1969). A formulation of fuzzy automata and its application as a model of learning systems. *IEEE Transactions On Systems, Science and Cybernetics*, SSC-5, pp. 215-223.

[Weinstein y Fineberg, 1980] Weinstein, M.C., Fineberg, H.V. (1980). *Clinical Decision Analysis*. Philadelphia: WB Saunders Co.

[Weszka, 1978] Weszka, J.A. (1978). Survey of Threshold Selection Techniques, *Computer Graphics and Image Processing*, Vol. 7, pp. 259-265.

[Weyn, et al., 1999] Weyn, B., van de Wouver, G., Koprowski, M., van Daele, A., Dhaene, K., Scheunders, P., Jacob, W., y van Marck, E. (1999). Value of Morphometry, Texture Analysis, Densitometry, and Histometry in the differential diagnosis and prognosis of malignant mesothelioma, *Journal of Pathology,* 189:581-589.

[Wilson, 1985] Wilson, S.W. (1985). Knowledge Growth in an Artificial Animal. *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pp. 16-23.

[Wilson, 1992] Wilson, S.W. (1992). Classifier System Mapping of Real Vectors. The *First International Conference on Learning Classifier Systems*. NASA Johnson Space Center. Houston. Texas.

[Wilson, 1994] Wilson, S.W. (1994). ZCS: A Zeroth Level Classifier System. *Evolutionary Computation,* Vol. 2 (1), MIT Press, pp 1-18.

[Wilson, 1995] Wilson, S.W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation,* Vol. 3 (2), pp 149-175.

[Withley, 1989] Withley, D. (1989). The genitor algorithm and selection pressure: why rank-base allocation of reproductive trials is best. En Schaffer, J.D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116-121, San Mateo, CA. Morgan Kaufmann, San Francisco, CA.

[Wolberg et al., 1993] Wolberg, W., Street, W., y Mangasarian, O. (1993). Breast Cytology Diagnosis via Digital Image Analysis. *Analytical and Quantitative Cytology and Histology,* Vol. 15 No. 6, pp. 396-404.

[Wolberg et al., 1994] Wolberg, W.H., Street, W.N. y Mangasarian, O.L. (1994). Breast cytology diagnosis via digital image analysis. *Analytical and Quantitative Cytology and Histology*, 15 (6), pp. 396-404

[Wolberg, 1992] Wolberg,W. (1992). *Wisconsin Breast Cancer Database*, University of Wisconsin Hospitals, Madison.

[Wright, 1991] Wright, A. (1991). Genetic algorithms for real parameter optimitation. En *Foundations of Genetic Algorithms – 1*, Rawlin, G.J.E. (ed.). Morgan Kaufmann, San Mateo, CA, pp. 205-218.

[Wu et al., 1995] Wu, K., Gauthier, D., y Levine, M. D. (1995). Live Cell Image Segmentation, *IEEE Trans. Biomedical Engineering*, Vol. 42, No. 1, pp. 1-12.

[Yam et al., 1999] Yam, Y., Baranyi, P. Y Yang, C.T. (1999). Reduction of fuzzy rule base via singular value decomposition. *IEEE Transactions on Fuzzy Systems,* 7(2), pp. 120-132.

[Yen y Wang, 1999] Yen, J. Y Wang, L. (1999). Simplifying fuzzy rule-based models using orthogonal transformations methods. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics,* 29(1):13-24.

[Yen, 1999] Yen, J. (1999). Fuzzy logic: A modern perspective. *IEEE Transactions on Knowledge and Data Engineering,* 11(1).

[Yogesan y Schulerud, 1998] Yogesan, K., y Schulerud, H. (1998). Ultrastructural texture analysis as a diagnostic tool in mouse liver carcinogenesis. *Ultrastructural Pathology* 22:37-37.

[Yoshinari et al., 1993] Yoshinari, Y., Pedrycz, W. y Hirota, K. (1993). Construction of fuzzy models through clustering techniques. *Fuzzy Sets and Systems*, 54, pp. 157-165.

[Yuize et al., 1991] Yuize, H. Tagyu, T., Yoneda, M., Katoh, Y., Tano, S., Gabrisch, M. y Fukami, S. (1991). Decision suport system for foreing exchange trading – practical implementation. *Proceedings International Fuzzy Engineering Symposium (IFES'91),* Yokohama, pp. 971-982.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, 8, pp. 338-353.

[Zanonni y Reynolds, 1996] Zanonni, E. y Reynolds, R. (1996). Extracting design knowledge from genetic programs using cultural algorithms. En Fogel, L., Angeline, P. y Bäck, T. (eds.), *Proceedings of the Fifth Evolutionary Programming Conference*, San Diego, CA, Cambridge, MA. MIT Press, Cambridge, MA.

[Zou et al., 1997] Zou, K.H., Hall, W.J. y Shapiro, D.E. (1997). Smooth non-parametric receiver operating characteristic (ROC) curves for continuous diagnostic tests. *Statist Med*,16, pp. 2143-2156.

[Zweig y Campbell, 1993] Zweig, M.H. y Campbell, G. (1993). Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clin Chem*, 39, pp. 561-577.