Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería de Sistemas de Comunicaciones

Trabajo Fin de Grado

Clasificadores neuronales para problemas binarios desequilibrados

Autora: Eva Pérez Íñigo
Tutor: Marcelino Lázaro Teja

Julio 2016

Trabajo Fin de Grado
Clasificadores neuronales para problemas binarios desequilibrados

Autora
Eva Pérez Íñigo

Tutor
Marcelino Lázaro Teja

La presentación del presente Trabajo Fin de Grado se realizó el día 6 de julio de 2016, siendo calificada por el siguiente tribunal:

Presidenta: Carmen Peláez Moreno

Secretario: José Francisco Algorri Genaro

Vocal: José Carlos Pulido Pascual

y habiendo obtenido la siguiente calificación:

Calificación:

Leganés, julio de 2016

Resumen

El presente Trabajo Fin de Grado se encuadra en el marco general de la Inteligencia Artificial (IA), en particular en el ámbito del aprendizaje máquina. La IA pretende dotar a las máquinas de la capacidad de solucionar problemas a través del paradigma de la inteligencia humana.

En concreto, este trabajo trata de la aplicación del aprendizaje máquina a la clasificación de patrones en problemas de clasificación binaria, que consiste en distinguir entre patrones de dos clases diferentes. Los métodos de aprendizaje máquina "*aprenden*" a resolver el problema de clasificación a partir de un conjunto de ejemplos etiquetados (conjunto de patrones con indicación de la clase a la que pertenece cada patrón). Este conjunto de ejemplos habitualmente se denomina conjunto de entrenamiento. Dentro de la clasificación binaria, se considerarán problemas desequilibrados o desbalanceados, que son aquellos en los que el número de patrones disponibles correspondientes a cada una de las dos posibles clases es sensiblemente diferente. Estos problemas son de gran importancia, ya que hay un gran número de aplicaciones con estas peculiaridades, como por ejemplo la detección de fraude (las operaciones fraudulentas son muchas menos que las legales) o la diagnosis médica de alguna enfermedad (el número de pacientes sanos es mucho mayor que el de enfermos). Además, como sucede en estos ejemplos, en muchas ocasiones el objetivo más importante es precisamente la detección de patrones de la clase minoritaria.

La utilización de métodos de aprendizaje máquina en este tipo de problemas tiene como dificultad potencial que los ejemplos de la clase mayoritaria pueden dominar en el aprendizaje y ocultar los ejemplos de la clase minoritaria. Para evitar este posible efecto es necesario tomar medidas que equilibren la aportación en el aprendizaje de las muestras correspondientes a las dos clases.

En el trabajo, en primer lugar se ha obtenido un conjunto de bases de datos reales correspondientes a problemas de clasificación binaria y con datos desbalanceados. Las bases de datos elegidas corresponden a problemas reales que han sido tratados en la literatura utilizando otros métodos de clasificación. Para resolver estos problemas de clasificación, se han utilizado redes neuronales artificiales, en concreto, perceptrones multicapa. Se han considerado varias alternativas para tener en cuenta el desequilibrio de los datos. Por un lado, se han utilizados dos funciones de coste para el aprendizaje de la red neuronal que tienen en cuenta el diferente número de muestras de cada clase: la primera es una función de coste basada en el error cuadrático medio ponderado; la segunda es una función de coste basada en el riesgo de Bayes. Por otro lado se han utilizado combinadores de clasificadores, que ya han demostrado en la literatura que pueden ser útiles en este tipo de problemas. Los clasificadores obtenidos se han evaluado utilizando varias figuras de mérito, y se han comparado las prestaciones obtenidas con los distintos

métodos considerados en cada una de las bases de datos.

Para realizar la evaluación de los clasificadores, se ha seguido la metodología habitual empleada cuando se utilizan métodos de aprendizaje máquina. Cada base de datos se divide en dos conjuntos de patrones: conjunto de entrenamiento y conjunto de test. Los parámetros del clasificador se obtienen a partir de los patrones del conjunto de entrenamiento. Una vez obtenidos estos parámetros, o lo que es lo mismo, una vez diseñado el clasificador, las prestaciones del mismo se evalúan utilizando el conjunto de test, cuyos patrones no se utilizaron en el procedimiento de aprendizaje.

Finalmente, a la vista de los resultados obtenidos con cada uno de los métodos considerados, se discute sobre las principales conclusiones extraidas a partir de dichos resultados.

Índice General

Lista de Figuras

Lista de Tablas

Capítulo 1

Motivación y objetivos

En este primer capítulo se van a presentar los aspectos que han motivado la realización de este trabajo, teniendo en cuenta el entorno socio-económico en que se ubica y el marco regulador aplicable. Además se describirán los objetivos planteados para este Trabajo Fin de Grado.

1.1. Motivación del trabajo

Este trabajo se encuentra situado en el marco general de la Inteligencia Artificial (IA), y en particular dentro del ámbito del aprendizaje automático o aprendizaje máquina. La IA pretende dotar a las máquinas de la capacidad de solventar problemas a través del paradigma de la inteligencia humana. Para ello se utilizan distintas herramientas estadísticas y computacionales.

La IA puede dividirse en dos ramas. La primera, denominada inteligencia artificial convencional o IA simbólica-deductiva se basa en el enfoque estadístico del comportamiento humano. La segunda se denomina inteligencia artificial computacional o IA subsimbólica-inductiva, y se basa en los datos empíricos.

La estadística y la informática han ido avanzando y desarrollando nuevas técnicas desde hace décadas. Sin embargo antes no se disponía de grandes cantidades de datos. Esto ha cambiado con Internet y la capacidad de almacenamiento de datos actual. Junto con Internet, a día de hoy no para de aumentar el número de sensores que ocupan nuestro mundo, esto produce una cantidad de información enorme. Gracias a ello es posible mejorar el aprendizaje de las máquinas. Además con la situación actual, no dejan de desarrollarse nuevos y mejorados mecanismos, y es posible solucionar problemas con mayor nivel de abstracción y complejidad.

Muchas empresas utilizan ya el aprendizaje máquina para mejorar sus

decisiones. Algunas como BlackRock (Williams, 2015) utilizan datos procedentes de Google y Twitter para tomar decisiones sobre sus inversiones. También es utilizado para la lucha contra el fraude, con la utilización de métodos como las redes neuronales, los árboles de decisión y los análisis bayesianos (BBVAOpen4U, 2015). Lo que permite ahorrar grandes cantidades de dinero.

Las empresas más potentes del mundo como Google, Apple, Facebook, Microsoft, etc. (Rodríguez, 2016) llevan años invirtiendo dinero en investigación para la realización de aplicaciones basadas en la IA. Algunos ejemplos son Siri de Apple o Cortana en Microsoft, cuya función de asistente virtual se basa en técnicas de reconocimiento de voz, búsqueda de contenidos y demás aspectos relacionados con la IA. Además con las nuevas técnicas de aprendizaje profundo (*deep learning*), el sistema RankBrain (Hof, 2015) ha logrado mejorar las prestaciones del buscador de Google en un 10 % (Palazuelos, 2015).

En el ámbito de la automoción, el desarrollo de vehículos inteligentes, con capacidades de conducción autónoma, es otro ejemplo del uso intensivo de la IA y del aprendizaje máquina. La inversión en el desarrollo de estas tecnologías en los últimos años ha sido enorme, no sólo por parte de los fabricantes, sino también de las administraciones, ya que su implantación puede ayudar a salvar vidas, y mejorar la fluidez del tráfico. Por citar un ejemplo, el gobierno de Estados Unidos propuso el pasado enero una inversión de 4 billones [1] de dolares durante los próximos 10 años para la incentivación de los vehículos autónomos (Spector y Ramsey, 2016).

Fuera del ámbito económico, estas técnicas también ayudan en aspectos tan críticos como salvar vidas. En la medicina el uso de herramientas de IA ayuda a la detección de enfermedades. Por ejemplo, en los últimos años se han desarrollado distintas aplicaciones o juegos para etiquetar datos médicos. Algunos juegos como TuberSpot (Luengo-Oroz, 2015) que consisten en encontrar las bacterias que provocan la tuberculosis, o MalariaSpot (Salas, 2013) cuyo objetivo es 'atrapar' los parásitos de muestras de sangre reales. Esto ayuda a obtener datos etiquetados, cuyos resultados promediados sobre todos los jugadores han resultado ser altamente satisfactorios.

También se ha observado un enorme interés en el desarrollo de herramientas de ayuda a la diagnosis médica basadas en aprendizaje máquina (Kononenko, 2001) (Foster et al., 2014), con aplicaciones tan diversas como análisis de señales de electrocardiograma (para detección de arrítmias (Moavenian y Khorrami, 2010), daños en válvulas cardíacas, o detección de apnea (Xie y Minn, 2012)), o procesado de imágenes de ultrasonido para la detección de la enfermedad de Hashimoto (inflamación de la glándula tiroidea) (Koprowsky et al., 2012), por citar sólo algunos ejemplos. Para hacerse una

[1] Billones 'americanos', miles de millones.

idea del número de contribuciones, basta con hacer una búsqueda en Google Scholar con los términos '*machine learning biomedicine*'.

Las cantidades de dinero que mueven todas estas aplicaciones hacen que tanto instituciones públicas como empresas privadas decidan invertir en ellas. Según (Bejerano, 2015) la Unión Europea ha invertido en el año 2015, 2500 millones de euros en un plan de 7 años con contribuciones públicas y privadas para el desarrollo de inteligencia artificial y robótica. Por otro lado en Estados Unidos en 2011 se realizó una inversión de 500 millones de dólares. En Asia también se está decidiendo invertir más dinero en estas tecnologías. La empresa Toyota por ejemplo, creó un plan en 2015 denominado 'Toyota Research Institute' para invertir 1000 millonres de euros en robótica e inteligencia artificial durante 5 años (Ling, 2015).

Uno de los problemas recurrentes que aparecen en la IA y en el aprendizaje máquina es la clasificación de patrones (Duda et al., 2001), que permite dotar a las máquinas de poder de decisión ante estímulos. La clasificación se puede utilizar en multitud de ámbitos, como por ejemplo clasificación de imágenes, detección de objetos, clasificación de sonidos, palabras, diagnosis médica, etc. Es una pieza fundamental para el aprendizaje máquina, y ha recibido una gran atención (existen por ejemplo revistas científicas dedicadas a este problema, como 'IEEE *Transactions on Pattern Analysis*' o '*Pattern recognition*', de la editorial Elsevier).

Dentro de la clasificación de patrones, existen numerosos problemas reales en los que los datos disponibles para realizar el aprendizaje están desequilibrados, en el sentido de que el número de patrones de cada clase puede ser sensiblemente diferente. Además por lo general en estos casos, la clase que menos muestras posee suele ser la que es más importante detectar correctamente. Algunos ejemplos de este tipo de problemas pueden ser la detección de fraude (de todo tipo, como uso de tarjetas de crédito, en telefonía, compañías de seguros, etc.), o la detección de enfermedades. En la detección de fraude se poseen más muestras de casos de usos legales que de usos fraudulentos, y estos últimos son los de mayor importancia, ya que fallar al detectar los usos fraudulentos puede suponer grandes pérdidas económicas. En el caso médico de la detección de enfermedades, se tienen más muestras de personas sanas que de personas que poseen la enfermedad, y éstas últimas son las de mayor riesgo, es decir sobre las que hay que minimizar el error de decidir incorrectamente. Este trabajo estará enfocado en la resolución de este tipo de problemas de clasificación desequilibrados.

En cuanto al marco regulador, éste dependerá fundamentalmente de la aplicación concreta sobre la que se utilice la herramienta de clasificación. El aspecto principal a tener en cuenta es la confidencialidad de los datos utilizados. Los datos, dependiendo de la aplicación concreta, estarán sujetos a la ley de protección de datos correspondiente a cada organización. Por

otro lado también hay que considerar el uso de las licencias del software a utilizar. Existen distintos tipos de licencias según el uso que se le vaya a dar a la herramienta, uso académico, doméstico, comercial, etc. También es posible que algunos algoritmos estén sujetos a patentes, lo que también habrá que tener en cuenta en caso de un uso con fines comerciales del mismo. Estos son los tres aspectos más importantes a tener en cuenta en el ámbito legal.

En definitiva, la IA y por consiguiente la clasificación, es un instrumento de gran utilidad e impacto social, y se prevee que en un futuro no muy lejano sea omnipresente, ya que sus aplicaciones tienen como único límite posible la imaginación.

1.2. Objetivos

El presente trabajo se sitúa en el ámbito de la clasificación de patrones. Concretamente se centra en problemas de clasificación binaria (decisión entre dos posibles clases) cuando los datos están desbalanceados o desequilibrados, es decir, que el número de muestras de cada clase es notablemente diferente.

Entre las distintas opciones para resolver este tipo de problemas, en este trabajo se utilizarán métodos de aprendizaje máquina, en particular redes neuronales.

Se han propuesto tres objetivos principales:

- Implementar clasificadores neuronales apropiados para problemas binarios desequilibrados, utilizando como elemento básico el perceptrón multicapa, que es la red neuronal más conocida y utilizada.
- Evaluar métodos de aprendizaje supervisado con modificaciones para afrontar con éxito problemas desequilibrados. Se probarán un método basado en la minimización del error cuadrático medio ponderado y un método basado en una formulación bayesiana que se ha propuesto recientemente.
- Evaluar la combinación de clasificadores individuales como método de clasificación para intentar aumentar las prestaciones en este tipo de problemas.

Además de los objetivos anteriores, se tendrán dos objetivos complementarios:

- Obtener un conjunto de bases de datos desbalanceados proveniente de aplicaciones reales. Los contenidos de las bases de datos serán de distintas temáticas, ya que lo que se pretende no es solucionar el problema

concreto de una aplicación específica, sino evaluar los métodos propuestos en el ámbito general de problemas desequilibrados.

- Comparar las distintas metodologías para el entrenamiento de la red sobre distintas bases de datos. De forma que se obtengan conclusiones sobre las prestaciones de las distintas metodologías según los parámetros de la red.

La memoria del trabajo se ha organizado de la siguiente forma:

- Este primer capítulo presenta la motivación y objetivos del trabajo, y lo contextualiza en su entorno socio-económico, y en el marco regulador aplicable.
- El capítulo 2 contiene un planteamiento formal del problema a resolver y una revisión del estado del arte relativo a dicho problema y a las soluciones propuestas en el trabajo.
- En el capítulo 3 se presentan las soluciones propuestas, justificando su elección frente a otras posibles alternativas, y se describe la metodología de trabajo para conseguir los objetivos propuestos.
- El capítulo 4 presenta los resultados más importantes que se han obtenido al aplicar los métodos propuestos en distintos problemas de clasificación desequilibrados.
- El capítulo 5 resume las principales conclusiones que se han obtenido a partir del trabajo realizado y plantea varias líneas posibles de continuación a partir del mismo que se consideran interesantes.
- En el apéndice A se muestra la planificación del trabajo y la estimación del presupuesto total desglosado en cada parte.
- En el apéndice B se encuentran los distintos acrónimos utilizados a lo largo del trabajo.
- Por último el listado de las referencias bibliográficas utilizadas.

Capítulo 2

Planteamiento del problema y estado del arte

En este capítulo se presenta una breve revisión del estado del arte relacionado con este trabajo. El capítulo comienza planteando el problema general de la clasificación de patrones, y su particularización a problemas binarios y desequilibrados. A continuación se revisarán las técnicas habitualmente utilizadas para resolver este tipo de problemas, y las medidas de prestaciones que se usan para comparar las distintas soluciones.

2.1. Clasificación de patrones

Un problema de clasificación trata de asignar a un patrón de datos D-dimensional, $\mathbf{x} \in \mathbb{R}^D$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_D \end{bmatrix}, \tag{2.1}$$

una clase o hipótesis de entre un conjunto de posibles clases

$$H = \{H_0, H_1, \ldots H_{N_c-1}\}, \tag{2.2}$$

siendo N_c el número de hipótesis del problema.

En la figura 2.1 se muestra un ejemplo de clasificación multiclase, en concreto de 4 clases, en un espacio de dimensión $D = 2$, donde los distintos tipos de formas geométricas representan los patrones de las diferentes clases, y las líneas azules describen las fronteras de decisión. Estas fronteras definen una región de decisión para cada una de las posibles clases. Se podría decir que el objetivo de un clasificador es establecer unas regiones de decisión apropiadas

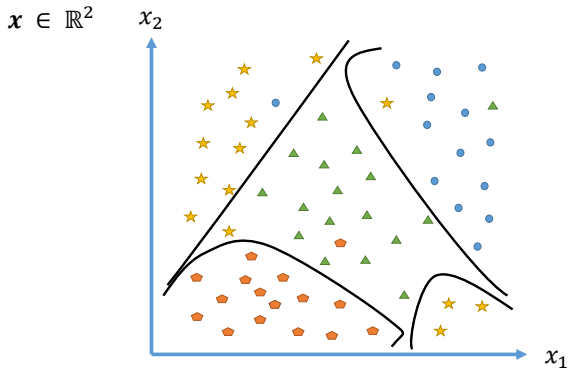

Figura 2.1: Ejemplo de clasificación multiclase, con 4 clases y un ejemplo de posibles regiones de decisión para cada clase.

para el problema en cuestión, teniendo en cuenta sus principales características. Las regiones que delimitan esas fronteras pueden ser disjuntas, como es el caso de las estrellas amarillas de la figura, pero las distinas áreas de decisión sumadas deben completar el espacio de observación del problema.

Como también se ilustra en la figura, es posible que no todos los patrones se clasifiquen de forma correcta, lo que significa que habrá ciertas probabilidades de error.

La clasificación es una herramienta muy útil en el mundo actual, y se ha utilizado en aplicaciones industriales, de negocios y científicas entre otras. En (Widrow et al., 1994) se puede encontrar una amplia lista de aplicaciones en distintos ámbitos.

La información de la que se dispone para resolver el problema puede ser de dos tipos principalmente:

- Conocimiento teórico del fenómeno. Por ejemplo una relación física entre los datos de entrada y las hipótesis. Esto quiere decir que se tiene un conocimiento previo de las características estadísticas del problema, como por ejemplo las probabilidades a priori de los sucesos y su distribución de probabilidad conjunta. Esto facilita el uso de técnicas como las que se describirán en la sección 2.2.

- Un conjunto de patrones de ejemplo, también denominado conjunto de entrenamiento. Se tienen datos y etiquetas. Las etiquetas indican la clase a la que pertenece cada uno de los patrones. Con esta información se pretende enseñar a la máquina a extraer relaciones no conocidas a priori. Son necesarios para los métodos de aprendizaje máquina que se presentarán más tarde en la sección 2.3.

2.1.1. Clasificación binaria

Se habla de clasificación binaria cuando el número de hipótesis del problema a tratar es dos, es decir

$$H = \{H_0, H_1\}, \tag{2.3}$$

donde H_0 es la hipótesis nula y H_1 la hipótesis positiva o alternativa.

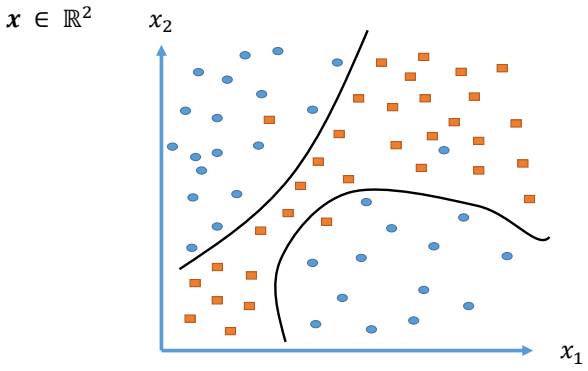

Figura 2.2: Ejemplo de clasificación binaria.

En la figura 2.2 se representa un ejemplo de clasificación con únicamente dos clases, y sus correspondientes regiones de decisión.

Las prestaciones de un clasificador binario están completamente determinadas por dos probabilidades de error. Si $\hat{H}$ denota la estima realizada por el clasificador, estas dos probabilidades se definen como:

- Probabilidad de falsa alarma:

$$p_{FA} = p_{\hat{H}|H}(H_1|H_0) \equiv P(\text{ Escoger } H_1|H_0 \text{ es correcta }). \tag{2.4}$$

- Probabilidad de pérdida:

$$p_M = p_{\hat{H}|H}(H_0|H_1) \equiv P(\text{ Escoger } H_0|H_1 \text{ es correcta }). \tag{2.5}$$

Es decir, la probabilidad de falsa alarma define la probabilidad de error del clasificador para patrones de la clase H_0, y la probabilidad de pérdida (*missing*) define la probabilidad de error del clasificador para patrones de la clase H_1.

Además, en ocasiones se usa, en lugar de la p_M, su probabilidad complementaria, denominada probabilidad de detección p_D :

$$p_D = 1 - p_M = p_{\hat{H}|H}(H_1|H_1) \equiv P(\text{ Escoger } H_1|H_1 \text{ es correcta }). \tag{2.6}$$

De esta forma, el tandem (p_{FA}, p_D) cuantifica las probabilidades de tomar una decisión positiva bajo las dos hipótesis.

No todos los problemas a clasificar tienen las mismas características, por lo que es importante tener en cuenta sus peculiaridades para usar unos métodos de decisión u otros. A continuación se describen algunos casos singulares.

2.1.2. Problemas con distintos costes

Hay multitud de problemas en la vida real en los que el coste de fallar sobre una clase o hipótesis no es igual que el de fallar sobre la clase contraria. Un ejemplo sería un caso médico en el que hay que decidir si el paciente tiene riesgo de sufrir cáncer para realizarle o no un tratamiento. No es lo mismo el coste de decidir que el paciente tiene cáncer y que luego no lo tenga (falsa alarma), que el coste de decidir que no lo tiene y que realmente sí lo padezca (pérdida).

Hay que remarcar esa diferencia por tanto para que se contemple con distinto grado la importancia de fallar en cada clase. Por ello existen distintos mecanismos posibles, como los que se verán reflejados en la sección 2.4.

