Un algoritmo para la extracción automática de reglas lógicas a partir de modelos FIR

Félix Castro y Àngela Nebot

Llenguatges i Sistemes Informatics. Universitat Politècnica de Catalunya

Abstract. En este reporte se describe el algoritmo LR-FIR (Logical Rules with FIR), que tiene como objetivo extraer de manera automática un conjunto de reglas lógicas que expliquen el comportamiento del sistema. LR-FIR parte del modelo del sistema identificado mediante la metodología del Razonamiento Inductivo Difuso (FIR, por sus siglas en inglés). Un modelo FIR está compuesto de la máscara que describe la estructura del sistema y la base de reglas patrón que aglutina el comportamiento de éste. Este reporte está organizado en dos secciones. En la primera de ellas se presenta en la metodología FIR mientas que en la segunda se describe en detalle el algoritmo LR-FIR desarrollado.

1. Razonamiento inductivo difuso (FIR)

La conceptualización de la metodología Razonamiento Inductivo Difuso, FIR por sus siglas en inglés, proviene de una especialización de la Teoría General de Sistemas (GSPS) desarrollada por G. Klir [Klir, 1985]. FIR es una metodología de modelado y simulación cualitativa basada en el análisis del comportamiento del sistema en lugar del conocimiento de su estructura interna. FIR ha sido utilizada con éxito para el modelado de sistemas dinámicos difíciles de describir mediante técnicas de matemáticas clásicas, como las ecuaciones diferenciales, logrando obtener relaciones cualitativas entre las variables que componen el sistema y logrando predecir con mucha certeza su comportamiento futuro.

FIR realiza dos tareas principales. La primera es identificar las relaciones causales y temporales entre las variables del sistema para construir el modelo cualitativo del sistema observado. La segunda es predecir el comportamiento futuro del sistema a partir de las observaciones pasadas y del modelo previamente identificado. Para cumplir con estas tareas, la metodología FIR cuenta con cuatro funciones básicas mostradas en la figura 1.1: **fusificación, modelado cualitativo, simulación cualitativa y defusificación**.

El proceso de *fusificación* convierte las entradas cuantitativas del sistema en datos cualitativos difusos. En esta etapa un valor cuantitativo se convierte en una tripleta cualitativa, compuesta por un primer elemento que corresponde a la clase, el segundo al valor de pertenencia difusa y el tercero corresponde al valor del lado de la función de pertenencia. El valor del lado es la estrategia que utiliza FIR para garantizar que no se perderá información durante el proceso de fusificación, logrando saber en todo momento donde se encuentra el valor cualitativo, izquierda, centro o derecha del

máximo de la función de pertenencia. La etapa de *modelado cualitativo* es la encargada de encontrar relaciones causales y temporales entre las variables del sistema, con la finalidad de obtener el modelo que represente mejor al sistema en análisis. En la notación de FIR, un modelo está compuesto por la *máscara*, que corresponde a la estructura del sistema, y la *base de reglas patrón*, que almacena el comportamiento del sistema.

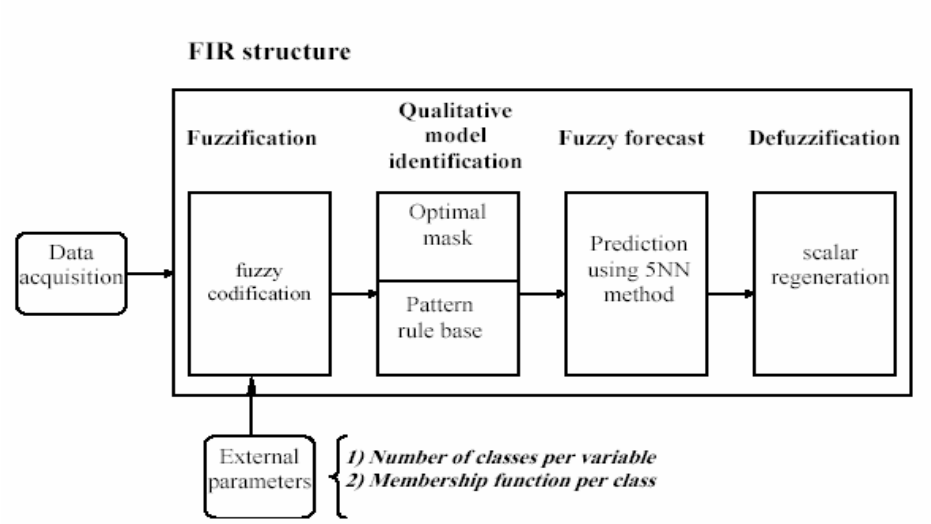

Figura 1.1. Etapas de la metodología FIR.

Una vez identificado el modelo que mejor representa al sistema es posible realizar el proceso de predicción del comportamiento futuro del sistema, utilizando el motor de inferencia de FIR. Este proceso es denominado *simulación cualitativa* y es una especialización de la técnica de los *k* vecinos más cercanos (k-Nearest Neighbors). La última etapa, denominada *defusificación*, es la operación inversa a la fusificación, en ésta etapa se convierte la salida cualitativa, predicha en la etapa anterior, en una variable cuantitativa. En los siguientes subapartados se detallan las etapas de la metodología FIR.

1.1. Fusificación

FIR toma como entrada los datos reales registrados del sistema. La función de fusificación es la encargada de convertir los valores cuantitativos del sistema (datos sin procesar) en sus valores cualitativos equivalentes (datos discretizados). En el proceso de codificación difusa un valor cuantitativo se convierte en una tripleta cualitativa, donde el primer elemento es el valor de la clase, el segundo el valor de pertenencia difusa y el último elemento corresponde al valor del lado. La clase corresponde al valor discretizado de la variable (etiquetas lingüísticas). El valor de pertenencia (membership) permite conocer en qué grado pertenece ese valor a la clase. El valor de lado permite conservar en la tripleta cualitativa el conocimiento completo del valor cuantitativo original, determinando con mayor precisión donde se encuentra el valor cualitativo, esto es, a la derecha, al centro o a la izquierda del máximo de la función de pertenencia. De esta forma no se pierde información durante el proceso de fusificación.

Para ejemplificar este proceso consideremos que tenemos un valor cuantitativo de 23 °C que indica la temperatura del medio ambiente. La discretización de esta variable se hace en tres clases cuyos valores cualitativos son: Fresco, normal y caliente. Cada una de estas etiquetas lingüísticas se representa por medio de una función de pertenencia, como se muestra en la figura 1.2. Así, 23°C corresponde a la clase "normal", con un valor de pertenencia de 0.755 y un valor de lado de "derecho", debido a que se encuentra en el lado derecho respecto al valor máximo de la función de pertenencia de la clase normal.

