ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Máquinas de Aprendizaje Extremo Multicapa: Estudio y Evaluación de Resultados en la Segmentación Automática de Carótidas en Imágenes Ecográficas

AUTOR: Adrián Sánchez Morales

DIRECTOR: José Luis Sancho Gómez

A mi familia

"Cada día sabemos más y entendemos menos",
Albert Einstein

AGRADECIMIENTOS

Con estas líneas me gustaría agradecer, en primer lugar, a José Luis Sancho Gómez por su paciencia y dedicación. Gracias por brindarme la oportunidad de dar un paso más, camino de conseguir que aquello que me gusta se convierta algún día en mi profesión.

Quiero agradecer, especialmente, a Rosa María Menchón Lara por su entrega y amabilidad, y por dedicar su valioso tiempo en el laboratorio a solucionar todas las dudas que hayan podido surgir, por absurdas que parecieran.

Gracias a José García-bravo García, por haberse preocupado por mí durante tantos años. Con él pisé por primera vez esta universidad y fue quién me ayudó a decidirme por esta carrera.

Agradecer a mi familia su apoyo incondicional, en especial a mis padres. Gracias por enseñarme a afrontar la vida con una sonrisa y por no fallarme nunca.

Quiero hacer especial mención a mi novia Cristina y agradecerle que haya estado ahí, día tras día, durante todos estos años de carrera, soportándome y dándome fuerzas en los buenos, pero sobre todo en los malos momentos.

En general, gracias a todas aquellas personas que han confiado en mí.

RESÚMEN

El aprendizaje automático es una rama de la inteligencia artificial, y se puede definir como el conjunto de técnicas, métodos y sus implementaciones algorítmicas capaces de aprender y mejorar su eficacia a través de la experiencia. En la mayoría de los casos se busca realizar ese aprendizaje a partir de información no estructurada y sin supervisión humana. En las últimas décadas, el uso de técnicas de aprendizaje automático en campos tan diversos como la informática, la estadística, la robótica, la medicina, etc. se ha visto incrementado de manera extraordinaria. En estas aplicaciones la necesidad de estructuras complejas es cada vez más frecuente debido a la cantidad de datos que deben tratarse. De ahí el desarrollo de lo que se llama Aprendizaje Profundo (*Deep Learning*).

Ha sido muy estudiado el hecho de que los Auto-Codificadores (*Auto Encoders, AE*) juegan un papel fundamental en el aprendizaje no supervisado y en las arquitecturas de redes de aprendizaje profundo. Este trabajo introduce los Auto-Codificadores basados en las Máquinas de Aprendizaje Extremo (*Extreme Learning Machines, ELM*), que son capaces de aprender las características de representación de los datos de entrada usando sus valores singulares, y que son usados como base para la construcción de Máquinas de Aprendizaje Extremo Multicapa (*Multi Layer Extreme Learning Machines, ML-ELM*).

A lo largo del trabajo se podrá comprobar cómo el funcionamiento de los auto-codificadores mejora el de un método tradicionalmente usado como es el de Descomposición en Valores Singulares (SVD). A su vez, se verá cómo las redes ML-ELM obtienen mejores resultados en problemas de clasificación que las simples ELMs, trabajando con la base de datos MNIST. Finalmente, se usará todo este conocimiento para el procesamiento de imágenes médicas con el objetivo de medir el grosor íntima-media carotídeo y comprobar que todos los resultados obtenidos anteriormente son consistentes en aplicaciones más complejas.

ABSTRACT

Machine learning is a subfield of artificial intelligence and it can be defined as the set of techniques, methods and algorithms which are capable of learn and improve its efficiency throughout experience. The aim is to learn from data without structure and, in most cases, without human supervision. During the last decades, the use of machine learning techniques in fields as wide as informatics, statistics, robotics, medicine, etc. have been increasing incredibly. Due to the huge amount of data that must be treated, there has been an increase in the need of architectures more complex for these applications. Hence, nowadays it is being developed what is called Deep Learning.

It is known Auto Encoders play a fundamental role in unsupervised learning and in deep architectures. This Final Project introduces Extreme Learning Machine based Auto Encoder (ELM-AE), which learns feature representations using singular values and is used as the basic building block for Multi-Layer Extreme Learning Machine (ML-ELM).

Throughout this work will be proved how the Auto Encoders performance improves the classical method called Singular Value Decomposition (SVD). Then, will be shown ML-ELMs get better results than a simple structure such as ELM in classification problems, for MNIST dataset. Finally, all this knowledge will be used for the medical image processing in order to measure the carotid intima-media thickness (IMT) and check all the results obtained above are consistent in more complex applications.

ÍNDICE GENERAL

CAPÍTULO 1.