2.1.3. Problemas desequilibrados

Un problema de clasificación se considera desequilibrado o desbalanceado cuando el número de muestras disponibles de cada hipótesis o clase es notablemente diferente. Según el artículo (Galar et al., 2013) se puede considerar la tasa de desequilibrio (IR: *Imbalance Ratio*) como el número de muestras de la clase negativa dividido entre el número de muestras de la clase positiva. En este artículo si el IR es superior a 9, se asume que los datos son altamente desequilibrados.

La particularidad de estos problemas hace que si se usan métodos de clasificación tradicionales diseñados para trabajar con datos equilibrados, pueda suceder que la detección sobre la clase mayoritaria sea buena. Pero como la clase minoritaria se ve menos reflejada en el conjunto de datos es posible que se produzcan más errores sobre ella. Para resolver este problema existen varias metodologías:

(a) Se modifican los datos para equilibrar el problema. O bien descartando muestras de la clase mayoritaria, o bien generando muestras sintéticas de la clase minoritaria. También se pueden hacer ambas cosas a la vez. Una descripción detallada de este tipo de técnicas se puede encontrar en (Galar et al., 2012) (He y Garcia, 2009).

(b) Sin modificar los datos originales, se modifica el clasificador máquina para que tenga en cuenta el desbalanceo de los datos.

(c) Utilización de conjuntos de clasificadores. Se utiliza la combinación de distintos clasificadores individuales. Esto ha demostrado en la práctica buenos resultados, como se puede ver, por ejemplo, en (Galar et al., 2013).

Además se pueden realizar estas metodologías de forma combinada.

Este trabajo se enfoca en solucionar este tipo de problemas con la opción (b), sin modificación previa de los datos modificando clasificadores individuales, y (b) en combinación con (c), combinando los clasificadores individuales en un clasificador conjunto.

Una vez presentado el problema básico considerado en este trabajo, la clasificación binaria en problemas desequilibrados, a continuación se presentarán los distintos tipos de métodos que se pueden utilizar para resolver este problema. En la sección 2.2 se expondrán los métodos estadísticos y en la sección 2.3 los métodos máquina.

2.2. Métodos estadísticos

Los métodos estadísticos basan sus decisiones en el conocimiento teórico del problema a priori. Es decir, que se parte del conocimiento de las características estadísticas de los sucesos. O bien se tienen las distribuciones de probabilidad individuales y la distribución de probabilidad conjunta, o se tienen las distribuciones de probabilidad condicionales de los patrones dada cada hipótesis.

A continuación se va a describir la técnica estadística más utilizada.

2.2.1. Aproximación bayesiana

La teoría bayesiana tiene en cuenta las probabilidades a priori de cada clase y el coste de equivocarse en cada hipótesis.

Estos parámetros permiten definir el denominado riesgo de Bayes de la forma

$$R = \sum_{h \in H} \sum_{d \in H} \pi_h \, c_{d,h} \, p_{\hat{H}|H}(d|h). \tag{2.7}$$

donde π_h denota la probabilidad de la hipótesis h, $c_{d,h}$ es el coste de decidir d, cuando la hipótesis correcta es h, y $p_{\hat{H}|H}(d|h)$ define la probabilidad condicional

$$p_{\hat{H}|H}(d|h) \equiv P(\text{Escoger } d|h \text{ es correcta}). \tag{2.8}$$

En el caso de un problema de clasificación binario, se podrían encontrar cuatro costes $c_{d,h}$ determinados por la decisión tomada d y por la hipótesis correcta h.

La regla de decisión que minimiza el riesgo de Bayes (Van Trees, 1968) viene determinada por las funciones de densidad de probabilidad condicionales, también denominadas verosimilitudes. Se puede escribir tal regla de la forma

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{X}|H}(\mathbf{x}|H_1)}{f_{\mathbf{X}|H}(\mathbf{x}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{(c_{1,0} - c_{0,0})}{(c_{0,1} - c_{1,1})} \frac{\pi_0}{\pi_1} = \gamma(> 0), \tag{2.9}$$

donde $\Lambda(x)$ es el cociente de verosimilitudes o *likelihood ratio*, y γ es el umbral con el que comparar dicho cociente.

Por simplicidad, y sin pérdida de generalidad, en la mayoría de los casos se asume que el coste de acertar es nulo, por lo que $c_{1,1} = c_{0,0} = 0$.

Es interesante observar que el valor de los costes $c_{1,0}$ y $c_{0,1}$ no es determinante, sin embargo sí lo es la relación entre ambos, es decir, que es indiferente que los costes sean $(c_{1,0} = 4,\ c_{0,1} = 2)$ o $(c_{1,0} = 2,\ c_{0,1} = 1)$, lo importante para la clasificación es su cociente.

De esta forma el riesgo de Bayes quedaría

$$R = \pi_0\ c_{1,0}\ p_{\hat{H}|H}(1|0) + \pi_1\ c_{0,1}\ p_{\hat{H}|H}(0|1). \tag{2.10}$$

Teniendo en cuenta las definiciones de las probabilidades de falsa alarma y de pérdida, se puede escribir el riesgo como

$$R = \pi_0\ c_{1,0}\ p_{FA} + \pi_1\ c_{0,1}\ p_M, \tag{2.11}$$

donde las probabilidades de cada hipótesis y los costes de equivocarse ponderan las probabilidades de fallar sobre cada una de las clases.

En ocasiones para simplificar la notación se pueden parametrizar estos dos términos de ponderación a través de un único parámetro α que establece directamente la ponderacón relativa entre p_{FA} y p_M. Se definen entonces $c_{1,0}$ y $c_{0,1}$ de la forma

$$c_{1,0} = \frac{\alpha}{\pi_0},\ c_{0,1} = \frac{1-\alpha}{\pi_1}, \tag{2.12}$$

donde α establece la importancia de fallar en cada una de las clases, ya que utilizando esta parametrización se puede escribir el riesgo de Bayes como

$$R(\alpha) = \alpha\ p_{FA} + (1-\alpha)\ p_M. \tag{2.13}$$

El umbral se define en función de la relación entre los costes y las probabilidades a priori, o de una forma más simplificada utilizando la parametrización con α

$$\gamma = \frac{c_{1,0}\ \pi_0}{c_{0,1}\ \pi_1} = \frac{\alpha}{1-\alpha}. \tag{2.14}$$

Además de Bayes, hay otros métodos estadísticos con otras características como Neyman-Pearson, minimax, máxima verosimilitud (ML: *Maximum-Likelihood*), etc. No se han incluido en esta revisión porque no se utilizarán en este trabajo, pero se puede encontrar más información sobre estos en cualquier libro clásico de detección estadística, como (Kay, 1998), (Van Trees, 1968) o (Poor, 1994).

2.3. Métodos de aprendizaje máquina

Cuando no se conocen las funciones de densidad de probabilidad condicionales para cada hipótesis o las estimaciones que podemos realizar de ellas no son buenas, la aplicación de los métodos estadísticos no es posible o proporciona pobres resultados. En este caso una de las alternativas más habituales es utilizar los denominados métodos de aprendizaje máquina, los cuales se basan en aprender a partir de ejemplos, denominados también conjuntos de datos de entrenamiento.

Se tiene como punto de partida un conjunto de N_p patrones etiquetados

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_p} \in \mathbb{R}^D, \tag{2.15}$$

donde $\mathbf{x}_i$ son los datos, y_i las etiquetas, y D es la dimensión de los datos.

Las etiquetas comúnmente toman para el caso binario el valor $y_i = +1$ para la hipótesis H_1, y para la hipótesis H_0 el valor 0 ó -1. Aquí en el segundo caso se ha optado por $y_i = -1$.

Los métodos de aprendizaje máquina se pueden clasificar primero en dos tipos, de aprendizaje supervisado y no supervisado. Los primeros se caracterizan porque los datos de entrada están etiquetados, es decir, se conoce cúal es la hipótesis correcta, y esa información se utiliza para entrenar la red. En el caso de los no supervisados (Isasi y Galván, 2004), no se tienen etiquetas sobre los datos, sólamente entradas, y se debe encontrar una estructura que los caracterice (Clustering, PCA, etc. (Barber, 2010)).

En este caso, se van a abordar métodos supervisados, ya que el estudio del trabajo trata sobre datos etiquetados. Con estos métodos, a partir de un conjunto de datos cuya etiqueta es conocida, las máquinas pueden aprender a sacar relaciones y clasificar nuevos conjuntos de datos de los cuales no tengamos etiquetas. Para el exterior funcionan como cajas negras que dados un conjunto de datos de entrenamiento, devuelven una hipótesis sobre cada muestra.

Los métodos están compuestos por dos partes. Primero, la arquitectura de la red, la cual define la relación entre los patrones y las hipótesis. Esta relación depende de una serie de parámetros de la máquina, comúnmente

denominados coeficientes o pesos. La salida de la red o_k para un patrón $\mathbf{x}_k$, dependerá por tanto de los pesos de la red,

$$o_k = f(\mathbf{x}_k, \mathbf{w}), \tag{2.16}$$

siendo $\mathbf{w}$ el vector que contiene los pesos de la red. Y en segundo lugar se encuentra el algoritmo de entrenamiento, que es el encargado de encontrar los valores de los coeficientes o pesos más adecuados de la red para obtener las prestaciones deseadas (Bishop, 1997).

Se van a describir a continuación algunas de las arquitecturas de red más importantes: las máquinas de vectores soporte, los árboles de decisión y las redes neuronales.

2.3.1. Máquinas de vectores soporte (SVM)

Las máquinas de vectores soporte (SVM: *Support Vector Machines*), fueron propuestas inicialmente por Vladimir Vapnik (Vapnik et al., 1997) gracias a su teoría de aprendizaje estadístico. Su uso ha sido tanto para clasificación de patrones como para aplicaciones de regresión no lineal. Se trata de un método de aprendizaje supervisado, y es una aplicación aproximada del método de minimización de riesgo estructural, que se basa en limitar la tasa de error de una máquina de aprendizaje sobre los datos según unos parámetros determinados. En consecuencia, las SVMs proporcionan una buena generalización.

Se pueden dar distintos casos de máquinas de vectores soporte según si los patrones se pueden separar de forma lineal o no. A continuación, se explican los conceptos teóricos cuando nos encontramos en el caso más sencillo, cuando los datos son linealmente separables, en concreto en clasificación binaria.

Caso linealmente separable

El caso separable linealmente es aquel cuyas muestras pueden dividirse por completo a través de una frontera lineal, como se observa en el ejemplo de la figura 2.3. Puede haber infinitas fronteras de clasificación que distingan las clases, pero el objetivo es encontrar aquella que haga que el hiperplano (extensión de un plano) que establece la separación entre clases tenga su máxima distancia o margen con respecto a las muestras más cercanas de las dos clases.

Cuando se pueden separar los patrones de las dos clases de forma lineal, el hiperplano de separación se puede representar de la forma:

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{2.17}$$

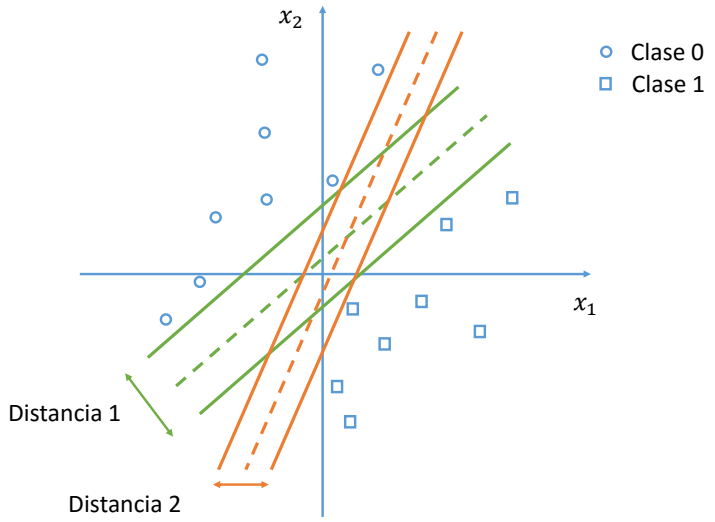

Figura 2.3: Ejemplo binario de un caso linealmente separable.

donde $\mathbf{w}$ es el vector normal al hiperplano.

La función de clasificación determinará la hipótesis asignada a las muestras, de forma que cuando la ecuación (2.18) sea mayor que 0 las muestras serán asignadas a una hipótesis, y por el contrario, si son menor que 0 a la otra

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b). \tag{2.18}$$

Lo deseado es hallar los valores de $\mathbf{w}$ y b que maximicen el margen o separación entre clases. Este clasificador se expresará de la forma

$$\mathbf{w}^T\mathbf{x}_i + b > +M,\ \mathbf{w}^T\mathbf{x}_i + b < -M, \tag{2.19}$$

donde M determina el margen o separación entre clases. De manera compacta, se tendría

$$y_i\ (\mathbf{w}^T\mathbf{x}_i + b) \geqslant M, i = 1, \ldots, N. \tag{2.20}$$

para valores $y_i = +1$ e $y_i = -1$.

Normalizando las expresiones anteriores, se llega a

$$\mathbf{w}^T\mathbf{x}_i + b > +1,\ \mathbf{w}^T\mathbf{x}_i + b < -1, \tag{2.21}$$

y de forma más compacta

$$y_i\ (\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, i = 1, \ldots, N. \tag{2.22}$$

Teniendo en cuenta los puntos que cumplen la igualdad anterior, se obtienen dos hiperplanos paralelos entre sí, y paralelos al hiperplano de separación

$$\mathbf{w}^T\mathbf{x}_i + b = +1, \;\; \mathbf{w}^T\mathbf{x}_i + b = -1, \tag{2.23}$$

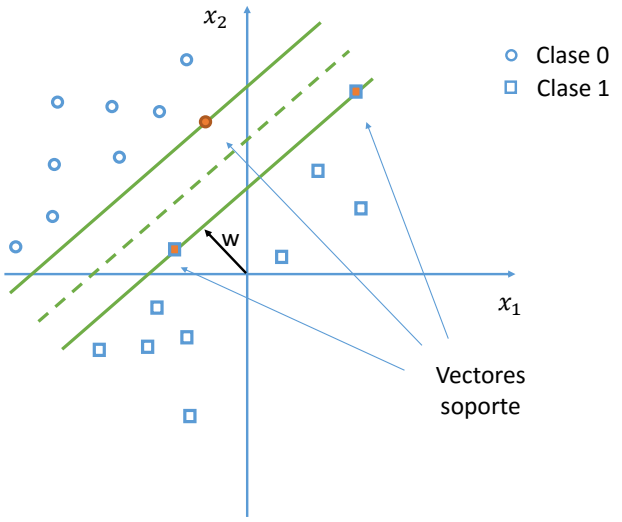

Figura 2.4: Ejemplo caso binario y separable linealmente.

Se puede demostrar que el margen del hiperplano es $\frac{2}{\|\mathbf{w}\|}$. Por lo tanto se hallará el hiperplano que proporciona la máxima separación minimizando la siguiente función

$$\tau(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}, \tag{2.24}$$

teniendo en cuenta las restricciones de la ecuación (2.22).

Los puntos más cercanos al hiperplano van a delimitar la frontera de decisión, y los que se encuentran en el propio margen son los denominados vectores soporte.

Como se puede ver por ejemplo en (Haykin, 1998) los multiplicadores de Lagrange permiten optimizar este problema. El lagrangiano se expresa de la forma

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i(y_i\,(\mathbf{w}^T\mathbf{x}_i + b) - 1), \tag{2.25}$$

donde α_i denota los multiplicadores de Lagrange (con valores positivos). Se desea minimizar la expresión (2.25) con respecto las variables $\mathbf{w}$ y b, y maximizarla con respecto a α_i. Para minimizarla, se realiza la derivada parcial con respecto a cada una de las variables y se iguala a 0, como aparece a continuación:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial w} = 0 \tag{2.26}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \tag{2.27}$$

Tales expresiones dan como resultado:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, y_i \, \mathbf{x}_i \tag{2.28}$$

$$\sum_{i=1}^{N} \alpha_i \, y_i = 0 \tag{2.29}$$

De esta forma la función de decisión se puede escribir de la forma

$$f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{N} y_i \, \alpha_i \, (\mathbf{x}^T \mathbf{x}_i) + b). \tag{2.30}$$

Sólo tomarán valores mayores que 0 los multiplicadores asociados a los vectores soporte, esto se traduce en

$$f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{N_{vs}} y_i \, \alpha_i \, (\mathbf{x}^T \mathbf{x}_i^{vs}) + b) \text{ para } \alpha_i \neq 0, \tag{2.31}$$

donde N_{vs} es en número de muestras vectores soporte, y $\mathbf{x}_i^{vs}$ tales vectores.

Pero no siempre los datos son linealmente separables, en el caso de no serlo las funciones de coste se ven afectadas con nuevos parámetros. Existe un denominado truco del núcleo (*kernel trick*) que permite establecer fronteras no lineales en el espacio de entrada manteniendo la formulación del hiperplano de máximo margen. Este lo que hace es definir una transformación sobre el espacio de entrada que se proyecta en otro espacio de dimensión mayor, y sobre ese nuevo espacio proyectado se puede plantear una frontera de separación lineal, es decir un hiperplano que maximice el margen.

Tras la explicación teórica es importante destacar las ventajas y desventajas más llamativas.

Una de las ventajas principales de las SVM es que la función a minimizar (ver ecuación (2.25)) es convexa (figura 2.5), y se puede encontrar la solución óptima que minimiza el coste con las restricciones impuestas. Aunque en el caso de datos no separables o de la utilización del truco del núcleo previamente hay que establecer unos hiperparámetros, estos habitualmente han de seleccionarse utilizando validación cruzada, explicada en (Barber, 2010), lo que aumenta la carga computacional.

En las SVM se utiliza comúnmente la programacion cuadrática (Nocedal y Wright, 2006), pero este algoritmo tiene el problema de aumentar mucho el

coste computacional cuando las dimensiones de los datos crecen. Para solucionarlo se han desarrollado técnicas de algoritmos rápidos como el método LibSVM (Chang y Lin, 2011).

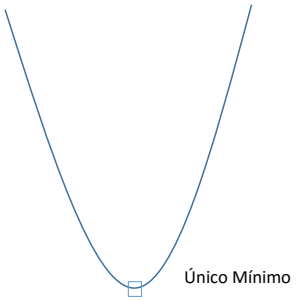

Figura 2.5: Función convexa.

Entre las desventajas de las SVM se encuentra el tiempo de entrenamiento cuando el volúmen de datos es demasiado grande. Además el hecho de que para conseguir la ventaja de un sólo mínimo previamente, hay que escoger los hiperparámetros óptimos de la red, los cuales a priori no se conocen. Para elegirlos es necesario hacer validación cruzada, explicada en (Barber, 2010).

Para más información sobre las SVM, se puede consultar, por ejemplo (Schölkopf et al., 1999), (Haykin, 1998) o (Barber, 2010).

2.3.2. Árboles de decisión

Los árboles de decisión forman un proceso secuencial de decisión. Se trata de un modelo fácil de interpretar. Está compuesto por nodos y ramas, los primeros pueden dividirse en tres tipos:

- Nodo de decisión: punto en el cual debe realizarse una elección sobre los datos. Cada rama saliente de este nodo representa una de las posibles alternativas (deben ser mutuamente excluyentes). Se suelen representar con un cuadrado.

- Nodo de evento o probabilidad: punto en el cual pueden suceder una serie de eventos con unas probabilidades determinadas. Cada rama define un evento diferente con una probabilidad concreta. Comúnmente se representan con un círculo.

- Nodo terminal u hoja: representa el valor final de la rama de decisión. Se expresan generalmente con un triángulo.

De esta forma la solución final se puede ver como el sumatorio de los pesos de las ramas. Según la naturaleza de los datos, los árboles de decisión se pueden diferenciar en dos tipos:

- Árboles de clasificación: cuando las variables son discretas. El resultado del nodo hoja representa la etiqueta de la clase (número finito).
- Árboles de regresión: cuando las variables son continuas.

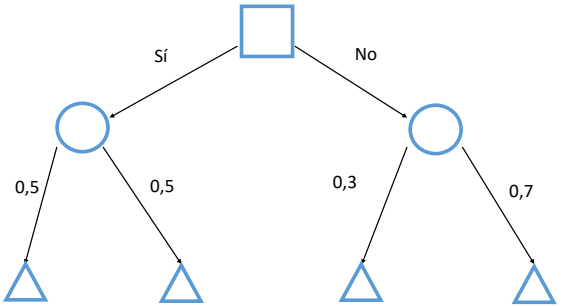

Figura 2.6: Ejemplo árbol de decisión.

Este método es uno de los más sencillos. Se puede observar un ejemplo en la figura 2.6, donde aparece un nodo de decisión que tiene dos alternativas para escoger y posteriormente dos nodos de evento, con dos ramas cada uno con sus respectivas probabilidades. Finalmente se encuentras las hojas.

Para más información sobre árboles de decisión se puede consultar, por ejemplo (Barber, 2010) o (Alpaydin, 2004).

2.3.3. Redes neuronales (NN)

También denominadas redes neuronales artificiales (ANN: *Artificial Neural Network*), son modelos inspirados en el funcionamiento del cerebro. Una red neuronal artificial aprende de patrones etiquetados, lo que en el cerebro se traduce en experiencias. Se basan en la interconexión de neuronas o nodos y se organizan en una capa de entrada, una capa de salida, y una o varias capas intermedias u ocultas.

Según (Isasi y Galván, 2004), las redes neuronales artificiales forman parte de la inteligencia artificial (IA), y dentro de ella más concretamente en el área subsimbólica. Esta área trabaja "de abajo hacia arriba (*bottom-up*), ya que los sistemas diseñados son simples e idénticos, recogen las caracterísicas

físicas de los sistemas que tratan de imitar, y se van generando cómputos cada vez más complejos, de forma automática, mediante mecanismos prefijados de aprendizaje". El objetivo ideal es que la red trabaje de forma paralela, de tal manera que sea lo más fiel posible al cerebro.