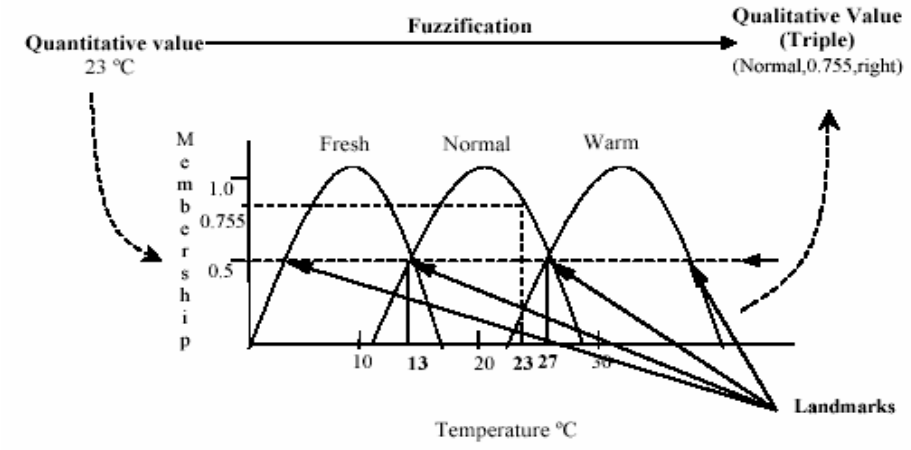

Fig. 1.2. Proceso de fusificación de FIR.

Una vez definidos los parámetros que describen el número de clases y los límites entre clases (landmarks) para cada variable, se lleva a cabo el proceso de fusificación de FIR sobre los datos registrados del sistema. Al finalizar este proceso se obtiene una tripleta de matrices con el mismo número de datos, la primera contiene los valores de clase, la segunda los valores de pertenencia y la tercera los valores de lado.

1.2. Modelado cualitativo

El comportamiento cualitativo del sistema está formado por las tres matrices obtenidas en el proceso de fusificación, en donde cada columna representa una de las variables observadas y los renglones son los registros realizados en cada intervalo de tiempo. En el proceso de modelado se realiza una búsqueda para encontrar el mejor modelo cualitativo que represente el comportamiento del sistema. Para ello se estudian las posibles relaciones causales y temporales que caracterizan el sistema. El comportamiento de éste puede ser pronosticado iterando sobre la matriz de transición de estados. Cuanto más determinista sea esta matriz, mayor será la probabilidad de que la predicción sea correcta.

La estructura del modelo cualitativo está definida por la *máscara*. Cada fila de la máscara representa un instante de tiempo correspondiente a un intervalo de muestreo. Al número de filas se le conoce como profundidad de la máscara. Los valores negativos representan las *m-entradas* o antecedentes, e indican que existe relación causal de éstas con el consecuente, o salida. El valor positivo corresponde a la *m-salida* o consecuente y los valores 0 significan que no existe ninguna relación causal

de éstos con el consecuente. Un ejemplo de una máscara se muestra en la ecuación 1.1.

$$
\begin{array}{lc}
t\backslash^{x} & \begin{array}{cccc} i_1 & i_2 & i_3 & o \end{array} \\
\begin{array}{l} t - 2\delta\, t \\ t - \delta t \\ t \end{array} & \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & -2 & 0 & 0 \\ -3 & 0 & 0 & +1 \end{pmatrix}
\end{array}
\qquad (1.1)
$$

La secuencia en que las m-entradas y la m-salida son numeradas no tiene ningún significado especial, aunque normalmente suelen ser numeradas de izquierda a derecha y de arriba a abajo. Los términos m-entrada (entrada de la máscara) y m-salida (salida de la máscara) son utilizados para evitar confusión con las entradas y las salidas del sistema. El número de elementos diferentes de cero en la máscara, indica el nivel de *complejidad* de ésta. Por ejemplo, en la ecuación *1.1,* la primera m-entrada $i_3(t-2\delta t)$ corresponde a la variable de entrada i_3 muestreada dos intervalos de tiempo anteriores al actual, la segunda m-entrada $i_2(t-\delta t)$ corresponde a la entrada del sistema i_2 muestreada en el instante de tiempo anterior al actual, etc. En FIR una máscara describe una relación dinámica entre las variables cualitativas del sistema.

La figura 1.3 muestra el proceso de conversión de relaciones dinámicas de los datos de entrada a las reglas patrón, utilizando la máscara de la ecuación 1.1. En la parte izquierda se muestra un fragmento de la matriz con los valores de clase, la primera de las tres matrices que forman el conjunto de datos cualitativos. En el ejemplo, se muestran las variables de entrada del sistema: u_1, u_2, y u_3 que se discretizaron en dos clases, mientras que la variable de salida y_1 fue discretizada en tres. El rectángulo punteado simboliza que la máscara va desplazándose hacia abajo, a través de la matriz de valores de clase. Los círculos de la máscara denotan las posiciones de las m-entradas, mientras que el cuadrado indica la posición de la m-salida. Los valores de clase se leen a través de los 'hoyos' de la máscara, y se colocan en forma consecutiva en la matriz entrada/salida. Como se puede apreciar en el lado derecho de la figura 1.3, cada renglón representa una posición de la máscara a través del recorrido que hace por la matriz de valores de clase. Estos valores se almacenan en la matriz de entrada/salida y se alinean con el último renglón de la máscara. Cada renglón de esta matriz representa un estado cualitativo pseudo-estático o **regla cualitativa basada en patrones**.