INTRODUCCIÓN

Permitir a los ordenadores modelar el mundo que nos rodea lo suficientemente bien como para que se hable de inteligencia, ha sido el foco de más de medio siglo de investigaciones. Para conseguirlo, sería necesario almacenar una gran cantidad de información en estos ordenadores. Parece impensable que la acción de gestionar toda esta información se realice manualmente por lo que, para que los ordenadores puedan usarla para contestar algunas preguntas y generalizar sobre nuevos contextos, muchos investigadores han recurrido a los algoritmos de aprendizaje para capturar una enorme fracción de toda esta información [1].

Es sabido que para solucionar un problema en un ordenador se necesita un algoritmo, que no es más que una secuencia de instrucciones llevadas a cabo con el objetivo de transformar una entrada de datos y obtener una salida. Para una misma tarea existirán varios algoritmos y, generalmente, interesará encontrar el más eficiente [2].

Figura 1.1: Dispositivos electrónicos.

Gracias a los avances tecnológicos, actualmente se tiene la posibilidad de almacenar y procesar grandes cantidades de datos, así como acceder a ellos desde diferentes localizaciones. La mayoría de los dispositivos de adquisición de datos son digitales y guardan datos fiables.

Como todo el mundo sabe, no se pueden conocer acciones que aún no han pasado. Si esto fuera así, no sería necesario analizar datos, pero como no lo es, sólo se pueden almacenar y esperar a que siendo analizados, se puedan transformar en información útil y usar, por ejemplo, para realizar predicciones.

Una de las claves del aprendizaje máquina es que existe un proceso que explica los datos que se observan y que, aunque no se conocen los detalles del proceso en sí, se sabe que no es del todo aleatorio.

Tanto este proyecto como aquellos que estén igualmente relacionados con el aprendizaje máquina usarán esta idea: no es posible identificar el proceso completamente, pero sí se cree en la posibilidad de construir una buena y útil aproximación. Esta aproximación puede no explicarlo todo, pero puede servir para parte de los datos, por lo que, aunque puede ser imposible encontrar el motivo subyacente del proceso, se pueden detectar patrones o regularidades asociadas. Aquí es donde nace el aprendizaje máquina. Estos patrones de los que se habla pueden ayudar a entender los procesos, o se pueden usar para realizar predicciones asumiendo que el futuro, al menos el futuro cercano, no será muy diferente del pasado, cuando los datos fueron recogidos. De esta forma, las predicciones pueden ser correctas.

Por tanto, el aprendizaje maquina consiste en la programación de algoritmos para reproducir un criterio de funcionamiento usando datos de ejemplo o experiencias pasadas. Es decir, se tiene un modelo definido sobre algunos parámetros, y el aprendizaje es la ejecución de un programa para optimizar los parámetros del modelo usando datos de entrenamiento. El modelo puede ser *predictivo* (predicciones sobre el futuro), *descriptivo* (adquirir conocimiento de los datos), o ambos.

En este campo es muy importante la teoría de la estadística para la construcción de modelos matemáticos, ya que la tarea principal es tomar decisiones a partir de un ejemplo. El papel de la ciencia es doble: primero, en el entrenamiento, se necesitan algoritmos eficientes para solucionar problemas de optimización, así como almacenar y procesar grandes cantidades de datos. En segundo lugar, una vez que el modelo ha aprendido, se procede a su solución y representación, que deben ser del mismo modo eficientes. En ciertas aplicaciones, la eficiencia del aprendizaje puede ser tan importante como la precisión.

Este proyecto se centrará en el trabajo con redes de neuronas artificiales (denominadas habitualmente como RNA o en inglés: *Artificial Neural Networks, ANN*), un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso. Una red neuronal artificial se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como redes de neuronas o redes neuronales.

Las investigaciones en este campo han ido aumentando de forma notable en estos últimos años. Desde 1943 cuando apareció el primer modelo de neurona artificial, han aparecido propuestas nuevas y más sofisticadas década tras década. El análisis matemático ha solucionado algunos de los misterios propuestos por estos nuevos modelos, pero se puede asegurar que aún quedan muchas preguntas abiertas a futuras investigaciones [3].

Una de las características más impresionantes de las redes neuronales artificiales es su capacidad para aprender. Para ello, se inspiran en el sistema nervioso biológico, en particular, el cerebro humano. Se debe tener en cuenta que la comprensión de cómo el cerebro realiza la

acción de aprender es aún muy primitiva, aunque se tiene un conocimiento básico del proceso. Se cree que durante este proceso de aprendizaje la estructura neuronal del cerebro se altera, aumentando o disminuyendo la fuerza de sus conexiones sinápticas en función de su actividad. Por esto es por lo que la información más relevante es más fácil de recordar que la información con la que no se ha trabajado durante mucho tiempo; la primera tendrá conexiones sinápticas más fuertes y la información menos relevante tenderá a debilitar sus conexiones gradualmente, lo que la hace más difícil de recordar.