Con los denominados algoritmos de entrenamiento se pretende encontrar los parámetros o pesos de la red neuronal que minimicen una determinada función de coste definida sobre los datos de entrenamiento y apropiada para el problema a resolver. Normalmente, la minimización de esta función de coste se realiza mediante técnicas de descenso de gradiente a partir de una inicialización aleatoria de los parámetros. La función de coste puede ser por ejemplo una función de error cuadrático.

La principal desventaja que tienen este tipo de redes es el aspecto de los mínimos locales, ya que la función a minimizar habitualmente no es convexa (figura 2.7). Como al inicializarse en un punto el algoritmo sólo desciende, dependiendo de cual sea el punto inicial se obtendrá un mínimo local o global, y el resultado del error puede variar notablemente.

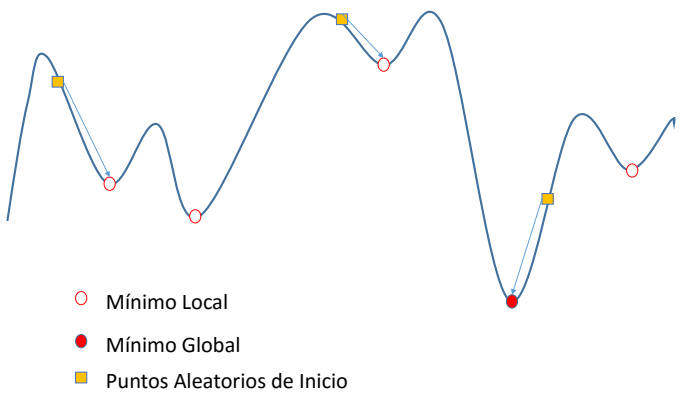

Figura 2.7: Ejemplo de función con varios mínimos locales.

En la figura 2.7 se puede observar el problema de mínimos locales. La solución consiste en repetir el entrenamiento desde distintos valores iniciales aleatorios un número suficiente de veces para tratar de llegar al mínimo global o en su defecto a un buen mínimo local, aunque esto implica un aumento de la carga computacional.

Por el contrario, algunas de las ventajas más importantes según (Haykin, 1998) son la no linealidad, el aprendizaje supervisado y la adaptabilidad a cambios, entre otras.

Dentro de los múltiples tipos de redes neuronales que existen, se van a describir dos de los más usados, el perceptrón multicapa (MLP: *Multilayer*

Perceptron), y la red de funciones de base radial (RBF: *Radial Basis Function network*).

Perceptrón multicapa (MLP)

También denominada red multicapa con conexiones hacia adelante (*feed-forward*), es un tipo de red neuronal cuyo algoritmo de entrenamiento más usado es el de retropropagación de errores (Rumelhart et al., 1986b) (Widrow y Lehr, 1990). Está formado por una capa de entrada, una o más capas ocultas, y una capa de salida como se refleja en la figura 2.8.

La primera capa capta directamente la información de los patrones de entrada, y no realiza procesado alguno, sino que transmite esa información hacia la siguiente capa de neuronas. En la capa o capas intermedias es donde se realiza el procesado no lineal de los datos. La última capa actúa como salida al exterior de la red.

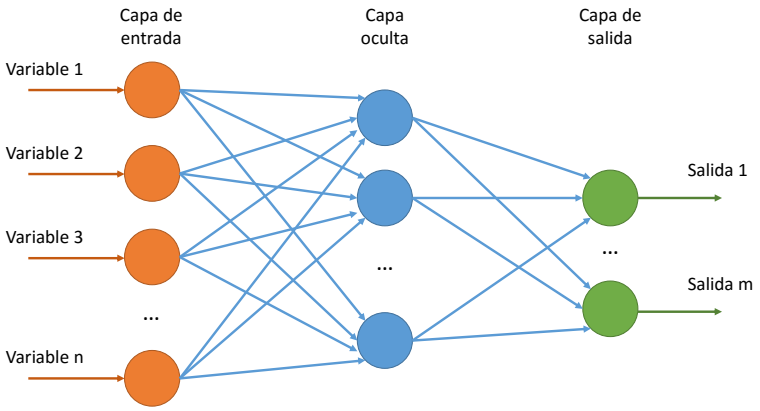

Figura 2.8: Esquema de una red neuronal con una única capa oculta.

Se trata de una de las arquitecturas más utilizadas para solucionar problemas gracias a la consideración como aproximador universal (Hornik et al., 1989) (Cybenko, 1989), en aplicaciones como diagnósticos médicos, predicción de temporales, reconocimiento de habla, etc. (Atiya, 2001) (Dorronsoro et al., 1997).

La red puede estár totalmente conectada, es decir que una capa tiene todas sus neuronas contectadas por completo con las neuronas de la capa anterior y de la capa posterior, o por el contrario estar parcialmente conectada.

Como se ha mencionado antes, se realiza una propagación hacia adelante, las neuronas procesan la información que reciben y generan una activación que se transmite por las uniones entre neuronas. Dicha activación se calcula de la siguiente forma.

Siendo N_c el número de capas de la red, N_n^c el número de neuronas de cada capa c y la matriz de pesos $W^c = (w_{ij}^c)$ asociada a la conexión entre la capa c y la $c+1$, donde el peso (w_{ij}^c) pertenece a la conexión entre la neurona i de la capa c y la neurona j de la capa $c+1$. Y siendo $U^c = (u_i^c)$ el vector de umbrales de las neuronas de la capa c. La activación de la neurona i de la capa c se denomina a_i^c, y se calcula de la forma que se muestra a continuación:

- Capa de entrada
$$a_i^1 = x_i \text{ para } i = 1, 2, ..., D \tag{2.32}$$
donde $\mathbf{x} = (x_1, x_2, ..., x_D)$ es el patrón de entrada. En este caso las neuronas sólo transmiten los valores de la entrada.

- Capas ocultas
$$a_i^c = f(\sum_{j=1}^{n_n - 1} w_{ji}^{c-1} a_j^{c-1} + u_i^c) \text{ para } i = 1, 2, ..., n_n \text{ y } c = 2, 3, ..., N_c - 1 \tag{2.33}$$
donde se procesa la información de la capa anterior y se le aplica la función de activación f al sumatorio de los productos de las activaciones anteriores por sus pesos.

- Capa de salida
$$y_i = a_i^c = f(\sum_{j=1}^{n_n - 1} w_{ji}^{c-1} a_j^{c-1} + u_i^c) \text{ para } i = 1, 2, ..., n_n \tag{2.34}$$
donde $\mathbf{y} = (y_1, y_2, ..., y_{n_n})$ son las salidas de la red. El proceso es similar al de las capas ocultas.

Las funciones de activación f más comunes se definen de la siguiente forma:

- Lineal.
$$f(x) = x \tag{2.35}$$

- Tangencial Hiperbólica.
$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{2.36}$$

- Logística.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.37}$$

En el caso de la función logística su valor es siempre positivo, y crece entre 0 y +1, en ambos valores la función satura. En el caso de la función tangencial satura en -1 y +1 y la lineal no satura. Las tres funciones se pueden ver representadas en la figura 2.9.

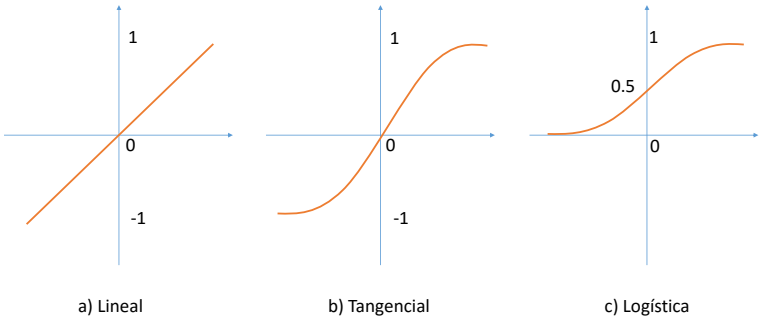

Figura 2.9: Funciones de activación utilizadas con mayor frecuencia en perceptrones multicapa.

El algoritmo de entrenamiento más común para que la red aprenda los parámetros es un algoritmo de retropropagación de errores (Rumelhart et al., 1986a), que lo que pretende es hallar los pesos de la red de forma que se minimice una función de coste definida por el error cuadrático medio entre la salida de la red y las etiquetas de los patrones del conjunto de entrenamiento. Debido a que las funciones de activación no tienen por qué ser lineales, el problema a minimizar no es lineal. Lo más frecuente es usar el método de descenso por gradiente, que se explicará en la sección 2.4, dentro de éste existen varios tipos, el más habitual es por bloque, aunque también se utiliza el estocástico donde en lugar de tener en cuenta el error global, el que se intenta minimizar de forma iterativa y secuencial es el error de cada patrón (Isasi y Galván, 2004).

Para profundizar más en estas redes se puede consultar, por ejemplo en (Haykin, 1998), (Widrow y Lehr, 1990) o (Funahashi, 1998).

Red de funciones de base radial (RBF)

Se trata de un tipo de red neuronal con conexiones hacia delante como el caso de MLP. Se caracteriza por tener únicamente una capa oculta, y por tener como función de activación las denominadas funciones de base radial

[1] (normalmente gaussianas). Por otro lado en la capa de salida las neuronas usan función lineal.

Las funciones de base radial le dan a las neuronas un carácter local, ya que se activan en una región concreta del espacio. Están definidas por:

- Centro, donde se sitúa la media de la función gaussiana.
- Ancho o varianza, que define la amplitud de la campana de Gauss.

Los nodos o neuronas de las distintas capas están completamente conectadas a las neuronas de la capa anterior y posterior. La estructura de una red RBF se puede ver representada en la figura 2.10.

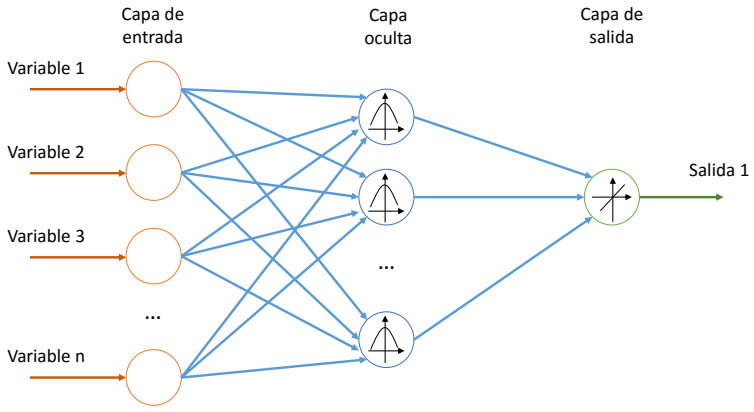

Figura 2.10: Esquema de una red RBF.

La activaciones de las neuronas se realizan de tal forma que para un patrón de entrada $\mathbf{x}_k$ la salida es

$$o_k(\mathbf{x}_k) = \sum_{i=1}^{N_n} w_{si}\ \phi(\mathbf{x}_k, \theta_i), \tag{2.38}$$

donde w_{si} son los pesos i-ésimos de la capa de salida, θ_i contiene los parámetros centro y desviación típica de la neurona i y $\phi(\mathbf{x}_k, \theta_i)$ es la función de base radial

$$\phi(\mathbf{x}_k, \theta_i) = f(\|\mathbf{x}_k - c_i\|). \tag{2.39}$$

[1] Una función de base radial se caracteriza por ser una función que depende de un punto central, y un ancho o varianza de los datos con respecto al centro.

Esta función depende de la distancia al centro. Como se ha mencionado anteriormente es muy común usar la función gaussiana que se define de la siguiente forma

$$\phi(\mathbf{x}_k, w_o^n) = \exp(\sum_{d=1}^{D} -\frac{(x_{k,d} - \mu_{n,d})^2}{2\sigma_n^2}), \tag{2.40}$$

donde $\mu_{n,d}$ es el centro o media y σ_n^2 es la varianza o anchura.

Uno de los principales inconvenientes de este tipo de redes proviene de la naturaleza local de sus funciones de activación, y es el crecimiento exponencial del número necesario de neuronas N_n en la capa oculta cuando la dimensión de los datos D aumenta. Para solucionar esta desventaja, se emplea una extensión, las GRBF (*Generalised Radial Basis Function network*) cuya diferencia más significativa es que puede considerar varianzas desiguales en cada dirección del espacio, de forma que se adapte mejor a las peculiaridades de los datos (Haykin, 1998).

Las aplicaciones de estas redes son amplias, algunas como análisis de series temporales (Moody y Darken, 1989) o procesamiento de imágenes, pero su uso no ha llegado a ser tan amplio como el de las redes MLP. Para más información sobre este tipo de red neuronal se podría consultar por ejemplo (Bishop, 1997) o (Haykin, 1998).

2.4. Funciones de coste para el entrenamiento de redes neuronales

Como ya se ha dicho con anterioridad las redes neuronales aprenden a partir de un conjunto de patrones etiquetados, o conjunto de entrenamiento, de esta forma la red aprende la relación entre los patrones y las etiquetas. El conjunto de entrenamiento con N_p patrones

$$\{\mathbf{x}_k, y_k\}_{k=1}^{N_p}, \tag{2.41}$$

donde $\mathbf{x}_k \in \mathbb{R}^D$ son los patrones de entrada con dimensión D, y $y_k \in \mathbb{R}^{N_s}$ son las etiquetas con dimensión N_s (N_s es el número de salidas de la red neuronal).

La red neuronal efectúa el mapeo

$$\mathbf{o}_k = f(\mathbf{x}_k, \mathbf{w}) = \begin{bmatrix} f_0(\mathbf{x}_k, \mathbf{w}) \\ f_1(\mathbf{x}_k, \mathbf{w}) \\ \cdots \\ f_{N_s-1}(\mathbf{x}_k, \mathbf{w}) \end{bmatrix},$$

donde $\mathbf{w}$ es el vector que contiene los parámetros de la red. El vector de salida tiene N_s componentes, es decir $o_{k,s} = f_s(\mathbf{x}_k, \mathbf{w})$.

Para un problema concreto de clasificación binaria con una única salida ($N_s = 1$) con las etiquetas

$$y_k = -1, \ si \ \mathbf{x}_k \in H_0 \ e \ y_k = +1, \ si \ \mathbf{x}_k \in H_1, \tag{2.42}$$

la decisión de la hipótesis se puede dar mediante la regla

$$o_k \underset{\hat{H}=H_0}{\overset{\hat{H}=H_1}{\gtrless}} 0. \tag{2.43}$$

Por lo que se compara la salida con un umbral en cero, y asumiendo que las etiquetas ± 1 identifican la clase como la regla de decisión

$$\hat{y}_k = \text{sgn}(o_k). \tag{2.44}$$

Los parámetros $\mathbf{w}$ se obtienen minimizando alguna función de coste definida sobre el conjunto de entrenamiento, y dependiente de los parámetros de la red, $J(\mathbf{w})$.

Para minimizar dicha función, se utiliza típicamente un algoritmo de optimización iterativo denomidado método de descenso por gradiente. Este método adapta parámetros en la iteración i a partir de los resultados de la iteración i-1, utilizando la regla de adaptación

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} - \mu \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}^{(i-1)}}, \tag{2.45}$$

donde μ es un parámetro de paso.

El gradiente se puede expresar de la siguiente forma

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{k=1}^{N} \frac{\partial J(\mathbf{w})}{\partial o_k} \frac{\partial o_k}{\partial \mathbf{w}}, \tag{2.46}$$

donde el primer término del sumatorio depende de la función de coste, y el segundo término depende de la arquitectura de la red, que en nuestro caso como se verá posteriormente en el capítulo 3 será una red MLP. Puesto que la arquitectura de la red utilizada va a ser única ese término no varía, pero la función de coste puede cambiar según los casos que se van a exponer a continuación.

2.4.1. Error cuadrático medio (MSE)

Una de las funciones de coste más comunes es la del error cuadrático medio (MSE: *Mean Square Error*), que como se trata de minimizarla en ocasiones

también se denomina función de mínimo error cuadrático medio (MMSE: *Minimum Mean Square Error*), la cual se define de la siguiente forma

$$J^{MSE}(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - o_k)^2. \tag{2.47}$$

Se realiza la derivada parcial de la ecuación (2.47) con respecto a o_k

$$\frac{\partial J^{MSE}(\mathbf{w})}{\partial o_k} = -\frac{2}{N}(y_k - o_k), \tag{2.48}$$

de forma que el resultado de la ecuación anterior se introduciría en la ecuación (2.46).

El algoritmo trata de ajustar las etiquetas de los patrones del conjunto de entrenamiento. En problemas desequilibrados, la función de coste puede tener el problema de ocultar los patrones de la clase minoritaria, que contribuirán en el coste en menor medida que las de la clase mayoritaria.

El algoritmo de retropropagación de errores mencionado anteriormente en la sección 2.3.3, fue definido inicialmente sobre esta función de coste (Rumelhart et al., 1986b).

2.4.2. Error cuadrático medio ponderado (WMSE)

Para problemas con datos desequilibrados o con problemas con costes distintos, puede resultar conveniente tener en cuenta ese desbalanceo en el entrenamiento de la red. Una posibilidad para tener esto en cuenta es utilizar el error cuadrático ponderado (WMSE: *Weighted Mean Square Error*) como función de coste, la cual se describe de la siguiente forma

$$J^{WMSE}(\mathbf{w}) = \frac{\alpha}{N_0}\sum_{k\in S_0}(y_k - o_k)^2 + \frac{1-\alpha}{N_1}\sum_{k\in S_1}(y_k - o_k)^2, \tag{2.49}$$

siendo S_0 y S_1 los conjuntos de índices correspondientes a los patrones de ambas clases, y donde N_0 y N_1 son el número de patrones de la clase 0 y de la clase 1 respectivamente. Además α define la relación de importancia sobre el ajuste de las etiquetas de los patrones de cada clase.

Al contener el parámetro α podemos modificar la función de coste para que tenga en cuenta las peculiaridades de los datos (desequilibrio) o del problema en sí, y de este modo mejorar las prestaciones en problemas como los comentados en las secciones 2.1.2 y 2.1.3.

Ahora la derivada parcial del coste respecto de la salida es:

$$\frac{\partial J^{WSE}(\mathbf{w})}{\partial o_k} = -2\ a_k\ (y_k - o_k),\ \text{con}\ a_k = \begin{cases} \frac{\alpha}{N_0} & \text{si} \quad y_k = -1 \\ \\ \frac{1-\alpha}{N_1} & \text{si} \quad y_k = +1 \end{cases} \tag{2.50}$$

2.4.3. Función de coste bayesiano

Recientemente se ha propuesto un método alternativo de entrenamiento para redes neuronales basado en la minimización de una aproximación de la función de riesgo bayesiano en (Lazaro et al., 2015).

El objetivo de este método consiste en encontrar los parámetros de la red neuronal que minimizan el riesgo de Bayes de la ecuación

$$R(\alpha) = \alpha\, p_{FA} + (1-\alpha)\, p_M. \tag{2.51}$$

El parámetro α se utiliza para ponderar la importancia de fallar o acertar sobre cada clase. Esto hace que sea un método adecuado para los problemas con datos desequilibrados o problemas con distintos costes.

Como aparece en (Lazaro et al., 2015) dada la regla de decisión de la ecuación (2.44), las probabilidades de falsa alarma y de pérdida que definen el riesgo de Bayes se escriben de la forma

$$p_{FA} = \int_0^\infty f_{O|H}(o|0)\ do,\ p_M = \int_{-\infty}^0 f_{O|H}(o|1)\ do. \tag{2.52}$$

Normalmente, las funciones de distribución condicionales en O, $f_{O|H}(o|0)$ y $f_{O|H}(o|1)$, son desconocidas. En (Lazaro et al., 2015) se propone estimar estas distribuciones con el conocido estimador no paramétrico de ventanas de Parzen (Parzen, 1962). A continuación se presentan las estimas obtenidas con este estimador.

Se definen los conjuntos S_0 y S_1 que agrupan los índices de los patrones de las etiquetas $H = 0$ y $H = 1$ respectivamente como

$$S_0 = \{k : y_k = -1\} \text{ y } S_1 = \{k : y_k = +1\}, \tag{2.53}$$

donde N_0 y N_1 son el número de patrones de la clase 0 y la clase 1 respectivamente. El estimador ventana de Parzen para la distribución condicional en O dado $H = h$ se expresa como

$$\hat{f}_{O|H}(o \mid h) = \frac{1}{N_h} \sum_{k \in S_h} K_\sigma(o - o_k),\ h \in \{0, 1\}. \tag{2.54}$$

La función ventana o núcleo $K_\sigma(o)$ es cualquier función de densidad de probabilidad con un parámetro σ que controla el ancho de la función. Por lo general, se usan funciones simétricas de media nula como la gaussiana. Ahora por tanto tendríamos las siguientes estimas de las probabilidades de falsa alarma y pérdida

$$\hat{p}_{FA} = \int_0^\infty \hat{f}_{O|H}(o \mid 0)\ do,\ \hat{p}_M = \int_{-\infty}^0 \hat{f}_{O|H}(o \mid 1)\ do. \tag{2.55}$$

Sustituyendo estas estimas en la función de riesgo bayesiano (2.51), se llega a la expresión de la función de coste bayesiana propuesta en (Lazaro et al., 2015)

$$J^{Bayes}(\mathbf{w}) = \alpha\hat{p}_{FA} + (1-\alpha)\hat{p}_M. \tag{2.56}$$

Si se define $L_\sigma(x)$ como la integral de la función ventana de Parzen se tiene

$$L_\sigma(x) = \int_x^{+\infty} K_\sigma(o)\ do, \tag{2.57}$$

donde teniendo en cuenta que la función núcleo es una función simétrica de media nula

$$\int_0^{+\infty} K_\sigma(x-o)\ dx = L_\sigma(-o) \text{ y } \int_{-\infty}^{0} K_\sigma(x-o)\ dx = L_\sigma(+o). \tag{2.58}$$

La función de coste por lo tanto se puede expresar como

$$J^{Bayes}(\mathbf{w}) = \frac{\alpha}{N_0}\sum_{k\in S_0} L_\sigma(-o_k) + \frac{1-\alpha}{N_1}\sum_{k\in S_1} L_\sigma(+o_k). \tag{2.59}$$

La derivada de la ecuación (2.59) con respecto a la salida de la red puede escribirse la forma

$$\frac{\partial J^{Bayes}(\mathbf{w})}{\partial o_k} = a_k\ K_\sigma(o_k),\ \text{con}\ a_k = \begin{cases} \frac{\alpha}{N_0} & \text{si} \quad y_k = -1 \\ \\ \frac{1-\alpha}{N_1} & \text{si} \quad y_k = +1 \end{cases} \tag{2.60}$$

Esta ecuación se introduciría en el primer término de la ecuación (2.46) para el algoritmo de adaptación por descenso de gradiente.