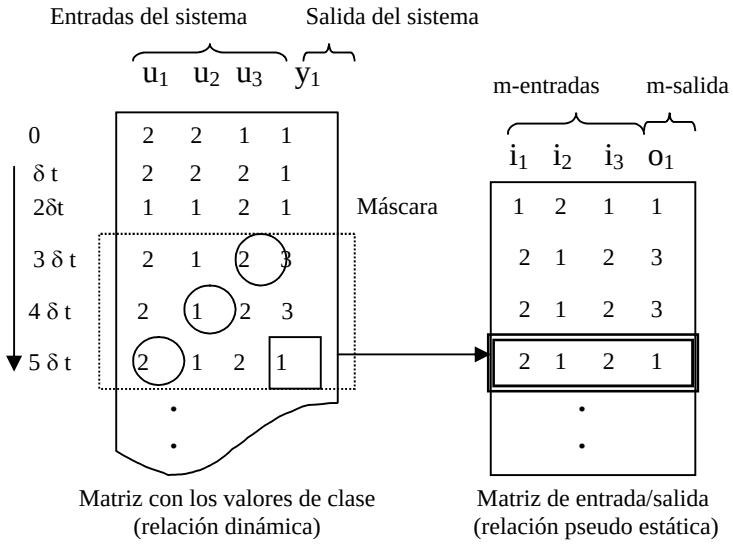

Figura 1.3. Proceso de identificación de la base de reglas patrón.

De la forma como se construye la matriz de transición de estados, un mismo patrón de entrada, llamado estado de entrada, puede generar diferentes valores de salida (estados de salida). Si la relación entre los estados de entrada y salida es no-determinista existirá incertidumbre en la predicción realizada. Por lo tanto, es conveniente tratar de obtener la matriz de transición de estados lo más determinista posible. En este sentido, se introdujo el concepto de máscara candidata con la finalidad de encontrar entre todo el conjunto de máscaras posibles (de cada complejidad hasta llegar a la máxima), la que mejor represente al sistema. A partir de la matriz de máscara candidata, el proceso de búsqueda de la máscara óptima procede a evaluar todas las posibles combinaciones a través de un mecanismo de búsqueda exhaustiva de complejidad exponencial o mediante estrategias de búsqueda utilizando algoritmos genéticos.

$$
\begin{array}{ll}
t\backslash^{x} & i_1 \quad i_2 \quad i_3 \quad o \\
t - 2\delta\, t & \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} \\
t - \delta t & \\
t &
\end{array}
\qquad (1.2)
$$

La máscara candidata contiene elementos “-1” donde tiene una m-entrada potencial, un elemento “+1” donde tiene la m-salida, y se utiliza el “0” para denotar relaciones prohibidas. Una posible máscara candidata para un sistema de cuatro variables podría ser aquella en la que se permita que FIR estudie todas las posibles relaciones causales entre las variables del sistema en los diferentes instantes de tiempo y la salida en el instante t, como se indica en la ecuación 1.2.

La optimalidad de la máscara se evalúa con respecto a su poder de predicción. La medida de entropía de Shannon se utiliza para determinar la incertidumbre asociada a la máscara evaluada. Para calcular la entropía total de la máscara, es necesario calcular la entropía de Shannon relativa a un estado de entrada, mediante la ecuación:

$$H_i = -\sum_{\forall o} p(o \mid i) \cdot \log_2 \, p(o \mid i) \tag{1.3}$$

donde $p(o \mid i)$ es la probabilidad condicional de que ocurra un cierto estado *o* en la m-salida, dado que ha ocurrido el estado *i* en el patrón de m-entradas. La entropía total de la máscara se calcula como:

$$H_m = \sum_{\forall i} p(i) \cdot H_i \tag{1.4}$$

donde $p(i)$ es la probabilidad de que ocurra ese estado de entrada. La entropía más alta posible H_{max} se obtiene cuando todas las probabilidades son iguales y una entropía cero corresponde a las relaciones totalmente deterministas.

El término reducción de entropía normalizada, H_r, se define como:

$$H_r = 1.0 - \frac{H_m}{H_{\max}} \tag{1.5}$$

H_r es un número real comprendido en el rango de 0.0 y 1.0, donde valores altos indican normalmente mayor poder de predicción. Las máscaras con valores de reducción de entropía altos generan predicciones con menor incertidumbre.

Es importante tener en cuenta la predictividad de la máscara que se está evaluando. En FIR se considera, desde el punto de vista estadístico, que cada estado se debe observar por lo menos cinco veces [Llaw and Kelton, 1990]. Por esta razón, se utiliza la razón de observación *Or*, como un factor adicional para determinar la calidad total de la máscara.

$$O_r = \frac{5 \cdot n_{5x} + 4 \cdot n_{4x} + 3 \cdot n_{3x} + 2 \cdot n_{2x} + n_{1x}}{5 \cdot n_{leg}} \tag{1.6}$$

Donde:

n_{leg} = número de estados legales de m-entradas;

n_{1x} = número de estados de m-entradas observados solamente una vez;

n_{2x} = número de estados de m-entradas observados dos veces;

n_{3x} = número de estados de m-entradas observados tres veces;

n_{4x} = número de estados legales de m-entradas observados cuatro veces;

n_{5x} = número de estados de m-entradas observados cinco o más veces.

Si todo estado legal de m-entradas ha sido observado al menos cinco veces, O_r es igual a 1.0 y no tiene ninguna influencia en la calidad de la máscara. Si no se ha observado ningún estado de m-entradas (es decir, no hay datos disponibles), O_r es igual a 0.0.

La calidad total de una máscara, Q_m, se define como el producto de su medida de reducción de entropía, H_r, y su razón de observación, O_r:

$$Q_m = H_r \cdot O_r \tag{1.7}$$

En este sentido, la máscara óptima es la que tiene una calidad, Q_m, mayor. Una vez identificada la máscara óptima, ésta es utilizada para obtener la base de reglas patrón por medio del proceso de conversión de patrones dinámicos a patrones pseudo-estáticos descrito anteriormente (ver figura 1.3). Como se mencionó anteriormente, la máscara en conjunto con la base de reglas patrón, representan el modelo FIR del sistema analizado. Para un estudio más detallado de la metodología FIR ver [Nebot, 1994].

1.3. Simulación cualitativa

El objetivo principal de la etapa de simulación es predecir los valores de la tripleta cualitativa (figura 1.2). Basándose en el modelo obtenido por FIR (la máscara óptima y la base de reglas patrón) se realiza el proceso de predicción de valores de clase, pertenencia y lado. Para la predicción se utiliza el motor de inferencia de FIR, que está basado en el algoritmo de los 5 vecinos más cercanos (5NN). En la figura 1.4 se ilustra el proceso de predicción.