Si se compara con las ecuaciones (2.48) y (2.50), se observa que el coste computacional de este método no es mucho mayor que el asociado a las funciones de coste MSE y WMSE.

2.5. Conjunto de clasificadores bayesianos

El clasificador conjunto o conjunto de clasificadores bayesianos tiene como propósito entrenar N_{cc} clasificadores neuronales individuales con el procedimiento del método bayesiano de la sección 2.4.3 para diferentes valores de α. Cada clasificador trabajará en un punto diferente de la ROC (ver sección 2.6.2), es decir que tendrá una $\alpha^{(j)}$ con $j \in \{1, 2, \cdots, N_{cc}\}$. Cada uno por tanto proporciona un par de probabilidades de falsa alarma y detección distintas, en total N_{cc} puntos

$$\{(p_{FA}^{(j)}, p_D^{(j)})\} \text{ para } j \in \{1, 2, \cdots, N_{cc}\}. \tag{2.61}$$

Es decir que para clasificar un patrón de entrada $\mathbf{x}_k$, la entrada del clasificador conjunto es un vector formado por las decisiones de los N_{cc} clasificadores individuales para ese patrón en concreto. Por lo que la entrada para el clasificador conjunto se puede expresar de la forma

$$\mathbf{x}_k^E \equiv \mathbf{x}_k^E(\mathbf{x}_k) = [\hat{y}_k^{(1)}, \hat{y}_k^{(2)}, \cdots, \hat{y}_k^{(N_{cc})}], \tag{2.62}$$

donde $\mathbf{x}_k$ es el patrón de entrada de los clasificadores individuales e $\hat{y}_k^{(j)}$ determina la decisión binaria del clasificador j.

Se van a utilizar dos tipos de combinaciones de clasificadores individuales. El primero tomará la decisión en base a lo que decidan la mayoría de los clasificadores individuales y el segundo utilizará una regla bayesiana basada en las estimaciones de cada clasificador individual.

2.5.1. Combinación de clasificadores con regla de decisión por mayoría

El clasificador conjunto por mayoría tiene una implementación sencilla. Para un patrón de datos $\mathbf{x}_k$, su hipótesis tomará el valor

$$\hat{H}_k = \begin{cases} 1 & si \quad \sum_{j=1}^{N_{cc}} \hat{y}_k^{(j)} \geq 0 \\ \\ 0 & si \quad \sum_{j=1}^{N_{cc}} \hat{y}_k^{(j)} < 0 \end{cases} \tag{2.63}$$

El umbral se coloca en 0, ya que las posibles etiquetas son +1 y -1, si la suma de todas las etiquetas estimadas es mayor o igual que 0, implica que más de la mitad de las estimas han tomado la decisión de $\hat{H} = 1$, por el contrario, si la suma es menor la mayoría de los clasificadores han decidido $\hat{H} = 0$.

De esta forma la decisión que tome el clasificador conjunto será la que haya tomado la mayoría de sus clasificadores individuales, ocasionando así el ignorar la decisión de clasificadores que a causa de su punto de trabajo no clasifican correctamente.

2.5.2. Combinación de clasificadores con regla de decisión basesiana

Este clasificador conjunto pretende juntar las decisiones de los N_{cc} clasificadores individuales usando la regla de Bayes.

Las distribuciones condicionales de la decisión de cada clasificador están dadas por

$$p^{(j)}_{\hat{Y}|H}(\hat{y}^{(j)}_k|0) = \begin{cases} p^{(j)}_{FA} & si \quad \hat{y}^{(j)}_k = +1 \\ \\ 1 - p^{(j)}_{FA} & si \quad \hat{y}^{(j)}_k = -1 \end{cases} \tag{2.64}$$

y

$$p^{(j)}_{\hat{Y}|H}(\hat{y}^{(j)}_k|1) = \begin{cases} p^{(j)}_{D} & si \quad \hat{y}^{(j)}_k = +1 \\ \\ 1 - p^{(j)}_{D} & si \quad \hat{y}^{(j)}_k = -1 \end{cases} \tag{2.65}$$

El cociente de verosimilitud (*likelihood ratio*) $\Lambda(\mathbf{x}^E_k)$ en función de las distribuciones condicionales anteriores se puede definir como

$$\Lambda(\mathbf{x}^E_k) = \prod_{j=1}^{N_{cc}} \frac{p^{(j)}_{\hat{Y}|H}(\hat{y}^{(j)}_k|1)}{p^{(j)}_{\hat{Y}|H}(\hat{y}^{(j)}_k|0)} = \prod_{j=1}^{N_{cc}} \frac{p^{(j)}_{D}\ \delta[\hat{y}^{(j)}_k - 1] + (1 - p^{(j)}_{D})\ \delta[\hat{y}^{(j)}_k + 1]}{p^{(j)}_{FA}\ \delta[\hat{y}^{(j)}_k - 1] + (1 - p^{(j)}_{FA})\ \delta[\hat{y}^{(j)}_k + 1]}, \tag{2.66}$$

donde se asume independencia condicional entre las salidas de los clasificadores individuales. La función discreta delta $\delta[n]$ se usa para reducir las ecuaciones (2.64) y (2.65).

Por último, para la toma de la decisión final sobre la entrada $\mathbf{x}^E$, se compara el cociente de verosimilitud con el umbral γ asociado al valor de α definido por el riesgo de Bayes del clasificador conjunto, es decir α^E

$$\Lambda(\mathbf{x}^E) \underset{\hat{H}=0}{\overset{\hat{H}=1}{\gtrless}} \gamma^E, \text{ con } \gamma^E = \frac{\alpha^E}{1 - \alpha^E}. \tag{2.67}$$

Este proceso se puede visualizar mediante la figura 2.11, donde aparecen representados los distintos pasos con las entradas y salidas correspondientes a cada clasificador.

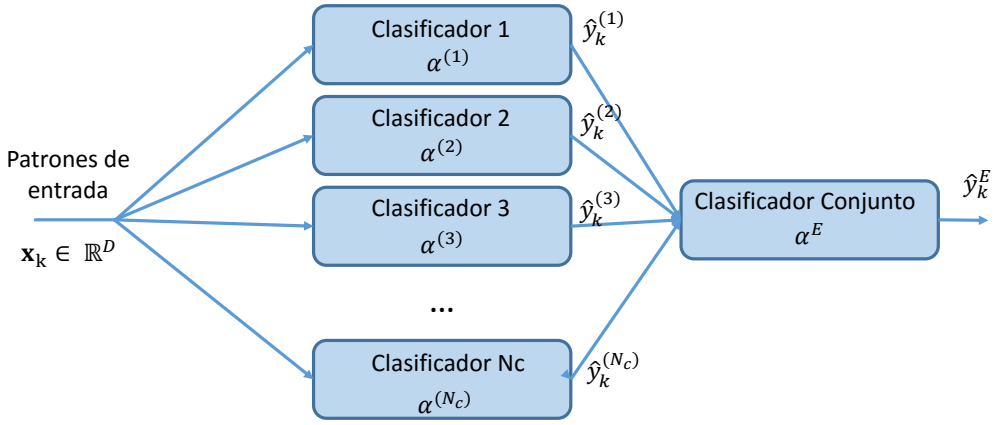

Figura 2.11: Esquema del clasificador conjunto planteado.

2.6. Análisis de prestaciones

A la hora de analizar las prestaciones de un clasificador binario se tienen distintas formas de hacerlo. A continuación se exponen algunas de las más importantes.

2.6.1. Tabla de verdad

Como se mencionó en la sección 2.1.1, en un problema de clasificación binaria existen dos probabilidades que determinan de forma completa las prestaciones, éstas son la probabilidad de falsa alarma p_{FA} y la probabilidad de pérdida p_M. Con ambas se puede representar en una tabla las cuatro posibles probabilidades de un clasificador con dos hipótesis (H_0, H_1). Dicha tabla se suele denominar tabla de verdad.

En la figura 2.12 se muestran las cuatro probabilidades en función de p_{FA} y p_M. La y define la etiqueta conocida y la $\hat{y}$ es la etiqueta estimada por el clasificador. También podría sustituirse la probabilidad de acierto $1 - p_M$, por p_D, que como se mencionó anteriormente son equivalentes. Esta p_D es muy útil junto con p_{FA} para la curva que se va a ver a continuación.

2.6.2. Curva característica de operación (ROC)

La curva característica de operación (ROC: *Receiver Operating Characteristic*) es una gráfica que representa la relación entre la probabilidad de

	$y = 0$	$y = 1$
$\hat{y} = 0$	$1 - p_{FA}$	p_M
$\hat{y} = 1$	p_{FA}	$1 - p_M$

Figura 2.12: Tabla de verdad en un problema de clasificación binario.

falsa alarma y la probabilidad de detección. Cada clasificador tiene un punto de trabajo que está definido por una p_{FA} y una p_D. Es por ello por lo que inicialmente era una herramienta de análisis para clasificadores estadísticos (Van Trees, 1968), ya que estaba relacionado por sus propiedades con la ecuación (2.51). Posteriormente se expandió su uso a otros clasificadores como los neuronales.

Las redes neuronales (entre otros clasificadores) para tomar la decisión comparan la salida de la red para un patrón de entrada con un umbral γ. Para cada valor de γ existe un punto de trabajo (p_{FA}, p_D). Al ser modificado el umbral se cambian las prestaciones de cada clase, de forma que al aumentarse el valor del umbral, la probabilidad de falsa alarma y de detección disminuyen, y al disminuir el umbral, ambas probabilidades aumentan. De esta manera si el umbral varía entre $-\infty$ e ∞ el punto de trabajo se va desplazando formando la curva ROC, desde el punto $(0, 0)$ al punto $(1, 1)$.

En la figura 2.13 se muestra una curva ROC. En línea roja discontínua se muestra el clasificador trivial, que consiste en la toma aleatoria de decisión, donde en un número suficientemente elevado de muestras, la probabilidad de falsa alarma es igual a la probabilidad de detección. Un ejemplo de clasificador trivial sería decidir tirando una moneda al aire, existe la misma probabilidad de acertar que de fallar.

En línea contínua de color verde se muestra un clasificador no trivial, donde el punto verde representa un punto de trabajo, es decir, una pareja (p_{FA}, p_D). Según se va modificando γ, el punto se va desplazando dibujando la línea verde.

Por debajo del clasificador trivial no tiene sentido encontrar un punto de trabajo, porque en ese caso bastaría con cambiar la decisión para que el clasificador se encontrase por encima del trivial, y de esa forma aumentar las

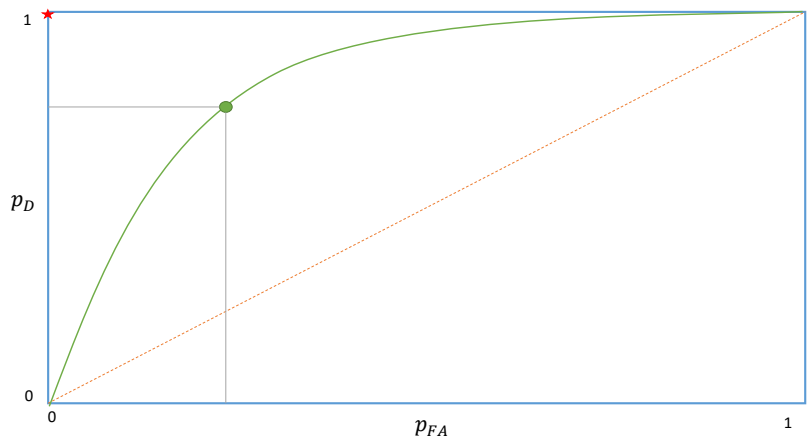

Figura 2.13: Ejemplo curva ROC.

prestaciones.

El punto de trabajo que maximiza la probabilidad de acierto aparece representado en la figura 2.13 como una estrella roja, reproduce el punto (0,1), donde no se tiene ninguna falsa alarma, y la probabilidad de detección es máxima.

La ROC tiene un gran uso en análisis de prestaciones en aprendizaje máquina. Motivo por el cual es una buena herramienta para comparar distintos clasificadores. Para más información consultar en (Fawcett, 2006).

2.6.3. Área debajo de la curva (AUC)

El área debajo de la curva (AUC: *Area Under Curve*), como su nombre indica es el área que se encuentra debajo de la curva ROC. En la ROC a simple vista de forma gráfica, puede tener más dificultades para ver en conjunto cuál es el mejor clasificador si hay tramos en los que un clasificador está por encima, y en otros en los que otro clasificador es mejor. Sin embargo la AUC, al ser un valor escalar es más directo a la hora de comparar.

La AUC está comprendida entre 0 y 1, aunque no tiene mucho sentido un valor por debajo de 0.5, ya que como se indicó en el apartado anterior, en ese caso se realizaría un cambio en la decisión de modo que la AUC ya estaría por encima de 0.5 (es decir, por encima del clasificador trivial). El valor de AUC que corresponde a la estrella roja de la figura 2.13 sería 1, donde la probabilidad de acertar será máxima.

En la figura 2.14 en color verde se observa un ejemplo del área debajo de la curva. Junto con la curva ROC, son las dos medidas de comparación

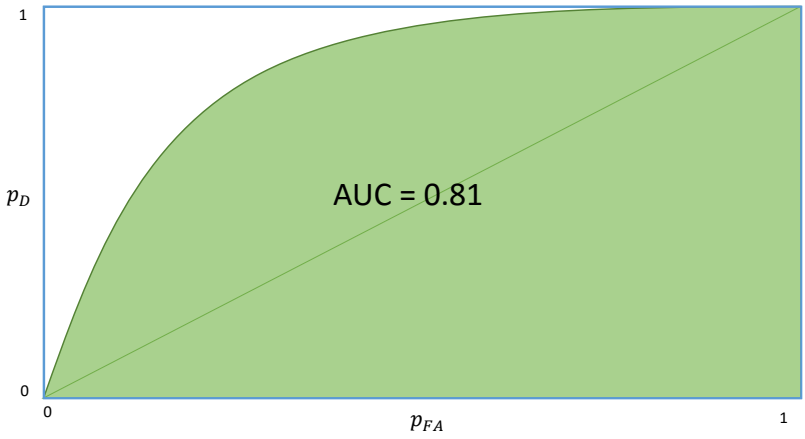

Figura 2.14: Ejemplo curva ROC y AUC.

más frecuentes en algoritmos de aprendizaje máquina. Ambas van a ser usadas en el presente proyecto por las características mencionadas. Para más información sobre la AUC consultar (Bradley, 1997).

Capítulo 3

Soluciones evaluadas y metodología de trabajo

En este capítulo se van a presentar las soluciones que se han evaluado en el trabajo para resolver el problema. Posteriormente se describirá la metodología utilizada para llevarlo a cabo.

3.1. Planteamiento de las soluciones analizadas

El trabajo está enfocado a la solución de un problema general de clasificación de patrones con datos desequilibrados. No se va a tratar una aplicación determinada, sino que se analizará sobre bases de datos reales provenientes de distintos fenómenos. Más concretamente, se realizará sobre problemas de clasificación binaria.

Como se mencionó en la sección 2.1.3 hay distintas soluciones para estos problemas. Modificar el número de muestras añadiendo o eliminando patrones de las distintas clases, modificando métodos de aprendizaje máquina para tener en consideración el número de muestras de cada hipótesis, y por último la combinación de clasificadores individuales.

En este trabajo, no se van a modificar los datos ni insertando muestras sintéticas ni eliminándolas. Por lo que se valorarán el resto de posibilidades mencionadas. En particular, se van a evaluar las prestaciones de los siguientes métodos:

1. Se evaluará un método basado en aprendizaje a partir de la función de coste del error cuadrático medio ponderado, que se ha visto en la

sección 2.4.2 aplicado sobre una red neuronal perceptrón multicapa (ver sección 2.4).

2. Sobre el mismo tipo de red neuronal, MLP con una capa oculta, se evaluará el algoritmo de entrenamiento basado en la formulación bayesiana vista en la sección 2.4.3.

3. Por último, se evaluarán también las prestaciones en este tipo de problemas de los combinadores de clasificadores. En concreto se evaluará en base a dos reglas de decisión distintas, una basada en la elección de la decisión por mayoría, y otra en la regla de Bayes (ver sección 2.5).

A continuación se explicarán brevemente las razones por las que se han considerado estos métodos para la solución de problemas desequilibrados de clasificación binaria.

La primera decisión fue trabajar con métodos que no se basan en la modificación de los datos disponibles (generando muestras sintéticas de la clase minoritaria o eliminando muestras de la clase mayoritaria), sino que tengan en cuenta el desequilibrio en los datos disponibles. La principal razón para esta elección es que la literatura (Galar et al., 2013) ha demostrado que la elección de las técnicas de equilibrado de los datos depende en gran manera de las características de los datos. Esto implica en la práctica tener que evaluar varias alternativas, lo que incrementa el coste computacional. En la misma línea, las técnicas que utilizan generación sintética de muestras, al aumentar el tamaño del conjunto de entrenamiento aumenta también la carga computacional.

Comparando el método tradicional MSE y su ecuación (2.47), con el método WMSE de la ecuación (2.49), se observa que el WMSE al contrario que el MSE tiene en cuenta por separado el número de muestras de cada clase. De esta forma y a través del parámetro de ponderación α, es posible hacer que los patrones de la clase minoritaria contribuyan en buena medida sobre el coste final. El parámetro α de la ecuación (2.49) nos permite ajustar de forma flexible la importancia sobre el coste de cada clase, ponderando el peso relativo del ajuste de la salida de la red a las etiquetas correspondientes a patrones de las dos clases. Todo ello hace que el WMSE sea a priori un método más adecuado para la clasificación de patrones con desbalanceo que el convencional MSE.

Por otro lado, la función de coste bayesiano expuesta en la sección 2.4.2 posee en su ecuación (2.51) un parámetro α que pondera la importancia que se da a la probabilidad de error sobre la clase nula (p_{FA}), y a la probabilidad de error sobre la clase positiva (p_M). Este parámetro da la posibilidad en los problemas desequilibrados o con costes descritos en las secciones 2.1.2 y 2.1.3 ajustarse a sus particularidades, permitiendo definir un compromiso

apropiado entre las tasas de error para patrones de cada clase. Por tanto, también parece a priori un método adecuado para la resolución del tipo de problemas considerados en este trabajo.

El último lugar, se utilizará la combinación de clasificadores neuronales individuales. La combinación de clasificadores en la práctica ha demostrado mejoras en las prestaciones de problemas tanto balanceados como desbalanceados. Prueba de ello son los resultados presentados, por ejemplo en artículos como (Galar et al., 2013) o (Lazaro et al., 2015). Motivación que ha llevado a la evaluación de tales clasificadores.

El objetivo del trabajo es evaluar las prestaciones de estos métodos en varios problemas diferentes y compararlos con los obtenidos con otro tipo de métodos. Como se ha dicho, los algoritmos de entrenamiento se implementarán en un modelo de aprendizaje máquina, más concretamente en una red neuronal MLP con una única capa oculta, y una sola neurona en la capa de salida.

La elección de la red neuronal se ha tomado en base a que las funciones de coste que se van a analizar son más fáciles de implementar en una red neuronal que en una red tipo SVM. Los árboles de decisión se han descartado por ser clasificadores más débiles (más sensibles a cambios en el conjunto de entrenamiento, lo que puede ser especialmente problemático en problemas con un número reducido de patrones de entrenamiento) y porque habitualmente sólo se utilizan en combinadores de clasificadores, y no como clasificadores individuales. Por lo que teniendo en cuenta esto, y las ventajas y desventajas de cada red explicadas en las secciones 2.3.1 y 2.3.3 se ha optado por la red neuronal. Dentro de las redes neuronales se escoge la MLP en lugar de otras como la RBF, porque es de las redes más utilizadas en la práctica, entre otras razones por su conocida capacidad de aproximador universal (Hornik et al., 1990) (Cybenko, 1989).

3.2. Metodología de trabajo

A partir de un conjunto de bases de datos reales, se han entrenado los pesos de la red MLP con cada una de las metodologías expuestas anteriormente, y se han evaluado las prestaciones con algunas de las herramientas de análisis presentadas en la sección 2.6. En la figura 3.1 se representa un esquema de las fases básicas de la metodología seguida para la evaluación de los distintos métodos.

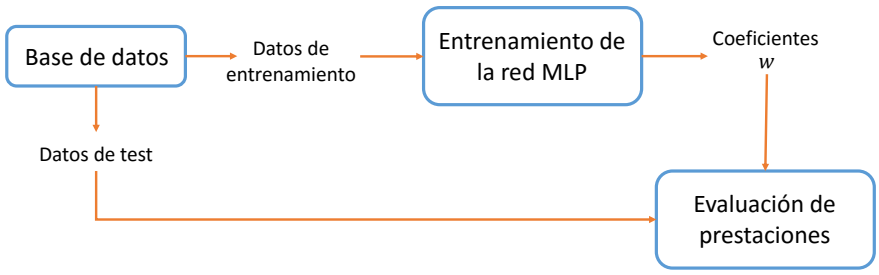

Figura 3.1: Esquema básico de la metodología para la evaluación de los distintos métodos de clasificación.

Las herramientas software utilizadas en el trabajo se han desarrollado en MATLAB. Se crearon todas las funciones principales de la red MLP y cada método de entrenamiento. Además se utilizaron algunas funciones básicas internas de MATLAB.

A continuación se presentan los aspectos principales de cada fase.

3.2.1. Base de datos

Se va a utilizar un conjunto de 8 bases de datos desbalanceados que se han obtenido del repositorio KEEL (Alcalá-Fdez et al., 2011). En la página web `http://www.keel.es/dataset.php` se puede encontrar información sobre las distintas bases de datos y sus características. Los datos comúnmente se dividen en dos conjuntos. Una parte de los datos se asocian al entrenamiento de los parámetros de la red y se suelen denominar conjunto de entrenamiento. Luego hay una segunda parte de datos que sirven para comprobar cómo se clasifica con datos que no han sido utilizados para entrenar la red, éstos se denominan conjunto de test. Ambos están etiquetados.

En el repositorio KEEL las bases de datos se encuentran organizadas en k-particiones, cada una de ellas con una separación diferente de los datos en conjunto de entrenamiento y conjunto de test, y en el caso concreto de este trabajo el número de particiones será 5. Es interesante el uso de estas bases de datos, porque son utilizadas en otros artículos como (Galar et al., 2013) y pueden ayudar para comparar resultados. En la tabla 3.1 se muestran las

características principales de las 8 bases de datos del trabajo, número de patrones o muestras, dimensionalidad y tasa de desequilibrio.

Base de datos	Número de patrones	Dimensión	Tasa de desequilibrio
Glass2	214	9	10.39
Ecoli067vs5	220	6	10.00
Yeast05679vs4	528	8	9.35
Cleveland0vs4	177	13	12.62
Led7digit0245678avs1	443	7	10.97
Yeast4	1484	8	28.41
Yeast5	1484	8	32.78
Yeast6	1484	8	39.15

Tabla 3.1: Bases de datos.

Hay que tener en cuenta que el número de patrones en algunas bases de la tabla 3.1 como la Cleveland0vs4 por ejemplo es bastante pequeño. Si las 177 muestras se dividen en dos grupos de entrenamiento y test por ejemplo con un porcentaje 60/40, 106 muestras serían de entrenamiento y 74 de test. Y luego a su vez con una tasa de desequilibrio del 12.62, sólo se tendrían en el subgrupo de entrenamiento 8 muestras de la clase minoritaria frente a las 98 de la mayoritaria. Es importante observar que la red en ese caso se entrenaría únicamente con 8 muestras de una clase, existiendo el peligro de que la red se sobreajuste y luego no consiga generalizar sobre los datos de test o nuevos datos sin etiquetar.

La elección de estas bases de datos también se ha escogido en función de que aun siendo todas de tamaño no muy elevado, entre ellas sí poseen diferente número de patrones. La base con menos muestras tiene 177 patrones, y las más grandes 1484 patrones. La mayor dificultad se encontrará en las bases más pequeñas, por lo mencionado anteriormente.

Dentro de cada base, la clase minoritaria estará representada por la etiqueta +1 (hipótesis 1), y la clase mayoritaria por la etiqueta -1 (hipótesis 0).

3.2.2. Entrenamiento de la red

Esta fase es la encargada de entrenar la red con el conjunto de datos de entrenamiento para obtener los pesos $\mathbf{w}$ que minimizan la función de coste a través del algoritmo adaptativo de descenso por gradiente.

Se realizará el mismo proceso en las distintas particiones de los datos, proporcionando así diversidad. Posteriormente se hará la media de los resultados en las 5 particiones.

Además, para intentar solventar el problema de los mínimos locales explicado en la sección 2.3.3, se realizarán distintas simulaciones, es decir distintas inicializaciones de los pesos de la red, y se presentarán resultados promedio.

3.2.3. Cálculo de prestaciones

Una vez entrenadas las redes neuronales, y obtenidos los parámetros $\mathbf{w}$, la evaluación de cada solución se realiza calculando las prestaciones de dicha solución sobre el conjunto de test.

En este caso se realiza la fase de entrenamiento y de evaluación en cada una de las particiones por cada base de datos, presentando los resultados promedio. Se han utilizado varias figuras de mérito para evaluar las prestaciones: la curva ROC y el área debajo de dicha curva presentados en la sección 2.6.

Se utilizarán estas medidas por el hecho de ser las más comunes para el cálculo de prestaciones en la clasificación de patrones binaria.

Capítulo 4

Resultados

En este capítulo se realizará la evaluación y se expondrán los resultados de los métodos WMSE, bayesiano y combinación de clasificadores sobre una red MLP. La variedad de posibles implementaciones dentro de estos métodos es alta, ya que estos tienen muchos parámetros que los condicionan. Por cuestiones de dimensionalidad del Trabajo Fin de Grado, sólo se llevarán a cabo algunas de ellas, siendo especificadas las condiciones dadas para cada implementación.

Las soluciones tendrán como objetivo comparar las distintas técnicas con algunas de las herramientas propuestas en la sección 2.6.

Primero se analizarán los parámetros utilizados en la red MLP y posteriormente se mostrarán los distintos resultados obtenidos.

4.1. Detalles sobre la implementación

Antes de mostrar los resultados, se van a indicar los parámetros, funciones y demás especificaciones necesarias sobre la red neuronal que se va a utilizar.

El objetivo de la elección de los parámetros es aumentar las prestaciones en los resultados de diferentes conjuntos de datos, y evitar aspectos como el problema de mínimos locales de las redes neuronales.

Lo primero a tener en cuenta es que el único pre-procesado que se le va a realizar al conjunto de bases de datos es la normalización. No se realizará ningún otro proceso previo.

La red MLP que se va a utilizar, como ya se mencionó anteriormente, tendrá una única neurona en la capa de salida, y en la capa oculta se compondrá por N_n neuronas. Este valor será $N_n = 6$ ya que en conjunto ofrece una buena relación entre complejidad y prestaciones.

En cuanto a las funciones de activación de la red neuronal explicadas en la sección 2.3.3, se ha optado por escoger la función tangencial hiperbólica, debido a que es la función más común.

El algoritmo de descenso por gradiente cuyas expresiones se pueden encontrar en la sección 2.4, tras cada iteración o época [1] compara el coste en la época i, $J(\mathbf{w}^{(i)})$ con el coste de la época anterior $J(\mathbf{w}^{(i-1)})$, de tal forma que el parámetro μ se incremente o decrezca según la siguiente condición:

- Si $J(\mathbf{w}^{(i)}) < J(\mathbf{w}^{(i-1)})$: el paso aumentá de la forma $\mu = c_I\ \mu$.
- Si $J(\mathbf{w}^{(i)}) \geq J(\mathbf{w}^{(i-1)})$: el paso decrecerá de la forma $\mu = \mu/c_D$.

donde los valores para incrementar o disminuir el parámetro μ son $c_I = 1.05$ y $c_D = 2$. Los valores no tienen por qué ser estos exactamente, pero han sido escogidos en base al artículo (Lazaro et al., 2015) que utiliza las mismas bases de datos. Además el valor inicial del paso se fija en $\mu = 1$, es importante que este valor no sea ni muy alto ni muy pequeño para que el número de iteraciones necesarias sea lo menor posible.

Además, en cuanto al método bayesiano, se necesita detallar qué ventana de Parzen se va a utilizar para la estimación de las funciones de distribución condicionales. La ventana escogida se muestra a continuación

$$K_\sigma(o) = \begin{cases} \frac{1}{2\sigma}(1 + \cos(\frac{\pi}{\sigma}|o|)), & si \quad |o| \leq \sigma \\ \\ 0, & si \quad |o| > \sigma \end{cases} \tag{4.1}$$

donde el parámetro $\sigma = 1$ determina el ancho de la ventana.

Se utiliza esta ventana de Parzen porque es muy similar a la gaussiana (una de las más comunes), pero con soporte finito. El hecho de que sea finito tiene la ventaja de que el gradiente toma valor 0 cuando se encuentra fuera del rango $\{-\sigma, +\sigma\}$, lo que reduce la carga computacional. La función de la ecuación (4.1) se muestra en la figura 4.1.

En la figura 4.2 se observa un ejemplo donde se tienen 4 muestras de la clase positiva en distintos puntos de un espacio unidimensional, cuya frontera de decisión se encuentra en 0. Sobre cada muestra se visualiza una ventana de Parzen con $\sigma = 1$. La muestra que se encuentra a la derecha del +1 no tiene área de su núcleo en la zona de la clase negativa, mientras que la muestra que se encuentra entre -1 y 0 está situada en la zona opuesta de su etiqueta y por tanto tiene gran parte del área de su núcleo en la zona negativa. Este área de los núcleos que se encuentran en el lado contrario está dibujado en la figura como líneas verticales contínuas, y la integral de dicha área es

[1] Una época está formada por un número de iteraciones de gradiente descendente cuyos pesos iniciales son los mismos.

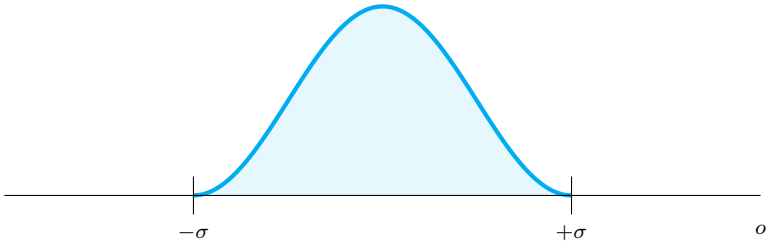

Figura 4.1: Ventana de Parzen.

la que describe la p_M. Como esta probabilidad se quiere minimizar, lo que se pretende es que los parámetros de la red 'empujen' las muestras al lado contrario para disminuir entonces ese área. Esto lleva a la decisión de cuál es el valor óptimo del parámetro σ. Cuando el valor de σ es muy alto, más a la derecha debe estar la muestra para que la red minimice la p_M, y cuando el valor es muy pequeño, apenas basta con que la muestra esté en el lado correcto de la decisión.

En este trabajo se ha escogido $\sigma = 1$ para que se traten de 'empujar' hacia el lado correcto todas las muestras que estén en el rango ± 1.

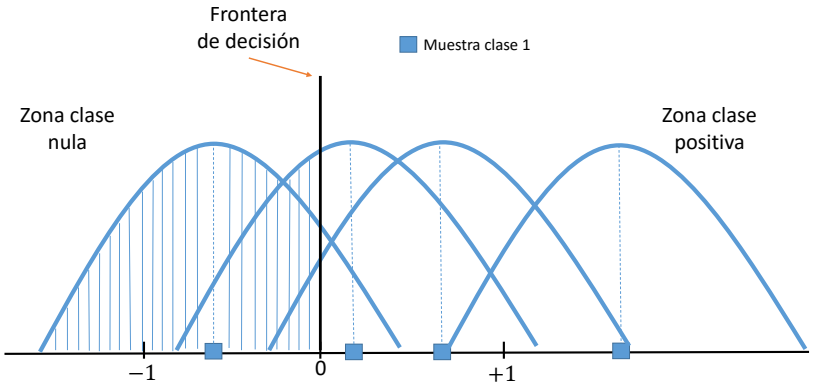

Figura 4.2: Ejemplo de función núcleo sobre muestras de la clase positiva.

Por último para solventar el problema de mínimos locales descrito en la sección 2.3.3, se van a realizar distintas simulaciones, de forma que cada simulación parte de unos pesos iniciales diferentes. Esto lleva a que el algoritmo de descenso por gradiente comience en distintos puntos de la función de coste dando la posibilidad de encontrar el mínimo global de dicha función. Para que los resultados obtenidos tengan una fiabilidad estadística suficiente se

realizarán en cada prueba 100 simulaciones distintas, y se promediarán los resultados obtenidos en las mismas.

A continuación se muestran los distintos resultados obtenidos.

4.2. Resultados con $\alpha = 0.5$ en método WMSE y método bayesiano

Primero se van a comparar los métodos WMSE y bayesiano cuando el parámetro α de sus funciones de coste es igual a 0.5. Lo que indica que ponderamos por igual la importancia de las muestras de cada clase. Esta comparación se realiza para observar los resultados de ambos métodos cuando las dos clases se consideran igual de relevantes en las prestaciones.

Se realiza el entrenamiento de la red con cada método, y posteriormente se evalúan sobre los datos de test. Para evaluar los métodos en este caso usaremos la medida del AUC. Los resultados para las 8 bases de datos se muestran en la tabla 4.1.

Datos	1	2	3	4	5	6	7	8
Bayes	**0.9163**	**0.8924**	**0.8372**	**0.9737**	0.9119	**0.8774**	**0.9872**	**0.9411**
WMSE	0.812	0.8385	0.7211	0.8906	**0.9832**	0.7164	0.7847	0.9006

Tabla 4.1: Resultados AUC de los métodos WMSE y Bayes para las 8 bases de datos con $\alpha = 0.5$.

Según los resultados obtenidos, el método bayesiano ofrece mejores prestaciones que el método WMSE en todas las bases de datos exceptuando la base 5. Esta base es la que menor número de muestras posee, y es la base que proporciona mejores resultados en el WMSE con respecto al resto de bases.

La variación del AUC en función de las bases de datos es más destacable en el WMSE, cuya variación es de 0.2668, sin embargo en la metodología bayesiana la diferencia es de 0.15. El WMSE es por tanto más sensible a la variabilidad de las 8 bases de datos.

A la vista de los resultados se optaría por escoger el método WMSE para la base 5, y el método bayesiano para el resto de ellas.

Tras esto, se va a analizar a continuación lo que ocurre cuando utilizamos distintos valores de α, para obtener las prestaciones de los clasificadores al dar mayor o menor importancia a las etiquetas de cada clase.

4.3. Resultados comparativos sobre método WMSE y método bayesiano

De nuevo vamos a entrenar la red con los métodos bayesiano y WMSE. Pero en esta ocasión el parámetro α tomará distintos valores, concretamente 9, $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$.

Las prestaciones se evaluarán a través de la curva ROC, y el AUC. Los cuales se han calculado 'barriendo' el umbral y realizando el promedio.

Es importante entender que los resultados tratan de mostrar las prestaciones generales del método, ya que dependiendo de cada aplicación interesará un valor de α concreto para cumplir con los requisitos específicos. Dicho esto, de la figura 4.3 a la 4.10 se muestran las curvas ROC de cada una de las bases en ambos métodos. La evaluación se ha realizado tanto para los datos de entrenamiento como los datos de test. Además en la tabla 4.2 se muestra el AUC de todas las bases de datos para los dos métodos.

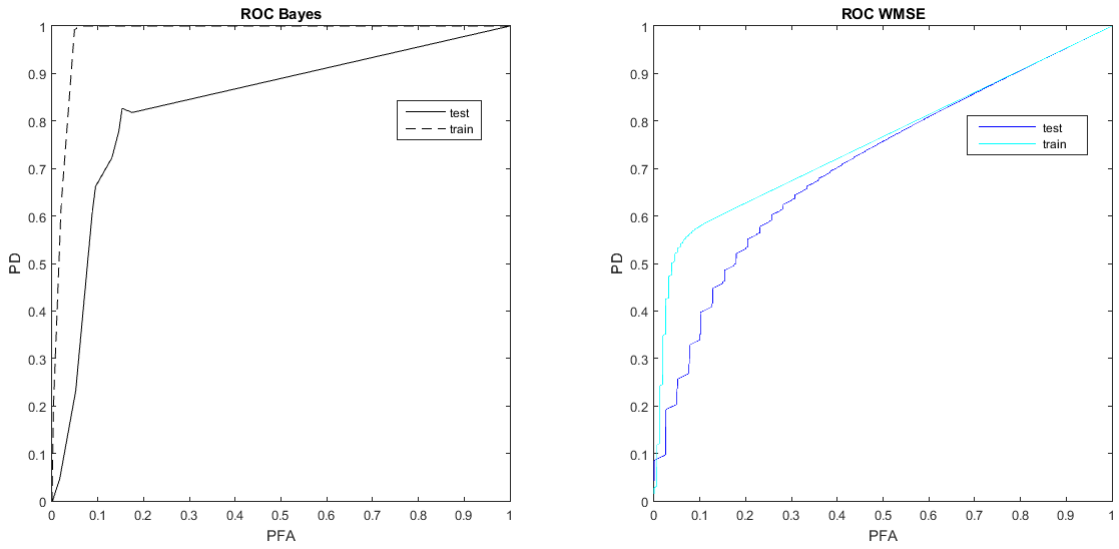

Figura 4.3: Curvas ROC resultantes de la base de datos 1 para los métodos WMSE y bayesiano.

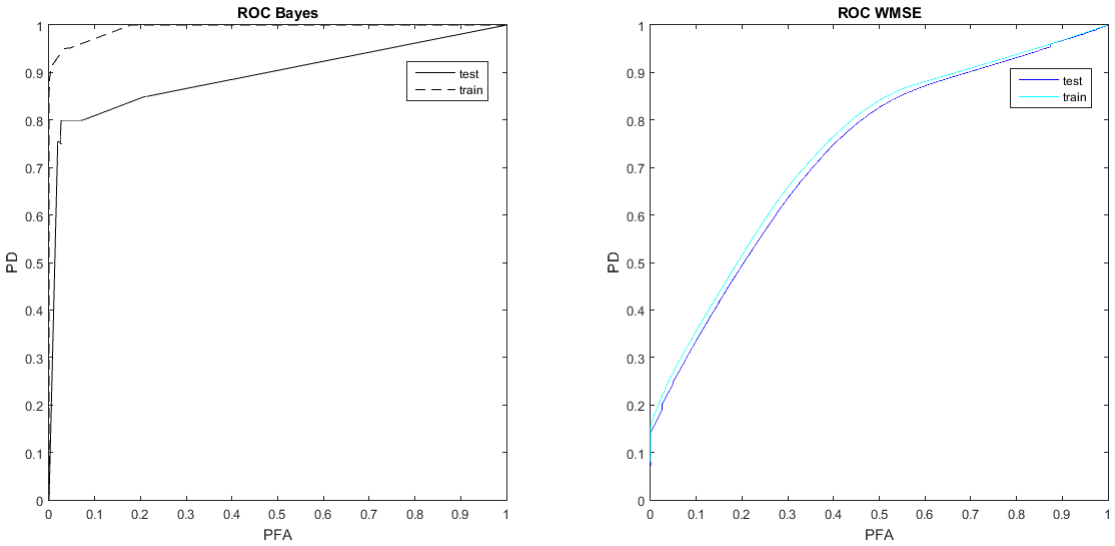

Figura 4.4: Curvas ROC resultantes de la base de datos 2 para los métodos WMSE y bayesiano.

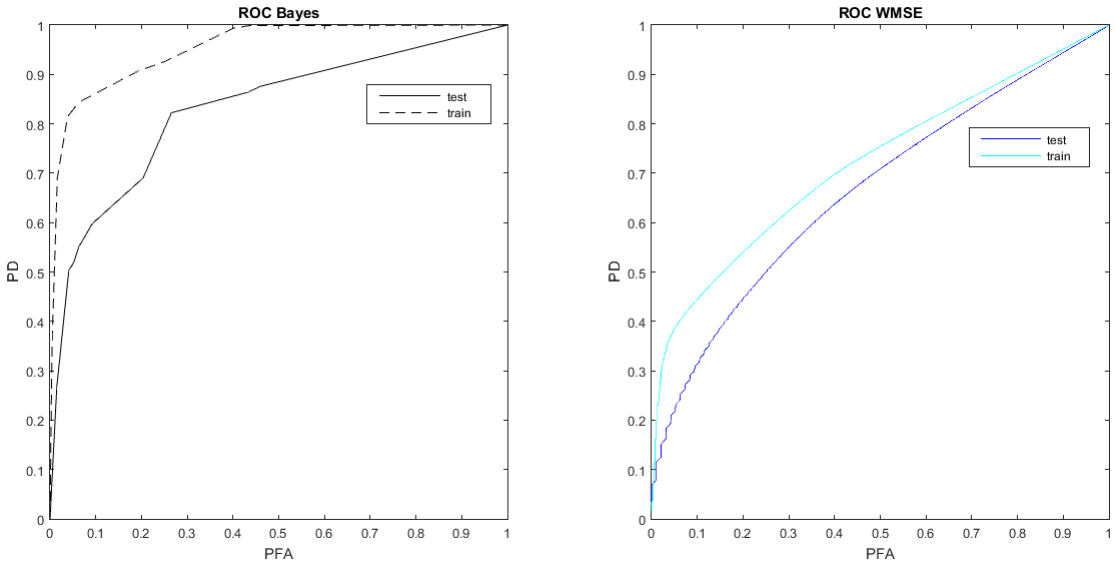

Figura 4.5: Curvas ROC resultantes de la base de datos 3 para los métodos WMSE y bayesiano.

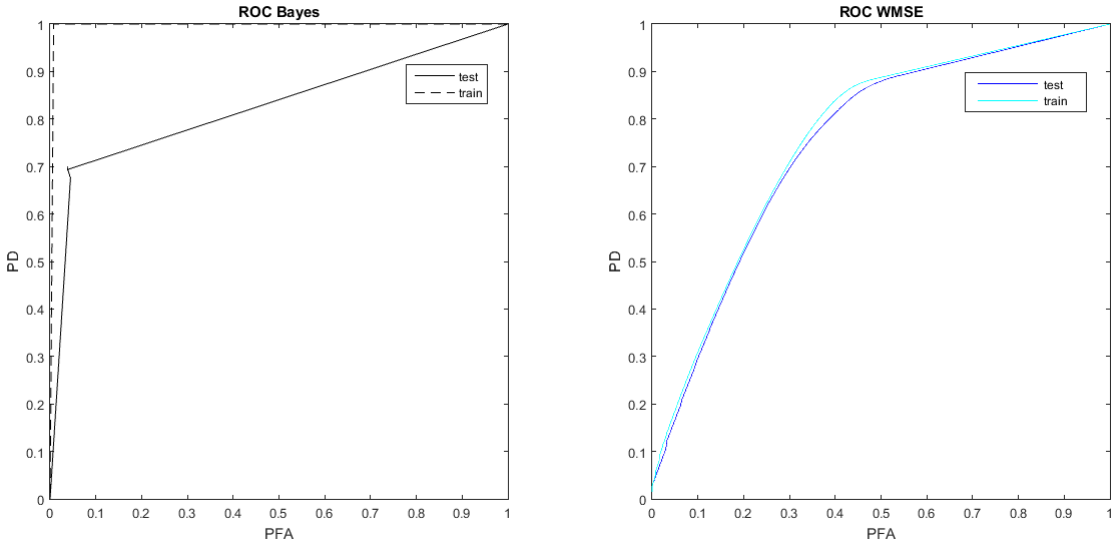

Figura 4.6: Curvas ROC resultantes de la base de datos 4 para los métodos WMSE y bayesiano.