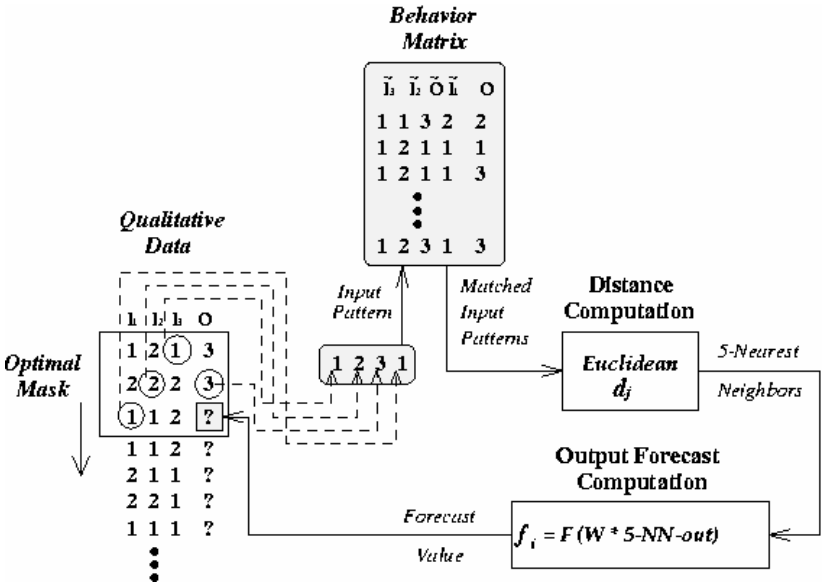

Figura 1.4: Proceso de simulación cualitativa en FIR.

Para el proceso de predicción la máscara es desplazada por los datos cualitativos. Se extraen los valores correspondientes a las m-entradas (círculos de la matriz "Qualitative Data" en la figura 1.4) formando un patrón de entrada estático. Se buscan en la base de reglas patrón ("behavior matrix" en la figura 1.4) todos aquellos patrones que coincidan con el patrón de entrada actual. Para todos ellos se calcula su distancia respecto al mismo y se conservan aquellos 5 que estén más cercanos. Los valores de la predicción se obtienen como una combinación ponderada de las salidas

de los 5 vecinos más cercanos seleccionados. Los valores cualitativos predichos (clase, pertenencia y lado) se almacenan en las matrices cualitativas de clase, pertenencia y lado. Una vez predicho un valor, la máscara se desplaza una posición hacia abajo para predecir el siguiente valor.

Antes de realizar la predicción del nuevo valor, se normaliza cada uno de los elementos del nuevo patrón de entrada. En FIR existen diferentes funciones de normalización, pero la más utilizada es la siguiente:

$$p_i = Side_i \cdot B \cdot \sqrt{\ln Memb_i} + 0.5 \tag{1.8}$$

donde $B = (4 \cdot \ln 0.5)^{-1/2}$. Las clases de los extremos se calculan de la siguiente forma: Para el extremo izquierdo se utiliza la ecuación:

$$p_i = C \cdot \sqrt{\ln Memb_i} \tag{1.9}$$

y para el extremo derecho:

$$p_i = 1 - C \cdot \sqrt{\ln Memb_i} \tag{1.10}$$

donde $C = (\ln 0.5)^{-1/2}$. Independientemente del rango de las señales originales, las señales p_i estarán entre 0.0 y 1.0.

Los valores p_i correspondientes a las diferentes variables de un estado de entrada son concatenados para formar el vector:

$$\mathbf{p} = [p_1, p_2, \ldots, p_j] \tag{1.11}$$

Asumiendo que el estado tiene *j* m-entradas. El vector **p** se denomina imagen normalizada del estado de entrada original.

El proceso continúa calculando las imágenes normalizadas para cada registro previo con el mismo espacio de entrada. Estos vectores se denominan $\mathbf{p_k}$. Cada vector $\mathbf{p_k}$ es diferente debido a que sólo coinciden los valores de clase del estado de entrada, pero no los valores de pertenencia ni los de lado.

Finalmente, se calcula la norma 2 de las diferencias entre el vector **p** que representa la imagen normalizada del nuevo patrón de entrada y los vectores $\mathbf{p_k}$ que corresponden a las imágenes normalizadas de los registros previos (reglas patrón) con el mismo estado de entrada, mediante la ecuación:

$$d_k = \|\mathbf{p} - \mathbf{p_k}\|_2 = \sqrt{\sum_{i=1}^{N} (p_i - p_{ik})^2} \tag{1.12}$$

Se identifican las 5 reglas patrón con menor valor en d_k. Las reglas identificadas corresponden a los cinco vecinos más cercanos, y serán utilizados para predecir el nuevo estado de salida. La contribución de cada vecino en la predicción del nuevo estado de salida, se determina en función a su proximidad. Esto se realiza dando un peso a cada vecino en función de su distancia. Si ninguna de las 5 distancias es igual a cero, los pesos absolutos se calculan utilizando la ecuación:

$$w_{abs_k} = \frac{(d_{\max}^2 - d_k^2)}{d_{\max} \cdot d_k} \tag{1.13}$$

donde *k* representa el vecino k-ésimo y $d_i \leq d_j, i < j$; $d_{\max} = d_5$. La ecuación 1.13 no funcionará si alguno de los vecinos tiene valor de 0.0, por lo que si uno de los vecinos tiene una distancia de 0.0, el peso de ese vecino será 1.0 y el peso de los demás vecinos será 0.0.

Utilizando los valores absolutos de los pesos de los 5 vecinos más cercanos, se calculan los pesos relativos mediante la formula:

$$w_{rel_k} = \frac{w_{abs_k}}{\sum_{\forall k} w_{abs_k}} \tag{1.14}$$

Los pesos relativos serán números entre 0.0 y 1.0, y su suma siempre será 1.0, por lo que se pueden interpretar como porcentajes. La predicción del nuevo estado de salida, en el espacio normalizado, se calcula como una suma ponderada de los estados de salida de los cinco vecinos más cercanos. De esta manera, el estado cualitativo de salida puede obtenerse utilizando la fórmula:

$$State_{out_{new}} = \sum_{\forall k} w_{rel_k} \cdot (Class_{out_k} + p_{out_k}) \tag{1.15}$$

donde $Class_{out_k}$ y p_{out_k} son la clase y la función de pertenencia normalizada de los valores de salida de los 5 vecinos más cercanos, respectivamente. El valor de predicción de clase se obtendrá mediante la ecuación:

$$Class_{out_{new}} = IFIX(State_{out_{new}}) \tag{1.16}$$

y el valor de predicción de pertenencia y de lado se obtiene mediante el inverso de la función de normalización anterior a partir de:

$$p_{out_{new}} = State_{out_{new}} - Class_{out_{new}} \tag{1.17}$$

1.4. Defusificación

En la etapa de defusificación los valores cualitativos predichos en la fase previa, se convierten en datos cuantitativos. La defusificación representa el proceso inverso de la fusificación.