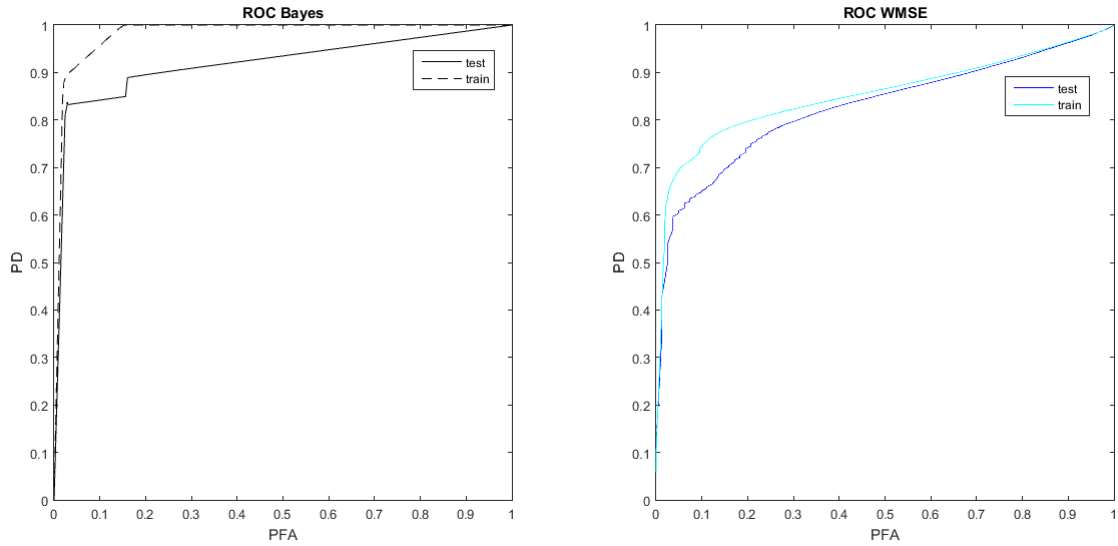

Figura 4.7: Curvas ROC resultantes de la base de datos 5 para los métodos WMSE y bayesiano.

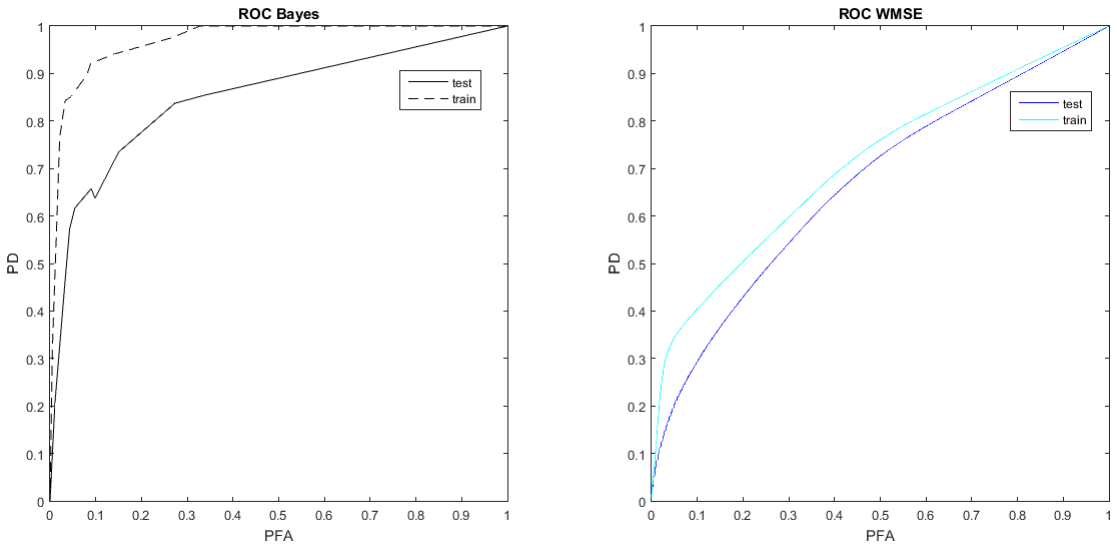

Figura 4.8: Curvas ROC resultantes de la base de datos 6 para los métodos WMSE y bayesiano.

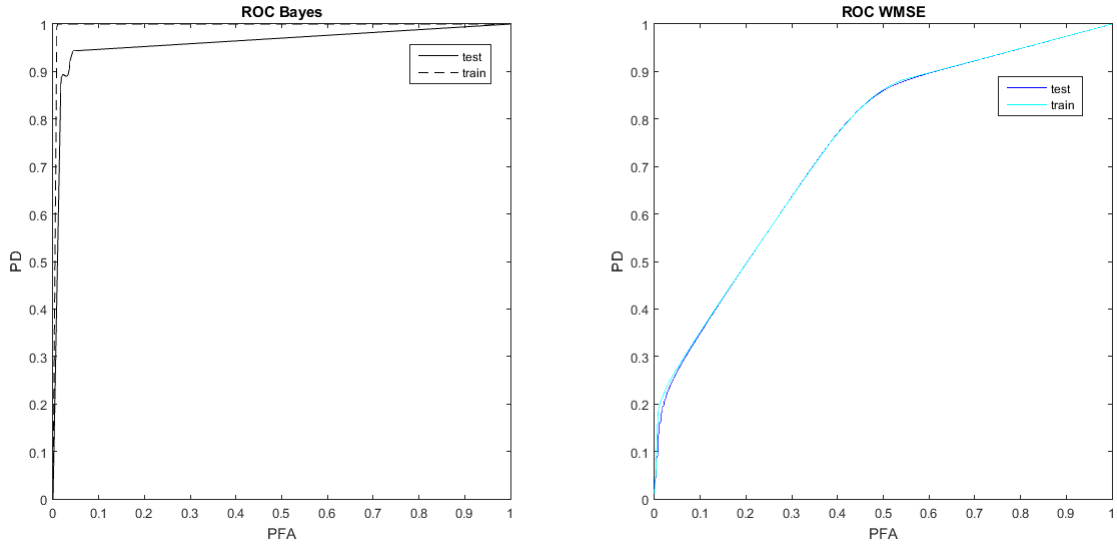

Figura 4.9: Curvas ROC resultantes de la base de datos 7 para los métodos WMSE y bayesiano.

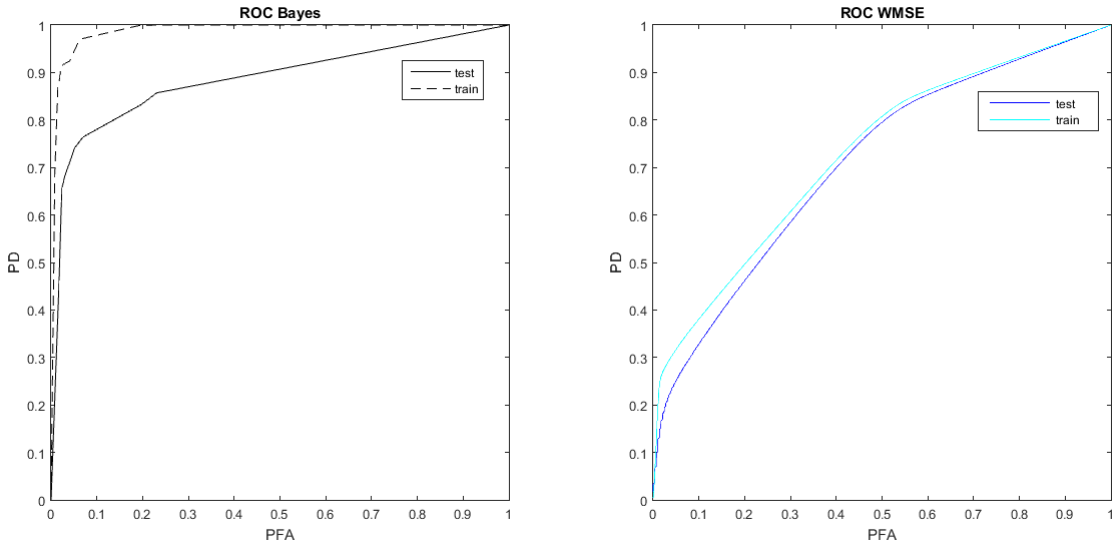

Figura 4.10: Curvas ROC resultantes de la base de datos 8 para los métodos WMSE y bayesiano.

Datos	1	2	3	4	5	6	7	8
Bayes	**0.8068**	**0.8715**	**0.8194**	**0.9728**	**0.9355**	**0.8666**	**0.9877**	**0.9216**
WMSE	0.7078	0.7296	0.6652	0.7502	0.8274	0.6658	0.7441	0.7085

Tabla 4.2: Resultados AUC de los métodos WMSE y Bayes para las 8 bases de datos.

Con los resultados obtenidos en la tabla 4.2, se comprueba que el método bayesiano logra mejores prestaciones en todas las bases de datos. La varianza de AUC de los métodos en las distintas bases de datos es similar, para el caso del WMSE es de 0.1622 y para el caso bayesiano de 0.1809.

Por otro lado, de la figura 4.3 a la figura 4.10, se contempla que en el método WMSE se sobreajusta menos, ya que la diferencia entre las prestaciones sobre los datos de entrenamiento y test son muy similares. Sin embargo en el método bayesiano, por lo general hay una mayor separación entre las curvas de entrenamiento y test, lo que implica que se produce mayor sobreajuste en el entrenamiento.

Por lo tanto, la clasificación alcanza mejores rendimientos con el uso del método bayesiano.

4.4. Comparación α distinta o α igual recorriendo umbral

Ahora, se va a proceder a comparar 2 pares de curvas ROC. La primera pareja de curvas pertenece al mejor clasificador de la sección 4.3, es decir el bayesiano. Con las especificaciones de dicha sección, realizando el valor medio de los resultados del clasificador para todos los valores de α. El par va a estar formado por la curva ROC sobre los datos de entrenamiento y la curva ROC sobre los datos de test.

La otra pareja de curvas se ha construido a través del entrenamiento de la red MLP con el método bayesiano cuando $\alpha = 0.5$, es decir la implementación de la sección 4.2 para el método que ofreció mejores prestaciones.

Esta comparación es muy interesante para analizar si una de las curvas está siempre por encima de la otra, o si por el contrario depende de la zona de la ROC en la que se encuentren.

Como se mencionó anteriormente, según la curva avanza hacia la derecha, el valor del parámetro α disminuye, y por lo tanto se le da más importancia a no fallar sobre la clase minoritaria (etiqueta +1). Al contrario si se desplaza hacia la izquierda de la curva ROC, α va aumentando, y se le da mayor importancia a acertar sobre las muestras de la clase mayoritaria (etiqueta -1).

Con todo ello, de la figura 4.11 a la 4.18 se muestran las distintas curvas ROC. Notar que en línea discontínua se representan las curvas de entrenamiento, y en contínua las de test. Además aparecen en color fucsia las pertenecientes a $\alpha = 0.5$ y en negro la media para todos los valores de α.

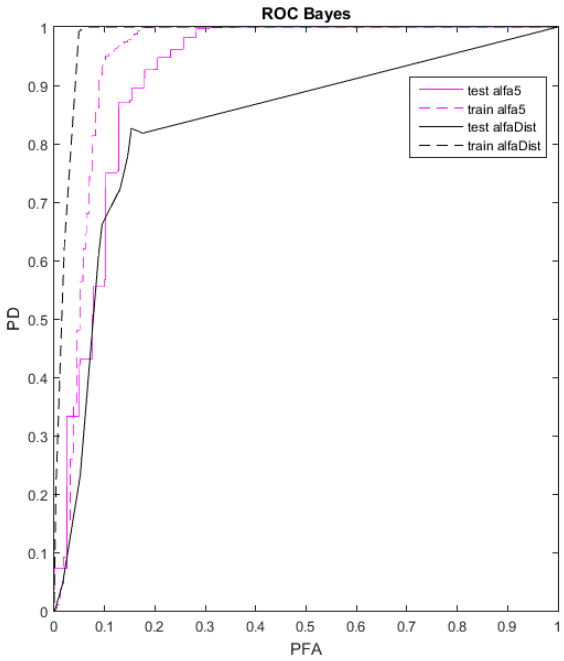

Figura 4.11: Curvas ROC resultantes de la base de datos 1 comparativas para $\alpha = 0.5$ y para distintos valores de α.

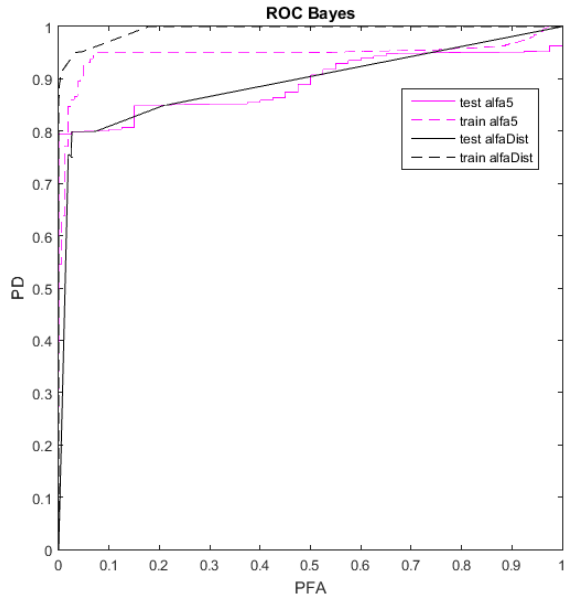

Figura 4.12: Curvas ROC resultantes de la base de datos 2 comparativas para $\alpha = 0.5$ y para distintos valores de α.

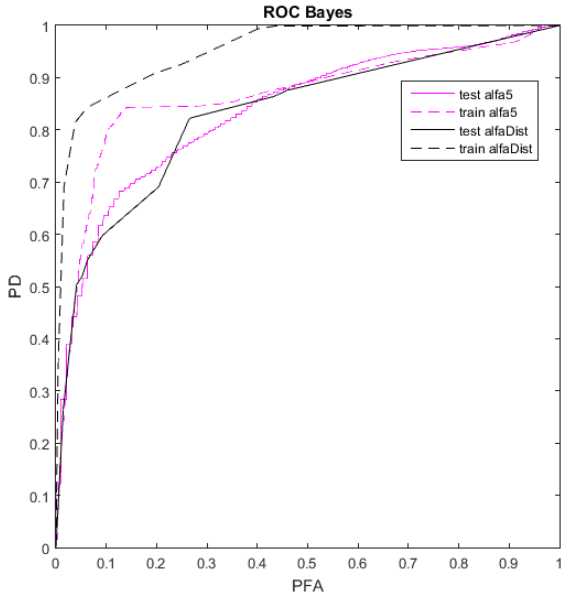

Figura 4.13: Curvas ROC resultantes de la base de datos 3 comparativas para $\alpha = 0.5$ y para distintos valores de α.

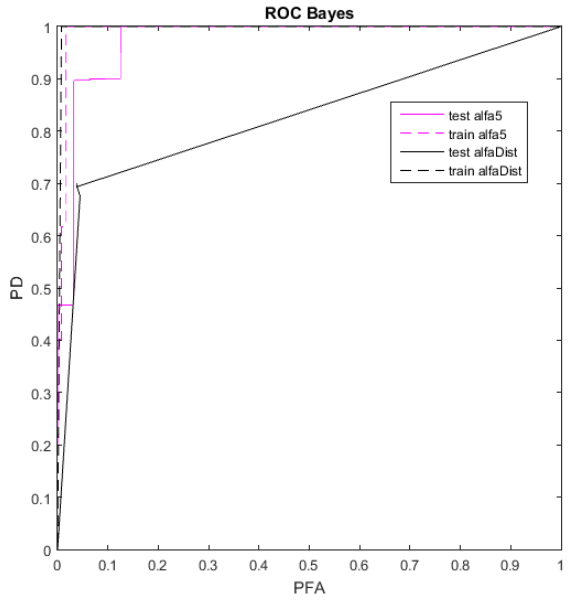

Figura 4.14: Curvas ROC resultantes de la base de datos 4 comparativas para $\alpha = 0.5$ y para distintos valores de α.

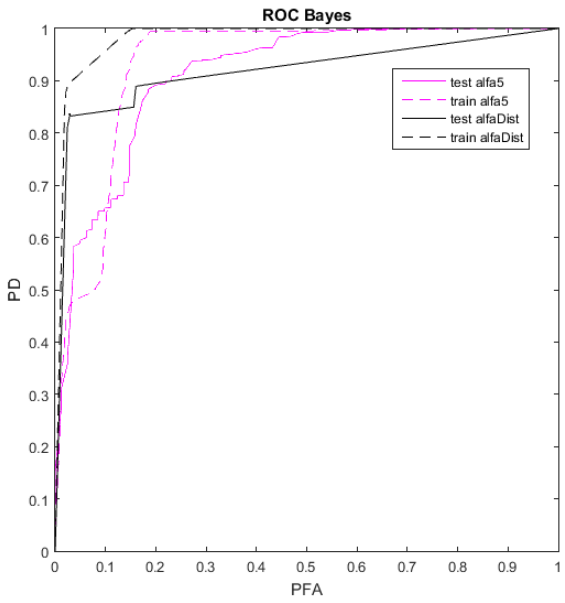

Figura 4.15: Curvas ROC resultantes de la base de datos 5 comparativas para $\alpha = 0.5$ y para distintos valores de α.

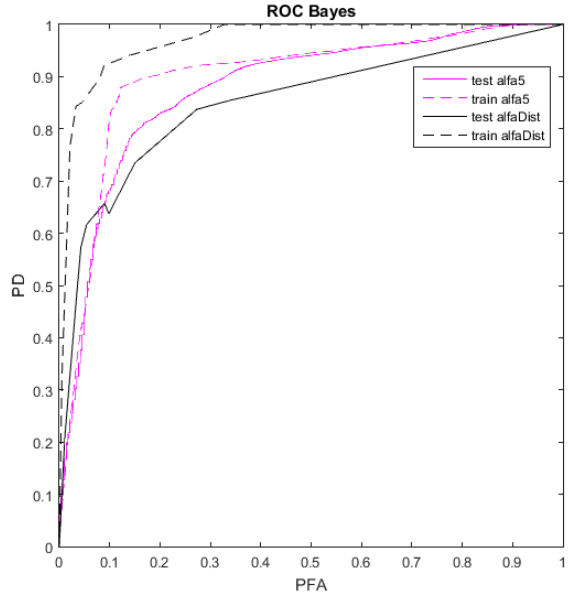

Figura 4.16: Curvas ROC resultantes de la base de datos 6 comparativas para $\alpha = 0.5$ y para distintos valores de α.

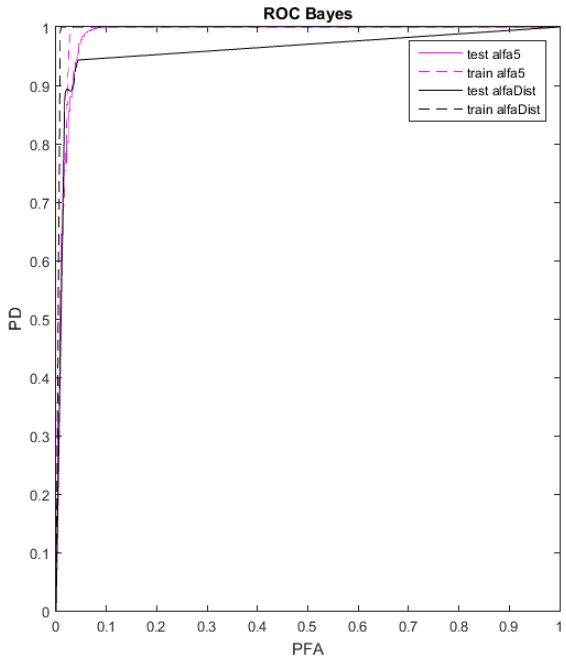

Figura 4.17: Curvas ROC resultantes de la base de datos 7 comparativas para $\alpha = 0.5$ y para distintos valores de α.

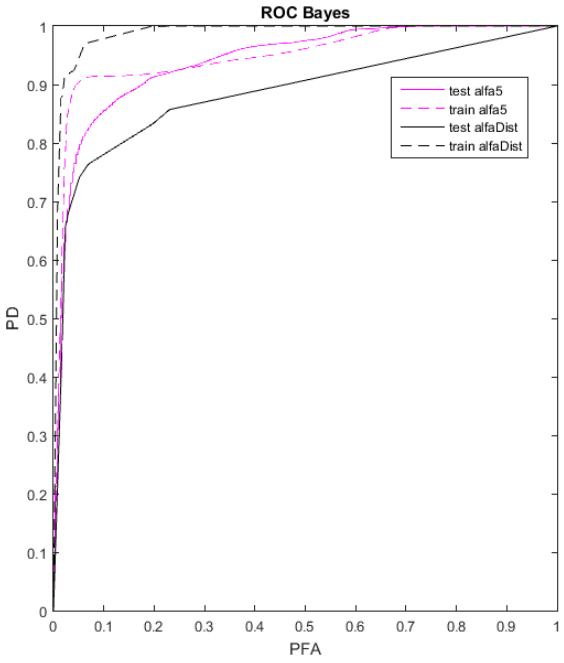

Figura 4.18: Curvas ROC resultantes de la base de datos 8 comparativas para $\alpha = 0.5$ y para distintos valores de α.