Para validar el modelo se comparan los resultados de predicción con los datos reales del sistema. Para calcular el error de las predicciones se puede utilizar el error cuadrático medio (en términos de porcentaje):

$$MSE2 = \frac{E\left[(y(t) - \hat{y}(t))^2\right]}{y_{\text{var}}} \cdot 100\% \tag{1.18}$$

donde y_{var} es la varianza definida como:

$$y_{\text{var}} = E\left[y^2(t)\right] - \left\{E\left[y(t)\right]\right\}^2 \tag{1.19}$$

donde y es el valor real y $\hat{y}$ es el valor de predicción.

2. Algoritmo para la extracción de reglas lógicas: LR-FIR

En esta sección se describe el algoritmo de extracción de reglas lógicas diseñado e implementado.

Primero se muestran los pasos del algoritmo y sus características, posteriormente se describe en detalle cada uno de ellos. En la figura 2.1 se presentan de forma esquemática las fases del algoritmo.

1. **Compactación básica**. Es un proceso iterativo que evalúa una a la vez todas las reglas patrón generadas por FIR, *R*. En esta etapa se compactan las reglas patrón basándose en el "conocimiento" existente en las reglas patrón generadas por FIR. Un subconjunto de reglas, R_c, puede ser compactado en una regla simple, r_c, cuando todas las premisas, *P*, excepto una, P_a, así como el consecuente, *C*, tienen los mismos valores. En el contexto de este trabajo, las premisas se corresponden con las m-entradas y el consecuente con la m-salida. Si R_c contiene todos los valores legales, LV_a, de P_a, las reglas candidatas, R_c, pueden ser reemplazadas por la regla compacta, r_c, que tiene un valor de -1 en la premisa P_a. Cuando existe más de un valor -1, P_{ni}, en la regla compacta, es necesario evaluar ésta en busca de conflictos con las reglas patrón originales, *R*. Para buscar conflictos se debe expandir todas las premisas, P_{ni}, a todos sus valores legales, LV_a, y comparar las reglas resultantes, *Xr*, con las reglas originales, *R*. Si existen conflictos, *Cf*, la regla compacta, r_c, es rechazada y contrariamente aceptada. Si se acepta, la regla compacta, r_c sustituye al conjunto de reglas, R_c. Los conflictos ocurren cuando una o más reglas expandidas, *Xr*, tienen los mismos valores en todas las premisas, *P*, pero diferente valor en el consecuente, *C*.

2. **Compactación mejorada**. En la compactación básica, sólo se estructura y representa el conocimiento disponible en una forma más compacta. En el paso de compactación mejorada, se extiende la base de conocimiento *R*, a casos que no se utilizaron para construir el modelo: R_b. En resumen, la compactación básica llega a una base de conocimiento compacta que sólo contiene conocimiento existente en el sistema, mientras que la compactación mejorada contiene conocimiento no existente o creencias. Para realizar la compactación

mejorada se tienen 2 opciones: En la primera opción, utilizando las reglas obtenidas en la compactación básica, *R'*, en un proceso iterativo, todas las premisas, *P*, que tienen valores no negativos (premisas no compactadas), en todas las reglas, *r*, son reemplazados por valores -1; Se expande la regla a todos sus valores legales, LV_a, y se comparan las reglas resultantes, *Xr*, con las reglas originales, *R*. Si no existen conflictos la regla se acepta y contrariamente se rechaza. La opción 2 es una extensión de la compactación básica, donde para compactar un subconjunto de reglas, R_c, en una regla compacta, r_c, debe existir un porcentaje mínimo, *MR*, pero razonable de los valores legales, LV_a, en las reglas candidatas, R_c. La opción 2 parece ser más razonable, ya que aunque se asumen creencias, el porcentaje de éstas es mínimo, por lo que no se compromete el modelo previamente identificado por FIR. En la opción 1 las creencias se asumen siempre y cuando sean consistentes con las reglas originales, por lo que podrían no concordar con el modelo identificado, especialmente cuando los datos utilizados para obtener el modelo (datos de entrenamiento) son pobres y no describen adecuadamente todo el comportamiento del sistema.

3. **Eliminación de reglas duplicadas**. En este paso se eliminan reglas duplicadas que se pudieran generar durante la etapa de compactación mejorada.

4. **Eliminación de reglas en conflicto**. En esta fase se eliminan las reglas en conflicto o "ambiguas". Como se comentó anteriormente, un conflicto ocurre cuando se tiene el mismo valor (clase) en todas las premisas, *P*, pero diferente valor en el consecuente, *C*. Por ejemplo, las reglas: 1 2 1 3 y 1 2 1 1, tienen el mismo valor en las premisas (1 2 1) y diferente valor en el consecuente (3 y 1, respectivamente). Estas reglas son ambiguas, por lo que se debe eliminar una de ellas para no causar conflictos y confusión al momento de evaluar el sistema en análisis. La regla que se elimina se escoge en función de las métricas de *especificidad* y *sensitividad* que se explicarán más adelante. Es importante aclarar que cuando se tienen reglas con valores del consecuente consecutivos, no se debe considerar como un conflicto, ya que comparten espacios de entrada contiguos por lo que se deben unificar en el paso posterior del algoritmo. Si se tienen las reglas: 1 2 1 1 y 1 2 1 2, aparentemente parece ser un conflicto, *Cf*, sin embargo, en el algoritmo se consideran como 2 reglas que se deben unificar en una única regla: 1 2 1 (1-2) y cuya interpretación es: si se tienen valores de clase 1 2 y 1 para las premisas respectivamente, el consecuente estará en el rango de las clases 1 y 2 de la salida.

5. **Unificación de reglas**. En la etapa de unificación de reglas, se "unifican" reglas que comparten espacios de entrada contiguos en alguna variable (premisas y consecuente) y que tengan el mismo valor en el resto de variables. Por ejemplo, las reglas 1 2 1 2 y 1 2 2 2, comparten en la tercera variable espacios de entrada contiguos (1 y 2), por lo que se pueden unificar en la regla: 1 2 (1-2) 2.