Si se observan las dos ROCs de los datos de entrenamiento, la ROC resultante de los distintos valores de α es superior (en todas las bases de datos) que cuando se considera una sola en $\alpha = 0.5$. Lo que indica que la red se ajusta mejor a los datos de entrenamiento cuando se tienen en cuentas los distintos valores de α que puede tomar el clasificador. Sin embargo al trasladarnos a las curvas de test se contempla que en todas las bases hay al menos un punto de intersección entre ambas curvas. Este punto como es lógico corresponde a $\alpha = 0.5$, ya que ambas curvas poseen ese valor de α. Es importante analizar entonces, lo que sucede a cada lado de la intersección.

Para los valores de α superiores a 0.5 (mayor importancia de acertar en la clase mayoritaria), las prestaciones de la curva con diferentes valores de α son superiores a las de la curva con $\alpha = 0.5$. Por el contrario, si lo que se pretende es ajustar mejor las etiquetas de la clase minoritaria (derecha de la intersección), la curva de $\alpha = 0.5$ supera las prestaciones de la curva que presenta diferentes valores de α. Aunque esto parezca contradictorio, ya que detecta peor la clase positiva cuanta mayor importancia se le da, esto viene ocasionado por el problema del sobreajuste. En entrenamiento el reducir α funciona mejor. Pero al ajustarse la red a un número de muestras de entrenamiento tan reducido, y luego intentar evaluar el clasificador sobre muestras desconocidas de la red (test), ésta no consigue generalizar correctamente porque los parámetros de la red se han sobreajustado sobre los datos de entrenamiento. Esto sucede en mayor o menor medida en todas las bases de datos.

Por lo que con los resultados obtenidos, es importante tener en cuenta lo que sucede en bases de datos de tamaño pequeño a la hora de escoger el valor correcto del parámetro α.

4.5. Comparación AUC frente a α

Tras los resultados de la sección 4.4, cabe preguntarse entonces, cómo varían las prestaciones en función del parámetro α. Por lo que en esta sección se va a llevar a cabo la evaluación del AUC en función de cada valor de α en cada base de datos. Esto se realizará en el método bayesiano, que es el que mejores prestaciones ha alcanzado hasta el momento.

También es importante darse cuenta de que cada base de datos tiene sus peculiaridades, no sólo en cuestión de tamaño, sino que algunas pueden tener mayor o menor variabilidad en los datos que otras. Motivos por los cuales, la variación de AUC con respecto a cada α puede ser muy cambiante según la base. A continuación se muestra una gráfica por cada base que representa en el eje 'x' el valor de α, y en el eje 'y' la AUC para cada clasificador. La AUC ha sido realizada como en implementaciones anteriores como el promedio de

las AUC en las 100 simulaciones, y en las 5 particiones.

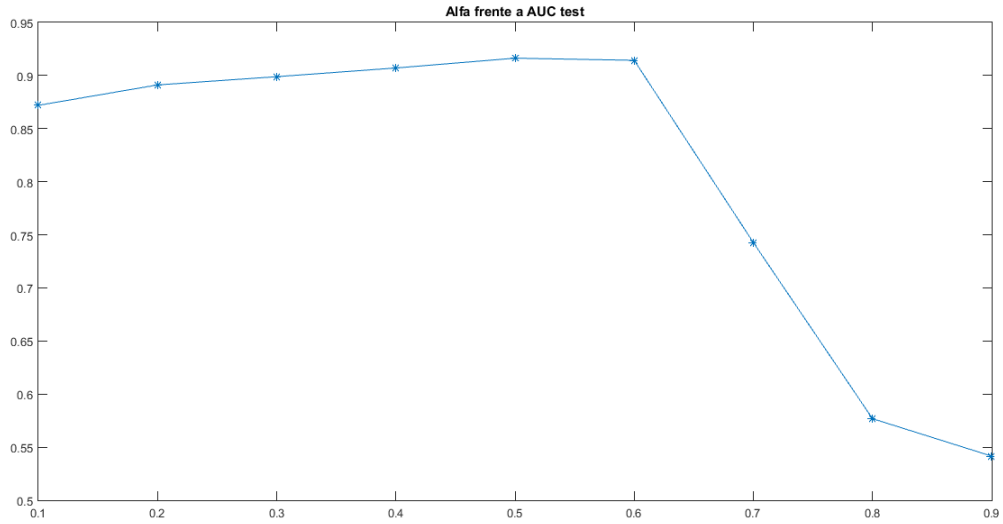

Figura 4.19: AUC en función de α para la base de datos 1.

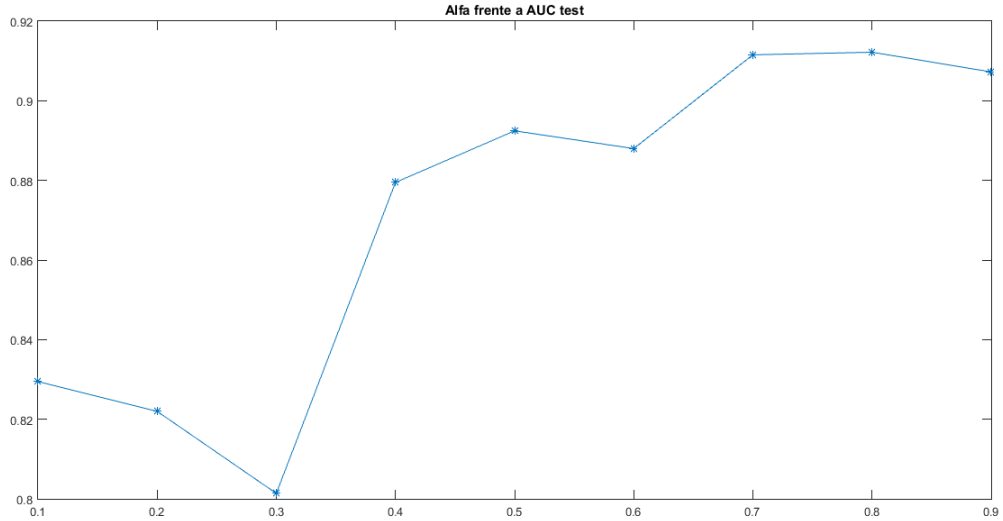

Figura 4.20: AUC en función de α para la base de datos 2.

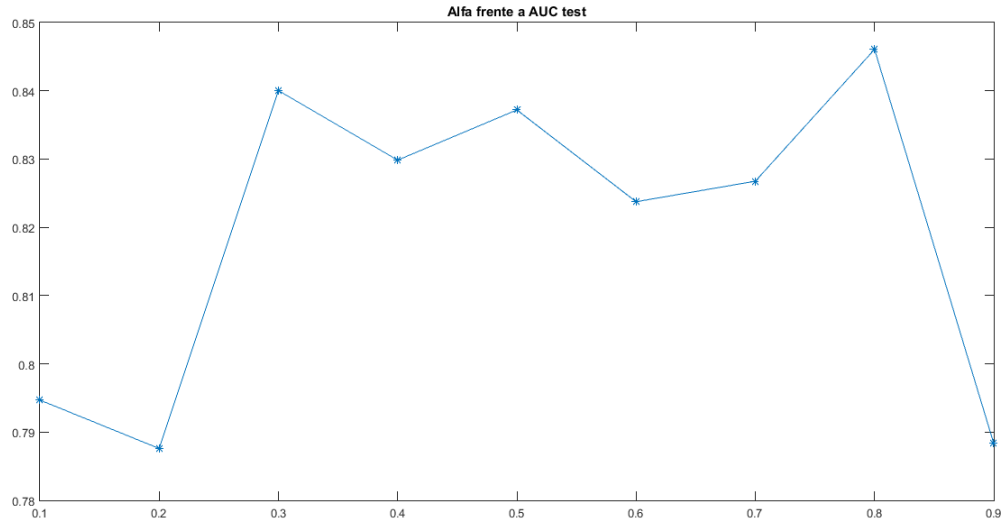

Figura 4.21: AUC en función de α para la base de datos 3.

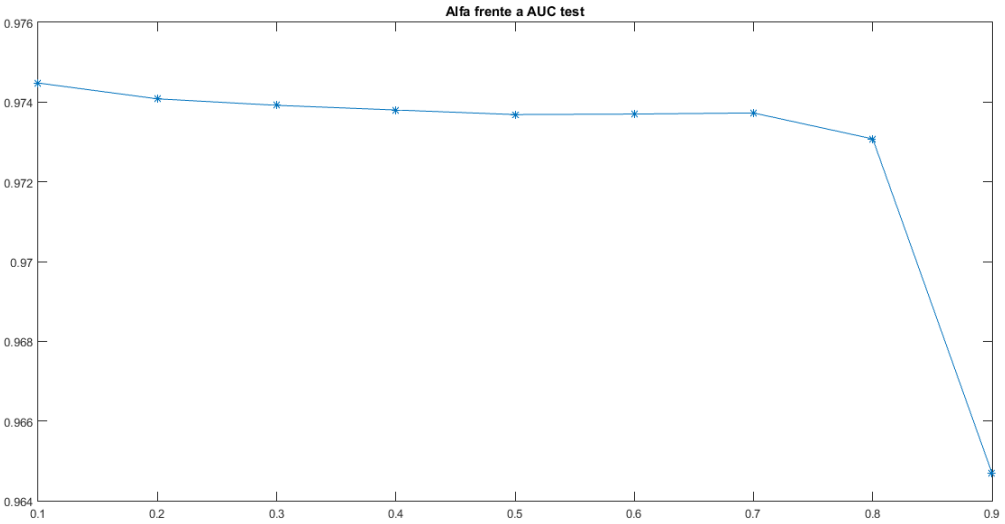

Figura 4.22: AUC en función de α para la base de datos 4.

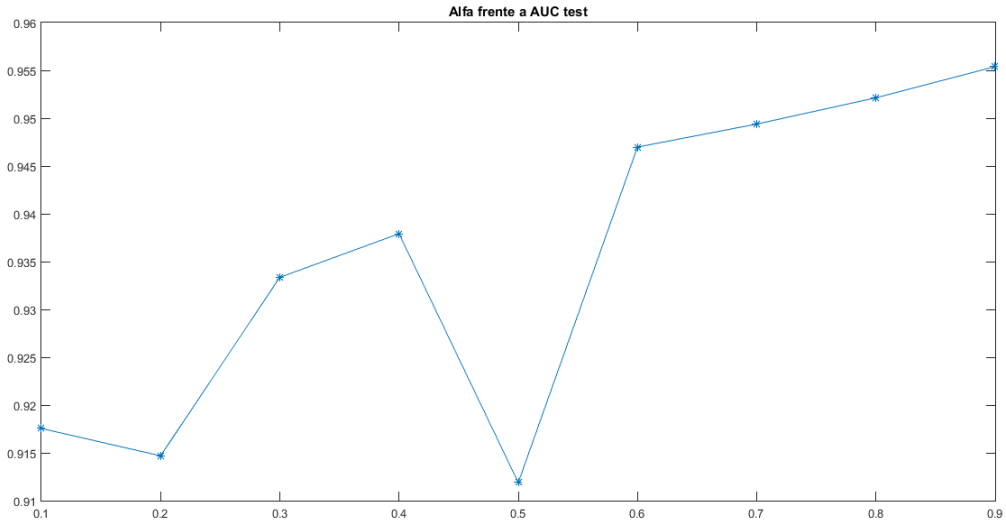

Figura 4.23: AUC en función de α para la base de datos 5.

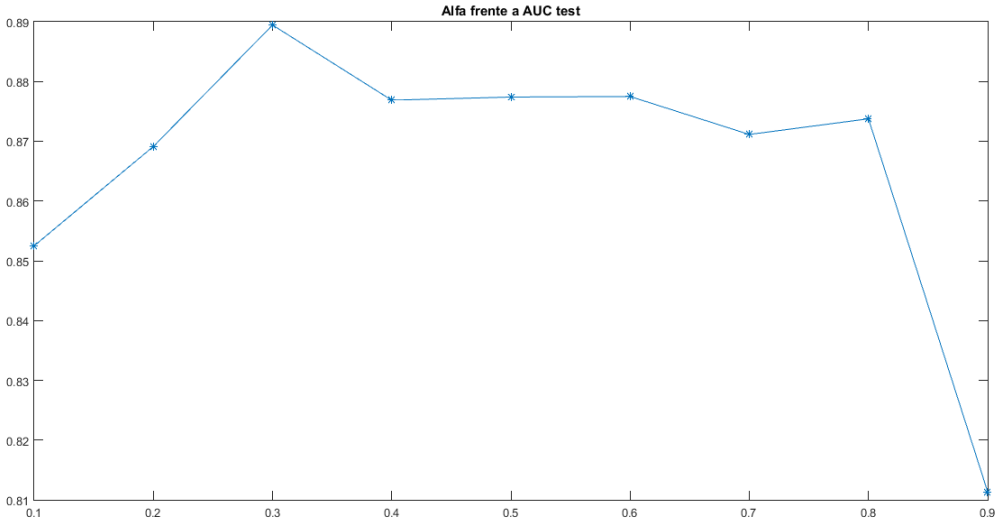

Figura 4.24: AUC en función de α para la base de datos 6.

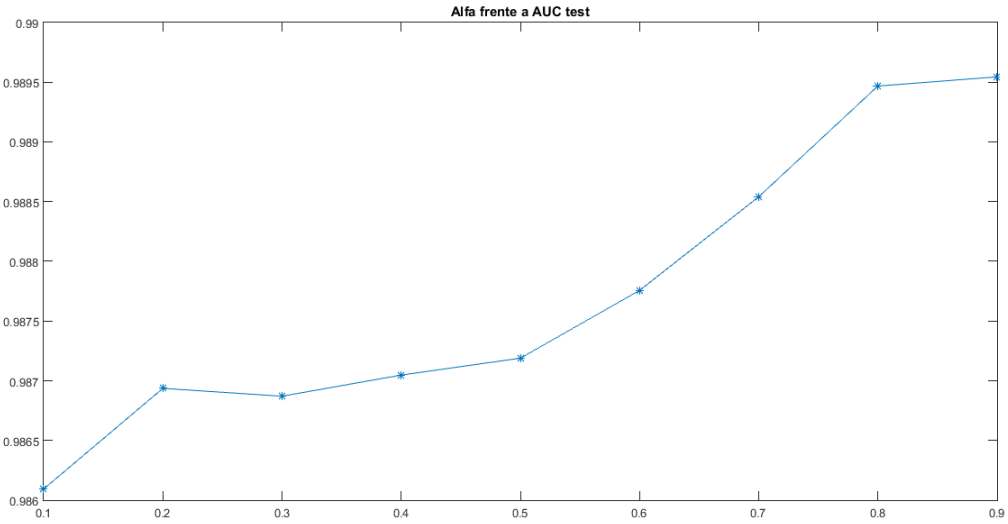

Figura 4.25: AUC en función de α para la base de datos 7.

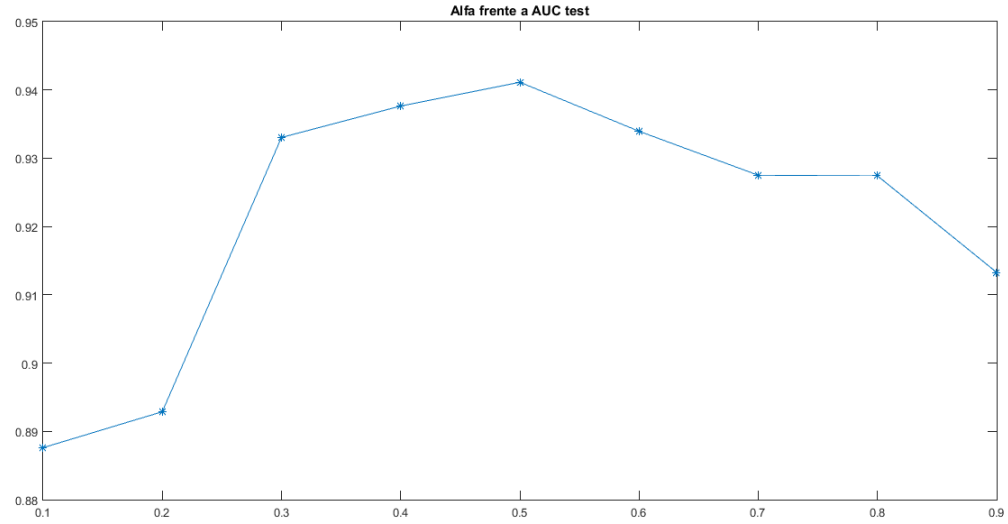

Figura 4.26: AUC en función de α para la base de datos 8.

De la figura 4.19 a la 4.26 se encuentran los resultados. Si se analizan las distintas gráficas, se puede percibir que no hay una relación clara entre comportamiento del AUC y tamaño de la base. Esto implica que el cómo varía AUC con α depende en gran medida de las caracterśticas intrínsecas de los datos. Observando los valores mínimos y máximos que toma la AUC en cada base, se concluye que la base 1 es la que mayor dependencia tiene con el parámetro α, y la base 7 la que menor dependencia.

4.6. Resultados de la combinación de clasificadores

Se han llevado a cabo 2 procedimientos para combinación de clasificadores como se expuso en el capítulo 2. Ambos parten de N_{cc} clasificadores individuales entrenados con la formulación bayesiana. Cada clasificador se construye con un valor de α diferente. En concreto $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$ con un total de $N_{cc} = 9$.

El primer clasificador conjunto evaluado tomará la decisión por mayoría. Es decir, para cada patrón de entrada $\mathbf{x}$, se observará la salida estimada de cada clasificador individual. La decisión del clasificador conjunto será la que hallan tomado al menos 5 de los 9 clasificadores. Éste se denominará clasificador conjunto por mayoría.

El segundo clasificador a evaluar sin embargo utiliza la regla de Bayes presentada en (Lazaro et al., 2015) y explicada en el capítulo 2. Ésta depende además del parámetro α^E como se muestra en la ecuación (2.67).

A continuación, en la tabla 4.3 se muestra la comparación de las prestaciones del AUC de ambos clasificadores conjuntos, junto con los resultados del clasificador individual de Bayes de la tabla 4.2.

Datos	Individual	Combinación por mayoría	Combinación con Bayes
1	0.8068	0.8268	0.9199
2	0.8715	0.8725	0.8775
3	0.8194	0.8284	0.8499
4	0.9728	0.9782	0.9863
5	0.9355	0.9379	0.9543
6	0.8666	0.8778	0.8991
7	0.9877	0.9885	0.9912
8	0.9216	0.9294	0.9311

Tabla 4.3: AUC comparativa del clasificador individual y las combinaciones de clasificadores individuales.

Como era de esperar, según aparece en la tabla 4.3 las prestaciones de la combinación de clasificadores mejoran con respecto al clasificador individual. Dentro de los dos tipos de combinación de clasificadores, el segundo ofrece mejores resultados. En algunas bases de datos la diferencia entre ambos es más destacable, como en la base 1. Esto lleva a preguntarse si siempre merece la pena utilizar clasificadores más complejos por un ligero o no tan ligero aumento de las prestaciones. Al final esto dependiendo del uso que se le vaya a dar compensará una mayor complejidad o no según las restricciones de la

aplicación.

Capítulo 5

Conclusiones y posibles líneas de continuación

En este capítulo se presentan las conclusiones extraídas de los resultados del capítulo 4 indicando qué métodos ofrecen mejores prestaciones y los aspectos más relevantes de la evaluación de las técnicas utilizadas. Por otro lado se propondrán algunas de las posibles implementaciones y consideraciones futuras de este Trabajo Fin de Grado.

5.1. Conclusiones

Las conclusiones extraídas a partir de los resultados son las siguientes:

- La elección de las bases de datos que van a servir para evaluar el problema debe ser adecuada para ofrecer resultados representativos de problemas reales.
- El tamaño de las bases de datos especialmente en problemas desbalanceados puede ocasionar cambios importantes en el sobreajuste de la red, concretamente en los datos de la clase minoritaria.
- Los parámetros de la red deben ser calculados y escogidos para que mejoren las prestaciones y eviten problemas de sobreajuste o mínimos locales. De esta forma la red puede adaptarse a las particularidades de cada aplicación.
- Cuando se le da la misma ponderación a cada clase, es decir el parámetro α toma el valor de 0.5, el método basado en la formulación bayesiana ofrece mejores resultados en cuando a AUC que el método WMSE. Esto ocurre en todas las bases de datos excepto la base 5, la cual posee el menor número de patrones dentro de las 8 bases del trabajo.

- A la hora de utilizar distintos valores de α, se observa en la curva ROC que el método bayesiano sufre mayor sobreajuste. Pero aun así el método bayesiano supera las prestaciones del WMSE en cuanto a AUC en todas las bases de datos.

- En bases de datos pequeñas como las utilizadas en el trabajo, cuando el valor de α es pequeño y se le da mayor importancia a acertar sobre la clase minoritaria en la función de coste bayesiano, la red mejora mucho en las prestaciones (ROC) sobre los datos de entrenamiento, pero generaliza peor sobre los datos de test. Es decir, que a valores bajos de α se sobreajusta sobre la clase minoritaria y se obtienen peores resultados que si se trabaja con $\alpha = 0.5$.

- El AUC en función de los distintos valores de α depende principalmente de la base de datos a utilizar. Algunas bases como la 1, tiene mayor varianza de AUC con respecto a α, y otras como la base 7 apenas varía su AUC cuando se cambia dicho valor.

- La combinación de clasificadores utilizando la formulación bayesiana supera las prestaciones de AUC del clasificador conjunto que toma la decisión por mayoría.

- La combinación de clasificadores en general obtiene mejores resultados que en el caso de trabajar únicamente con clasificadores individuales.

- En aplicaciones contretas, se elige el parámetro α en función del tipo de objetivo que requiera la actividad, y no sólo en función del coste global.

- El método a escoger en la práctica para un servicio determinado, dependerá tanto de aspectos económicos como de calidad. Para aplicaciones de bajo riesgo, puede suceder que un clasificador individual sea mejor opción ya que a pesar de ofrecer peores prestaciones es menos complejo (lo que por lo general se traduce en menor coste económico). Sin embargo para aplicaciones críticas, un pequeño aumento en las prestaciones puede tener grandes repercusiones sin importar el coste económico o complejidad del clasificador.

- El tiempo de ejecución de las 100 simulaciones de cada partición, en cada α, entrenando los distintos métodos es demasiado elevado. Por lo que para una aplicación comercial sería necesario optimizar el código para aumentar el rendimiento del clasificador.