6. **Evaluación de las reglas obtenidas**. En este paso se evalúan las reglas obtenidas en la etapa anterior, utilizando los datos reales del sistema, tanto de entrenamiento (training) como de prueba (test). Las reglas obtenidas se evalúan utilizando métricas estándares de evaluación de clasificadores basadas en la "matriz de confusión". Las métricas de evaluación utilizadas permiten

valorar objetivamente las reglas, independientemente de la distribución de los datos de cada clase.

7. **Eliminación de reglas de baja calidad**. Como última etapa del algoritmo de extracción de reglas se eliminan aquellas reglas de baja calidad, es decir, con valores bajos de las métricas de evaluación, *especificidad* y *sensitividad*, descritas en el apartado 2.6.

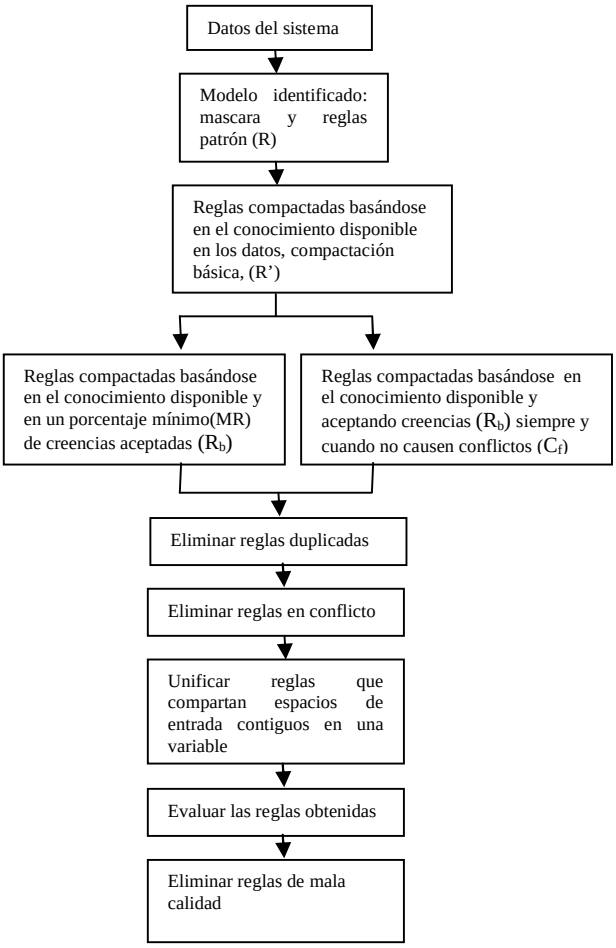

Fig. 2.1. Pasos ordenados del algoritmo LR-FIR.

2.1. Algoritmo de compactación básico

Este algoritmo fue inicialmente propuesto por Medina y Cellier como una alternativa a los métodos de compactación de reglas utilizados en algebra booleana, con la ventaja que puede compactar reglas con atributos multi-valuados.

Es importante aclarar que el algoritmo es dependiente del número de variables y reglas patrón, por lo que, cuanto mayor sean éstos, mayor será el tiempo que el algoritmo tardará en compactar las reglas originales. Sin embargo, las reglas ya compactadas son mucho más intuitivas y entendibles que las originales. Las reglas compactadas describen patrones de comportamiento del sistema analizado, identificando las variables que son realmente importantes en la predicción de los datos de salida que correspondan a esos patrones. A continuación se presentan los pasos del algoritmo, posteriormente se presenta un ejemplo que ilustra el funcionamiento del mismo.

Descripción del algoritmo

El algoritmo toma como entrada las reglas patrón, *R*, generadas por FIR, y ejecuta un proceso iterativo donde convierte, *R*, en reglas compactas, *R'*. Las reglas compactadas tendrán un valor de -1 en aquellas variables que no son importantes, denotando condiciones no relacionadas con la variable de salida del sistema. El algoritmo es un proceso iterativo que extrae todas las reglas, R_c, que coinciden con la i-esima regla, r_i, en todas las variables (*P* y *C*), excepto en la j-esima variable, v_j, y si existen todos los valores legales, LV_a, se sustituye R_c por r_c.

Los pasos más importantes del algoritmo son:

1. Obtener las reglas que coinciden con la regla r_i en todas las variables excepto en la variable v_j. Es importante aclarar que si existe un valor -1 en alguna de las variables no actuales (diferente a v_j), éste se considera igual a los valores restantes, ya que significa que puede tomar cualquier valor del espacio de entrada de la variable. Por ejemplo, si se tienen las reglas:

No. Regla	Entradas			Salida
1	0	2	0	1
2	-1	2	1	1

   y v_j = 3, estas reglas se deben compactar en: **-1 2 -1 1**, siempre y cuando no exista conflicto con las reglas originales, *R*, al expandir la regla compactada.

2. Si las reglas obtenidas, R_c, cubren todo el espacio de entrada de la variable v_j, entonces, las reglas son candidatas a sustituirse por una regla, r_c, que tendrá el valor de -1 en la variable v_j, mientras que el resto de valores de las variables permanece igual. Por ejemplo, si la variable 3 fue discretizada en 3 clases, para poder compactar las reglas en **1 1 -1 1**, es necesario que existan las RP originales: **1 1 0 1**, **1 1 1 1** y **1 1 2 1**. Si alguna de las reglas patrón anteriores no existe, no se podría compactar ya que no estaría cubierto todo el espacio de entrada de la variable v_j.

3. Si la regla compactada tiene más de un valor -1, entonces ésta se expande a todas las reglas posibles, *Xr*, basándose en la discretización de las variables utilizada por FIR. Si en las reglas expandidas no existe ningún conflicto, *Cf*, con las reglas patrón originales, *R*, la regla se toma como válida y reemplaza a las reglas obtenidas en el paso 1, en caso contrario se rechaza, debido a que es incompatible con el modelo del sistema en análisis.

4. Repetir los pasos anteriores desde la regla r_{i+1} hasta r_n, donde *n* es el total de reglas.

5. Repetir los pasos anteriores desde la variable v_{j+1} hasta v_m, donde *m* es el número de m-entradas del sistema.

Ejemplo

En este apartado se presenta un ejemplo que ilustra el funcionamiento del algoritmo de compactación básico. El ejemplo consiste en 3 variables de entrada y 1 de salida, discretizadas en 2, 3, 2 y 2 clases, respectivamente. El conjunto de reglas patrón inicial es:

No. Regla	Entradas			Salida
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	0	2	0	1
5	0	2	1	1
6	1	0	1	0
7	1	1	0	1
8	1	2	1	1
9	1	1	1	1

Como se mencionó anteriormente, la ejecución del algoritmo se inicia con la regla #1 y la m-entrada #1. Las reglas 1 y 6, son candidatas a compactarse ya que coinciden en todas las variables excepto en la primera:

No. Regla	Entradas			Salida
1	0	0	1	0
6	1	0	1	0

Dado que la variable 1 fue discretizada en 2 clases, el subconjunto de reglas cubre todo el espacio de entrada de la variable, por lo que se puede compactar, dando lugar a la regla:

No. Regla	Entradas			Salida
1a	-1	0	1	0

Debido a que sólo existe una variable con valor de -1, no es necesario expandir la regla para evaluar conflictos, por lo que la regla compacta se toma como válida y reemplazará a las reglas originales 1 y 6.

Siguiendo el algoritmo, las reglas 3 y 9; y 5 y 8, se sustituyeron por su correspondiente regla compacta, y las reglas resultantes al aplicar el algoritmo de compactación para la m-entrada # 1 se muestran a continuación.

No. Regla	Entradas			Salida
1	-1	0	1	0
2	0	1	0	0
3	-1	1	1	1
4	0	2	0	1
5	-1	2	1	1
6	1	1	0	1

Durante la ejecución del algoritmo para la m-entrada # 2, las reglas 3 y 5 coinciden en todas las variables excepto en la 2, sin embargo, dicha variable fue discretizada en 3 clases, por lo que el subconjunto de reglas no cubre todo el espacio de entrada de la variable y no es posible realizar el procedimiento de compactación. No fue posible realizar ninguna compactación en la m-entrada # 2, por lo que las reglas resultantes son las mismas de la tabla anterior.

En el funcionamiento del algoritmo para la m-entrada (premisa) 3, las reglas 3 y 6 son candidatas a compactarse ya que cubren todo el espacio de entrada de la variable 3 (fue discretizada en 2 clases):

No. Regla	Entradas			Salida
3	-1	1	1	1
6	1	1	0	1

La regla candidata a sustituir a las reglas 3 y 6 es:

No. Regla	Entradas			Salida
3a	-1	1	-1	1

Como se menciona en el paso 3 del algoritmo, cuando existe más de un valor -1 en una regla candidata a compactarse, r_c, ésta se debe expandir a todas las posibles reglas, *Xr*. Basándose en la discretización utilizada para las variables, r_c se expande a todos los estados legales, LV_a, de la variable y se evalúan conflictos con las reglas patrón originales. Las reglas resultantes, *Xr*, de la expansión son:

No. Regla	Entradas			Salida
3a	**0**	**1**	**0**	**1**
3b	0	1	1	1
3c	1	1	0	1
3d	1	1	1	1

Debido a que la regla 3a, está en conflicto con la regla original #2, ya que ambas tienen el mismo valor en las premisas pero diferente valor en el consecuente, la regla compactada no debe sustituir a las reglas 3 y 6 del paso de evaluación de la premisa 3.

Siguiendo con el algoritmo en la compactación de reglas para la premisa # 3, las reglas 4 y 5 son candidatas a compactarse ya que cubren todo el espacio de entrada de la variable y debido a que no existe ningún conflicto con las reglas originales, la regla compacta es: **-1 2 -1 1**.

Debido a que se han evaluado todas las m-entradas del sistema en análisis, el algoritmo de compactación básico finaliza su ejecución y el resultado final se muestra en la siguiente tabla.

No. Regla	Entradas			Salida
1	-1	0	1	0
2	0	1	0	0
3	-1	1	1	1
4	-1	2	-1	1
5	1	1	0	1

2.2. Algoritmo de compactación mejorado

Como se mencionó anteriormente, existen 2 opciones a elegir para realizar una compactación mejorada de las reglas, *R'*, generadas por el algoritmo de compactación básico: *Compactación Mejorada 1*, Propuesta por Medina y Cellier y *Compactación Mejorada 2*, propuesta en este trabajo de investigación.

En la compactación mejorada se asumen "creencias" para la generación de reglas compactas, a diferencia del algoritmo básico, que para compactar un subconjunto de reglas, debe existir todo el "conocimiento" suficiente en las reglas patrón originales, *R*. En la compactación mejorada se "asume" conocimiento inexistente (creencias) en los datos originales, siempre y cuando sea "compatible" (no existan conflictos) con el conocimiento existente en los datos originales.

En el algoritmo de Compactación Mejorada 1, una vez que se ha aplicado el algoritmo básico a las reglas originales, *R*, se cambia al valor -1, una a la vez, cada una de las premisas (m-entradas) de cada regla con valor diferente a -1 y se extiende la regla (generar todas las reglas posibles con la regla compacta). Se evalúa si existen conflictos, *Cf*, con las reglas patrón originales. Si no existen conflictos, la regla (modificada) es considerada válida, aunque no existan todas las clases de las variables compactadas. En caso contrario, la regla compactada se deshecha y se continúa con la siguiente variable diferente a -1, y así sucesivamente hasta finalizar con todas las premisas y todas las reglas.

En el contexto del algoritmo de compactación de reglas presentado aquí, un conflicto existe cuando existen reglas con los mismos valores en todas las premisas, pero diferente valor en el consecuente. A continuación se muestra un ejemplo donde existe conflicto entre 2 reglas:

No. Regla	Entradas			Salida
1	2	1	3	1
2	2	1	3	3

Las reglas 1 y 2 están en conflicto debido a que tienen el mismo valor en las variables de entrada y diferente valor en la salida.

El algoritmo de Compactación Mejorada 2 funciona de la misma forma que el algoritmo de compactación básico, con la diferencia que se incluye una métrica, *InfClases*, que representa la influencia de las clases presentes en los datos analizados, con respecto a la totalidad de clases que pueden existir para dicha variable (discretización inicial). Para decidir si un subconjunto de reglas, R_c, se debe compactar, se define un porcentaje de influencia "mínimo" de las clases de cada variable, *MR*, y el número mínimo de clases, *Ncm*, que debe tener la variable para aplicar la métrica. Por ejemplo, si se define una influencia mínima de las clases de 0.75 (75%) y un mínimo de clases de las variables de 3, podrán compactarse todas las reglas que cubran más del 75% del espacio de entrada de dicha variable, si ésta fue discretizada en 3 o más clases, por el contrario, si la variable fue discretizada en menos de 3 clases, para compactarse las reglas deben cubrir el 100% del espacio de entrada de la variable.