5.2. Posibles líneas de continuación

Al tratarse de un Trabajo Fin de Grado, la duración del proyecto estaba limitada a 300 horas. Por lo que debido a esa restricción no ha sido posible realizar todas las pruebas o mejoras posibles. Aun así se van a exponer a continuación algunas de las posibles etapas futuras a realizar a partir de los resultados obtenidos en este proyecto:

- Probar en bases de datos reales con un número de patrones más elevado, de forma que se analice cómo influye esto a las prestaciones de la red neuronal. Probablemente disminuya el sobreajuste sobre la clase minoritaria, y pueda disminuirse el valor de α sin tener ese problema.
- Comprobar el efecto de la elección de la ventana de Parzen en el método bayesiano. Sería interesante obtener estos resultados, y evaluar el grado de influencia sobre estos.
- Evaluar el papel del ancho de la ventana de Parzen. Por ejemplo probar la posibilidad de que sea un parámetro adaptativo, que empiece con un tamaño grande para que todas las muestras influyan en el ajuste de los pesos, y posteriormente disminuirlo progresivamente para centrarse en las muestras que se encuentran cerca de la frontera de decisión.
- Utilización de herramientas de computación distribuida para la reducción del tiempo de ejecución del software, y aumentar así el rendimiento.

Apéndice A

Presupuesto y planificación

En este apéndice se va a exponer la planificación y seguimiento sobre el proyecto, con los tiempos de cada fase. Además se realizará el presupuesto final del trabajo en base a los recursos utilizados a lo largo del mismo.

A.1. Planificación del proyecto

La planificación es un aspecto muy importante dentro de un proyecto, ya que ayuda a la organización y optimización de los recursos a lo largo de un periodo determinado.

El orden de las tareas fue acordado al inicio del trabajo en base a los objetivos que se querían conseguir. A cada tarea a su vez se le iban marcando unos objetivos semanales, que se establecían y revisaban en las reuniones que se realizaban todos los viernes entre el tutor y la estudiante.

Hay múltiples herramientas que ayudan a la planificación y seguimiento de las tareas. En este apéndice se van a mostrar dos de ellas:

- Método PERT (*Program Evaluation and Review Technique*), herramienta útil para la planificación y control de trabajos, muestra las actividades previas necesarias a cada tarea. Permite saber el tiempo mínimo necesario para la ejecucción del proyecto. También indica las actividades que componen el camino crírico, es decir aquellas que si sufren un retraso afectan a la duración total del proyecto. Éste se muestra en la figura A.1, y con la información sobre el inicio y fin de cada actividad en la figura A.2. En esta última aparecen las etapas que conforman el camino crítico en color amarillo.

- Diagrama de Gantt, es una herramienta gráfica que tiene como objetivo representar el tiempo previsto para cada tarea o actividad a lo largo de

un tiempo total determinado. Es una de las herramientas más utilizadas para la planificación de trabajos. A continuación se muestra en la figura A.3.

Las figuras A.2 y A.3 han sido creadas con el programa ganttproject.

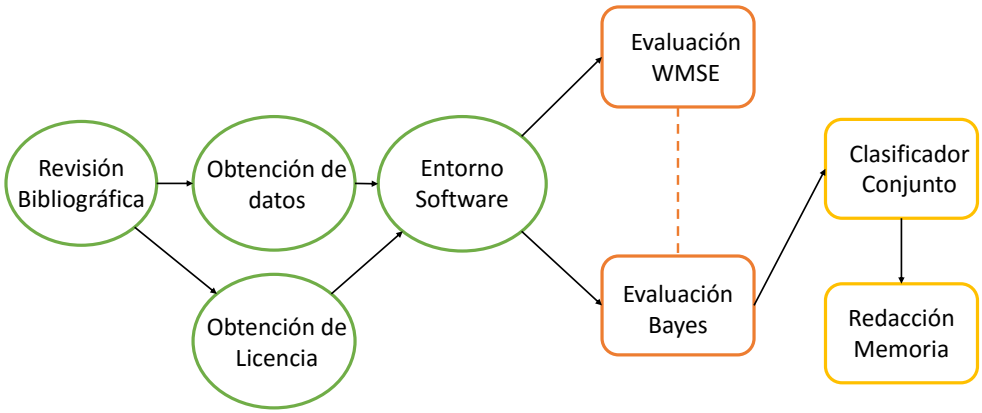

Figura A.1: PERT.

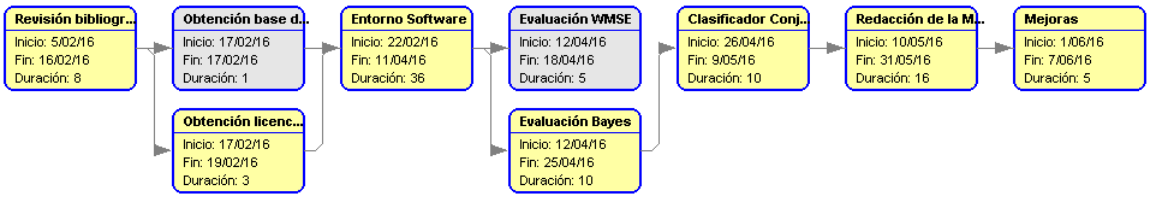

Figura A.2: PERT detallado con fecha de inicio y fin de cada tarea.

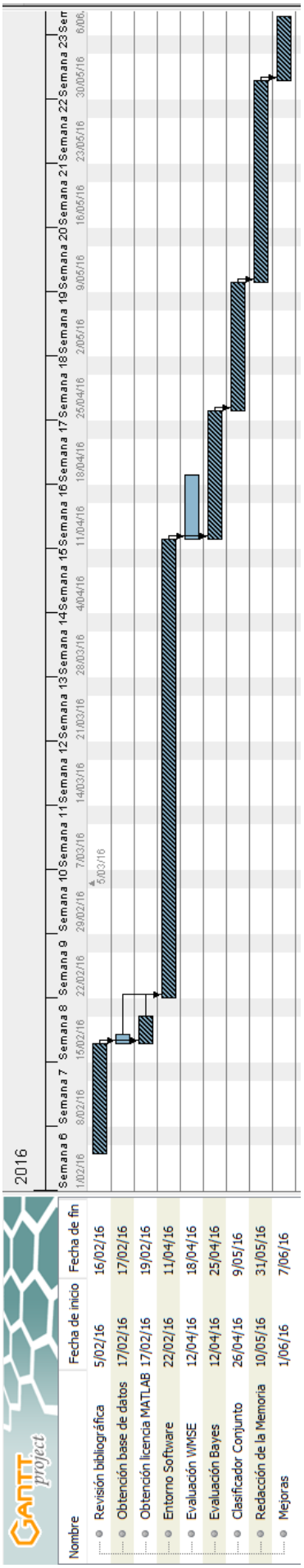

Figura A.3: Diagrama de Gantt.

A.2. Presupuesto del trabajo

En esta sección se explica cómo se ha realizado el presupuesto del trabajo, mostrando el coste individual de cada recurso y el coste final.

En primer lugar es importante diferenciar los tipos de costes que se pueden encontrar:

- Costes directos: aquellos que se pueden imputar de forma directa a la realización del trabajo.
- Costes indirectos: aquellos que no se pueden imputar directamente al trabajo, pero que sí son necesarios para el desarrollo del mismo.

Dentro de los costes directos, se realiza una nueva separación entre dos apartados. Uno de ellos destinado a los recursos materiales, y otro para la mano de obra del personal implicado en el trabajo.

En cuanto a la duración temporal de dicho trabajo, se ha realizado en un total de 4 meses, comenzando el 5 de febrero y finalizando el 7 de junio. El personal ha estado formado por 2 personas, el profesor y la estudiante. El tutor, ha dedicado 1 hora a la semana (reunión semanal), lo que en 4 meses implica 16 horas. La alumna ha trabajado 300 horas en total como está determinado en las normas del Trabajo Fin de Grado, aproximadamente 4 horas al día [1].

Teniendo en cuenta todo lo anterior, a continuación se procede a la especificación de cada coste.

Los recursos materiales contienen el ordenador portátil, el local donde se desarrolló la actividad, y la licencia del software matemático. El ordenador utilizado ha sido el ASUS K52J con un coste total de 613.61 €, con una vida útil aproximada de 5 años, la amortización del equipo equivale a 10.23 €/mes, lo que se traduce en 40.91 € en 4 meses. El alquiler del local tiene un coste mensual de 250 €, en los 4 meses un total de 1000 €. Por último el precio de la licencia del software de MATLAB, 6000 € con un tiempo de vida de 8 años, cuya amortización es de 62.5 €/mes, por 4 meses un total de 250 € en la duración del trabajo.

Ahora por otro lado, se considerará el coste de la mano de obra. Según el boletín oficial del estado (BOE) en su artículo (BOE, 2015) el salario mínimo interprofesional se fija en 21.84 €/día. Este salario se refiere a una jornada legal de trabajo (8 horas). Como la jornada laboral no llega a las 8 horas,

[1] Hay que tener en cuenta que la dedicación semanal ha variado en función de la carga de trabajo semanal asociada a las otras asignaturas que se han cursado al mismo tiempo que el Trabajo Fin de Grado.

se tendría que realizar la parte proporcional, pero como se indica en (BOE, 2015), en el caso de no superar los 120 días el salario debe ser la suma del salario antes indicado junto con la parte proporcional de las gratificaciones extraordinarias, resultando un mínimo de 31.03 €/día. Se considera que la estudiante recibe el salario mínimo por su baja experiencia laboral, por lo que con 31.03 €/día y 4 horas/día, sale 7.76 €/hora. Con un total de 300 horas, el salario estipulado para los 4 meses es de 2328 €. El tutor sin embargo, debido a su experiencia y su alto grado de estudio se obtiene un aumento de salario con respecto a la estudiante, quedando en 20 €/hora, por 16 horas de trabajo, el sueldo de la actividad completa termina en 320 €.

Los costes indirectos se estiman considerando un 5 % de los costes directos.

Finalmente con unos costes directos de 3938.91 € e indirectos de 196.95 €, queda un presupuesto total de 4135.86 €. En la tabla A.1 se encuentra un resumen del presupuesto.

Concepto	**Coste/Unidad**	**Número de unidades**	**Coste total**
Costes directos			3938.91 €
- Recursos materiales			1290.91 €
Ordenador portátil	10.23 €/mes	4 meses	40.91 €
Local + Luz	250 €/mes	4 meses	1000 €
Licencia software MATLAB	62.5 €/mes	4 meses	250 €
- Mano de obra			2648 €
Profesor	20 €/hora	16 horas	320 €
Estudiante	7.76 €/hora	300 horas	2328 €
Costes indirectos			196.95 €
Presupuesto total			4135.86 €

Tabla A.1: Presupuesto.

Apéndice B

Glosario de acrónimos

ANN: Red neuronal artificial (*Artificial Neural Network*)

AUC: Area debajo de la curva (*Area Under Curve*)

BOE: Boletín Oficial del Estado

GRBF: Funciones de base radial generalizadas (*Generalised Radial Basis Function network*)

IA: Inteligencia Artificial

IEEE: Instituto de ingeniería eléctrica y electrónica (*Institute of Electrical and Electronics Engineers*)

MATLAB: Laboratorio de matrices (*MATrix LABoratory*)

ML: Máxima verosimilitud (*Maximum Likelihood*)

MLP: Perceptrón multicapa (*Multilayer Perceptron*)

MMSE: Mínimo error cuadrático medio (*Minimum Mean Square Error*)

MSE: Error cuadrático medio (*Mean Square Error*)

NN: Red neuronal (*Neural Network*)

PCA: Análisis de componentes principales (*Principal Component Analysis*)

PERT: (*Program Evaluation and Review Technique*)

RBF: Funciones de base radial, (*Radial Basis Function network*)

ROC: Curva característica de operación (*Receiver Operating Characteristic*)

SVM: Máquinas de vectores soporte (*Support Vector Machines*)

WMSE: Error cuadrático medio ponderado (*Weighted Mean Square Error*)

Bibliografía

A. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, y F. Herrera. "KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework". *Journal of Multiple Valued Logic & Soft Computing*, 17:255–287, 2011.

I. G. Alonso, M. R. Fernández, J. J. Peralta, A. C. García, y J. M. O. Quintana. "Técnicas de aprendizaje automático al servicio de la eficiencia energética en el hogar digital". Disponible en http://aulagreencities.coamalaga.es/tecnicas-de-aprendizaje/. 2013 (Última visita: junio 2016).

E. Alpaydin. *Introductions to Machine Learning*. The M.I.T. Press, Cambrigde (Massachusetts), 2004.

A. F. Atiya. "Bankruptcy prediction for credit risk using neural networks: A survey and new results". *IEEE Transactions on Neural Networks*, 12(4):929–935, julio 2001.

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2010.

BBVA Open4U. "Aprendizaje automático: cómo un algoritmo ayuda a un médico o a un banco.". Disponible en https://bbvaopen4u.com/es/actualidad/aprendizaje-automatico-como-un-algoritmo-ayuda-un-medico-o-un-banco. 2015 (Última visita: junio 2016).

P. G. Bejerano. "Apoyos públicos al desarrollo de la inteligencia artificial.". Disponible en http://blogthinkbig.com/apoyos-publicos-al-desarrollo-la-inteligencia-artificial/. 2015 (Última visita: junio 2016).

C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1997.

BOE. "Ministerio de empleo y seguridad social.". *Boletín Oficial del Estado*, (312):123262–123264, diciembre 2015.

A. P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". *Pattern Recognition*, 30(7):1145–1159, julio 1997.

C. C. Chang y C. J. Lin. "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology*, 2(3):A27:1–A27:27, 2011.

C. C. Chang y C. J. Lin. "LIBSVM: A library for support vector machines". Software disponible en http://www.csie.ntu.edu.tw/~cjlin/libsvm/. Versión 3.21, 2015 (Última visita: junio 2016).

R. Christensen. "Testing fisher, neyman, pearson, and bayes". *The American Statistician*, 59(2):121–126, mayo 2005.

G. Cybenko. "Approximation by superposition of a sigmoidal function". *Math. Control, Signals and Systems*, 2:303–314, 1989.

J. R. Dorronsoro, F. Ginel, C. Sánchez, y C. Santa-Cruz. "Neural fraud detection in credit card operations". *IEEE Transactions on Neural Networks*, 8(4):827–834, julio 1997.

R. O. Duda, P. E. Hart, y D. G. Stork. *Pattern classification*. John Wiley & Sons, segunda edición, 2001.

T. Fawcett. "An introduction to ROC analysis". *Pattern Recognition Letters* 27(8):861–874, junio 2006.

K. R. Foster, R. Koprowsky, y J. D. Skufka. "Machine learning, medical diagnosis, and biomedical engineering research - comentary". *BioMedical Engineering OnLine*, 13(94):1–9, 2014.

K. Funahashi. "On the approximate realization of continuous mappings by neural networks". *Neural Networks*, 2:183–192, 1989.

K. Funahashi. "Multilayer neural networks and Bayes decision theory". *Neural Networks*, 11(2):209–213, marzo 1998.

M. Galar, A. Fernández, E. Barrenechea, H. Bustince, y F. Herrera. "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches". *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 42(4):463–484, julio 2012.

M. Galar, A. Fernández, E. Barrenechea, y F. Herrera. "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling". *Patter Recognition*, 46:3460–3471, 2013.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, segunda edición, 1998.

H. He y E. A. Garcia. "Learning from imbalanced data". *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, septiembre 2009.

R. D. Hof. "La nueva generación de productos Google tendrá una inteligencia más humana". Disponible en https://www.technologyreview.es/robotica/48584/la-nueva-generacion-de-productos-google-tendra/. 2015 (Última visita: junio 2016).

K. Hornik. "Approximation capabilities of multilayer feedforward networks". *Neural Networks*, 4(2):251–257, 1991.

K. Hornik, M. Stinchcombe, y H. White. "Multilayer feedforward networks are universal approximators". *Neural Networks*, 2:183–192, 1989.

K. Hornik, M. Stinchcombe, y H. White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". *Neural Networks*, 3:551–560, 1990.

P. Isasi y I. M. Galván. *Redes Neuronales. Un enfoque práctico*. Pearson Educación – Prentice Hall, Madrid, 2004.

S. M. Kay. *Fundamentals of statistical signal processing: detection theory*. Prentice Hall, Upper Saddle River (New Jersey), 1998.

I. Kononenko. "Machine learning for medical diagnosis: hystory, state of the art and perspective". *Artificial Intelligence in Medicine*, (23):89–109, 2001.

R. Koprowsky, W. Zieleznik, Z. Wrobel, J. Malyzek, y W. Wojcik. "Assessment of significance of features adquired from thyroid ultrasograms in Hashimoto's disease". *BioMedical Engineering OnLine*, 11(48):1–20, 2012.

L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, Hoboken (New Jersey), 2004.

M. Lazaro, F. Herrera, y A. R. Figueiras-Vidal. "Classification of binary imbalanced data using a bayesian ensemble of bayesian neural

networks". En *Proceedings of the 16 International Conference on Applicaitons of Neural Networks* (EANN 2015). Rodas, Grecia, septiembre 2015.

A. Ling. "Toyoya y su millonaria inversión en inteligencia artificial.". Disponible en https://www.unocero.com/2015/11/07/toyota-y-su-millonaria-inversion-en-inteligencia-artificial/. 2015 (Última visita: junio 2016).

M. Luengo-Oroz. "El videojuego que ayuda a salvar vidas". Entrevista disponible en http://www.eleconomistaamerica.com/ciencia-eAm-mx/noticias/6611572/04/15/El-videojuego-que-ayuda-a-salvar-vidas.html. 2015 (Última visita: junio 2016).

M. Moavenian y H. Khorrami. "A qualitative comparison of artificial neural networks and support vector machines in ECG arrythmias classification". *Expert Systems and Applications*, (37):3088–3093, 2010.

J. Moody y C. J. Darken. "Fast learning in networks of locally-tuned processing units". *Neural Computation*. 1(2):281–2944, 1989.

J. Nocedal y S. J. Wright. *Numerical Optimization*. Springer Verlag, New York, segunda edición, 2006.

F. Palazuelos. "Esto es lo que hace Google cuando le preguntas algo que nunca ha respondido". Disponible en http://hipertextual.com/2015/10/google-preguntas-inteligencia-artificial. 2015 (Última visita: junio 2016).

E. Parzen. "On the estimation of a probability density function and the mode". *Annals of Mathematical Statistics*, (3):1065–76, septiembre 1962.

H. V. Poor. *An Introduction to Signal Detection and Estimation*. Springer Texts in Electrical Engineering (Springer Verlag), New York, segunda edición, 1994.

M. Rodríguez. "Cómo están utilizando la inteligencia artificial las grandes empresas". Disponible en https://www.euroresidentes.com/tecnologia/avances-tecnologicos/como-estan-utilizando-la-inteligencia-artificial-las-grandes-empresas. 2016 (Última visita: junio 2016).

L. Rokach. *Pattern Classification Using Ensemble Methods*. World Scientific (Series in Machine Perception and Artifical Intelligence), River Edge (New Jersey), 2010.

D. E. Rumelhart, G. E. Hinton, y R. J. Williams. "Learning internal representations by error propagation". En D. E. Rumelhart et al. (eds.), *Parallel Distributed Processing*, volumen 1, *Foundations*, págs. 318–362. MIT Press, Cambridge (Massachusetts), 1986.

D. E. Rumelhart, G. E. Hinton, y R. J. Willians. "Learning representations by back-propagating errors". *Nature (London)*, 323:533–536, 1986.

M. P. Salas. "Jugando online puedes ayudar a diagnosticar la malaria (y salvar vidas)". Disponible en http://www.eldefinido.cl/actualidad/mundo/792/Jugando_online_puedes_ayudar_a_diagnosticar_la_malaria_y_salvar_vidas/. 2013 (Última visita: junio 2016).

L. S. Scharf. *Statistical signal processing: detection, estimation, and time series analysis*. Addison-Wesley. 1991.

B. Schölkopf, C. Burges, y A. Smola. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge (Massachusetts), 1999.

M. Spector y M. Ramsey *The Wall Street Journal*, edición del 14 de enero de 2016. Disponible en http://www.wsj.com/articles/obama-administration-proposes-spending-4-billion-on-driverless-car-guidelines-1452798787 (Última visita: junio 2016).

H. L. Van Trees. *Detection, Estimation, and Modulation Theory: Part I*. John Wiley and Sons, New York, 1968.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag. 1995.

V. N. Vapnik, S. Golowich, y A. Smola. "Support vector method for function approximation, regression estimation, and signal processing". En M. Mozer, M. Jordan, y T. Petsche, editores, *Neural Information Processing Systems*, págs. 169–184, The M.I.T. Press, Cambridge (Massachusetts), 1997.

P. J. Werbos. "Backpropagation through time: what it does and how to do it". *Proceedings of the IEEE*, 78(10):1550–1560, octubre 1990.

B. Widrow y M. A. Lehr. "30 years of adaptive neural networks: perceptron, madaline and backpropagation". *Proceedings of the IEEE*, 78(9):1415–1441, septiembre 1990.

B. Widrow, D. E. Rumelhard, y M. A. Lehr. "Neural networks: Applications in industry, business and science". *Communications of the ACM*, 37(3):93–105, marzo 1994.

A. Williams. "Blackrock y Google negocian una posible 'joint venture"'. Disponible en http://www.expansion.com/empresas/2015/11/09/5641228322601d960f8b4570.html. 2015 (Última visita: junio 2016).

B. Xie y H. Minn. "Real-time sleep apnea detection by classifier combination". *IEEE Transactions on Information Technology in Biomedicine*, 16(3):469–477, mayo 2012.

G. P. Zhang. "Neural networks for classification: A survey". *IEEE Transactions on Systems, Man, and Cybernetics*, 30(4):451–462, noviembre 2000.

Z. Zhou y X. Liu. "Training cost-sensitive neural networks with methods addressing the class imbalance problem". *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, enero 2006.