Para ejecutar la compactación mejorada 2, es necesario que el usuario proporcione el valor mínimo de la métrica, *MR*, y el número mínimo de clases, *Ncm*, de las variables.

En la implementación actual se permite al usuario realizar experimentos con diferentes valores de la métrica para obtener de manera experimental el valor óptimo. Un valor óptimo para un conjunto de datos puede no serlo para otro, por lo que es necesario realizar investigaciones más detalladas que permitan definir valores "óptimos" de la métrica para diferentes contextos. La métrica propuesta se calcula con la formula siguiente:

InfClases = NumClasExist / NumTotClasVarAct,

donde:

NumClasExist = Numero clases encontradas en los datos, y

NumTotClasVarAct = Total de clases en que se discretizó la variable.

Es importante realizar una serie de pruebas para determinar el valor óptimo de la métrica anterior, ya que cuando la discretización de las variables es poco granular, es decir, el número de clases es pequeño, no tiene mucho sentido utilizarla. Sin embargo, cuando es grande, puede apoyar en el proceso de obtención de reglas más realistas. Una primera idea es que cuando las variables sean discretizadas en un número de clases igual o menor a 3, se compactarán las reglas si el valor de la métrica es 1, es decir, que existan todas las clases, en el caso contrario, el valor de la métrica puede ser mayor o igual a 0.75. La finalidad de esto es tener reglas lógicas más realistas y que no abarquen un rango muy grande del total del espacio de entrada de la variable.

2.3. Eliminación de reglas duplicadas

Al aplicar el algoritmo de compactación mejorado, pueden existir reglas duplicadas, R_d, debido a que sólo se evalúan conflictos con las reglas originales, *R*. Por ejemplo, si al aplicar el algoritmo básico, se generan las reglas: **1 -1 2 1** y **1 -1 3 1**, suponiendo que al aplicar el algoritmo mejorado en la tercera variable de ambas reglas no se genera conflicto con las RP originales, el resultados sería: **1 -1 -1 1** y **1 -1 -1 1**, lo que generaría 2 reglas duplicadas, ya que el algoritmo extendido únicamente evalúa la existencia de conflictos y no de duplicidad.

2.4. Eliminación de reglas en conflicto

Durante los pasos previos del algoritmo es posible encontrar reglas "ambiguas" o en conflicto, es decir, que tienen el mismo valor en todas las premisas pero valor diferente en el consecuente. Por ejemplo, las reglas: **1 2 1 3** y **1 2 1 1**, tienen el mismo valor en las premisas (1 2 1) y diferente valor en el consecuente (3 y 1). Estas reglas son ambiguas, por lo que se debe eliminar una de ellas y no causar conflicto y confusión al momento de evaluar el sistema en análisis. Las métricas que se utilizan para eliminar reglas en conflicto son la *especificidad* y la *sensitividad* (ver sección 2.6). Es importante aclarar que cuando se tienen reglas con valores de la variable de salida consecutivos no se debe considerar como un conflicto, ya que comparten espacios de entrada contiguos.

2.5. Unificación de reglas

El paso de unificar reglas se lleva a cabo en 2 etapas. En una primera fase se unifican reglas que comparten, en una misma variable, espacios de entrada contiguos y el mismo valor en las variables restantes. En la segunda etapa se evalúan las reglas unificadas en la primera fase y las reglas no unificadas, en busca de nuevas unificaciones.

Ejemplo

Un ejemplo de unificación de reglas en la primera etapa sería el siguiente. Si se tienen las reglas:

2	2	2	1
1	2	1	2
2	1	2	2
3	1	2	2
1	2	1	1
1	3	1	1

Se pueden unificar la tercera y cuarta en: **(2-3) 1 2 2**. La quinta y sexta regla se convierten en: **1 (2-3) 1 1**. El resultado final de la primera etapa sería:

Reglas no unificadas:

2	2	2	1
1	2	1	2

Reglas unificadas:

(2-3)	1	2	2
1	(2-3)	1	1

Basándose en las reglas resultantes de la primera fase de unificación, es posible unificar, en la segunda etapa, la segunda regla no unificada con la segunda unificada, debido a que comparten espacios de entrada contiguos en la cuarta variable. El resultado sería el siguiente:

Reglas no unificadas:

2	2	2	1

Reglas unificadas:

(2-3)	1	2	2
1	(2-3)	1	(1-2)

2.6. Evaluación de reglas

El objetivo del paso actual es evaluar las reglas obtenidas, basándose en los datos reales del sistema. Con esta finalidad se utilizan métricas estándares que permiten

obtener una evaluación objetiva y realista, independientemente de la distribución de los datos de cada clase. Estas métricas son las siguientes [Bradley, 1997]:

Sensitividad = TP / (TP + FN)

Especificidad = TN / (TN + FP)

Valor Predictivo Positivo (PPV) = TP / (TP + FP)

Donde:

TP ("true positive") = Número de datos que la regla predice que pertenece a la clase *x*, y que realmente pertenecen a la clase *x*.

FN ("false negative") = Número de datos que la regla predice que no pertenece a la clase *x*, y que realmente pertenecen a la clase *x*.

FP ("false positive") = Número de datos que la regla predice que pertenece a la clase *x*, y que realmente no pertenecen a la clase *x*.

TN ("true negative") = Número de datos que la regla predice que no pertenece a la clase *x*, y que realmente no pertenecen a la clase *x*.

2.7. Eliminación de reglas de mala calidad

Para evitar tener reglas de mala calidad, se eliminan aquellas reglas que no cumplan con un valor mínimo en las métricas descritas en el paso anterior. Actualmente, éste valor es determinado por el analista del sistema. Sin embargo, es necesario realizar un estudio específico que nos ayude a determinar el valor óptimo.

Actualmente se está evaluando este algoritmo en aplicaciones de diferente ámbito. Concretamente, en el campo de la educación a distancia y en el área de la contaminación ambiental.

Referencias

[Bradley, 1997] Bradley, P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7), (1997) 1145-1159.

[Klir, 1985] Klir, G.: Architecture of Systems Problem Solving. Plenum Press. New York (1985).

[Llaw and Kelton, 1990] Llaw A. and D. Kelton.: Simulation Modeling and Analysis 2nd Edition. McGraw-Hill, New York (1990).

[Nebot, 1994] Nebot, A.: Qualitative Modeling and Simulation of Biomedical Systems Using Fuzzy Inductive Reasoning. Ph.D. thesis, Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya. Barcelona, Spain (1994).