Aunque de forma simplificada, las redes neuronales artificiales pueden modelar este proceso de aprendizaje mediante el ajuste de las conexiones entre las neuronas en la red. Esto emula con eficacia el fortalecimiento y debilitamiento de las conexiones sinápticas que se encuentran en nuestros cerebros, lo que permite a la red aprender.

La clasificación es una de las tareas más frecuentes a la hora de tomar decisiones por parte de la actividad humana. Un problema de clasificación ocurre cuando un objeto necesita ser asignado dentro de un grupo predefinido o clase basándose en un número de atributos observados relacionados al objeto. Muchos problemas de negocios, ciencia, industria y medicina pueden ser tratados como problemas de clasificación. Algunos de ellos pueden ser diagnosis médica, control de calidad o reconocimiento de caracteres escritos [4].

Las redes neuronales han emergido como una herramienta importante para la clasificación. Los últimos estudios en clasificación neuronal han establecido que las redes neuronales son una alternativa prometedora a varios métodos de clasificación convencional. La ventaja de éstas reside en una gran variedad de aspectos teóricos, entre los que destacan:

1) Son métodos auto-adaptativos, que pueden ajustarse por sí solos a los datos sin ninguna especificación explícita de forma funcional.
2) Son aproximadores funcionales universales, en el sentido en que pueden aproximar cualquier función con una precisión arbitraria. Ya que cualquier procedimiento de clasificación busca una relación funcional entre la pertenencia a un grupo y las características del objeto, la identificación exacta de esta función es, sin duda, importante.
3) Las redes neuronales son modelos no lineales, lo que las hace flexibles a la hora de modelar las relaciones complejas del mundo real.
4) Finalmente, son capaces de estimar probabilidades a posteriori, lo que proporciona las bases para establecer la regla de clasificación y el análisis estadístico.

La eficacia de clasificación de las redes neuronales ha sido probada empíricamente. Para ello, han sido aplicadas satisfactoriamente en una gran variedad de tareas de clasificación de la vida real relacionadas con la industria, empresa y ciencia. Se han utilizado en tareas tales como predicción de bancarrota, reconocimiento de textos escritos, reconocimiento de voz, inspección de productos y diagnosis médica.

Se verá en el próximo capítulo que existen muchos tipos de redes, pero este trabajo se centrará en las redes de tipo feedforward. Estas redes han sido usadas en muchos campos debido a sus habilidades de aproximación, así como su capacidad de proporcionar modelos

para un amplio rango de fenómenos naturales o artificiales que son difíciles de manipular usando las técnicas clásicas. Como contrapartida, son necesarios algoritmos más rápidos, ya que los tradicionales son normalmente bastante más lentos de lo requerido. No es sorprendente que el entrenamiento de una red pueda durar varias horas, varios días o incluso más [5].

Muchos investigadores han explorado, desde un punto de vista matemático, las capacidades universales de aproximación de las redes neuronales de tipo feedforward, centrándose en dos aspectos: aproximación sobre conjuntos de entrada compactos y aproximación sobre conjuntos de un número infinito de muestras de entrenamiento. Así, se han realizado algunos desarrollos científicos que son claves para que hoy en día se puedan llevar a cabo proyectos como éste. Por ejemplo, Hornik [6] demostró que si la función de activación usada en la red es continua, limitada y no constante, se pueden aproximar aplicaciones continuas a través de conjuntos de entrada compactos. Leshno [7] mejoró los resultados de Hornik [6] y demostró que las redes de este tipo con una función de activación no polinómica pueden aproximar funciones reales. En aplicaciones reales, las redes neuronales se forman con conjuntos finitos de entrenamientos. Para conjuntos de entrenamiento finito, Huang y Babri [8] muestran que una red neuronal de tipo feedforward de una sola capa oculta (SLFN) con un máximo de N nodos ocultos y con cualquier función de activación no lineal puede llegar a aproximar exactamente N observaciones distintas. Para ello hay que denotar que tanto los pesos como los sesgos de la capa oculta deben ser ajustados previamente de forma teórica tal y como se hace en la mayoría de los algoritmos de aprendizaje prácticos de las redes neuronales de este tipo.

Tradicionalmente, todos los parámetros de las redes feedforward necesitaban ser entrenados y por lo tanto existía una dependencia entre todos ellos (pesos y sesgos). Durante las últimas décadas, los métodos de descenso por gradiente han sido principalmente usados en varios algoritmos de aprendizaje para estas redes. Sin embargo, se sabe que estos algoritmos son más lentos debido a posibles pasos de aprendizaje incorrectos o porque pueden converger fácilmente a mínimos locales. Además, necesitan muchas iteraciones para obtener un buen funcionamiento.

Está demostrado [9,10] que las redes (con N nodos ocultos) con pesos y sesgos de entrada elegidos de forma aleatoria (estos nodos se pueden llamar nodos aleatorios ocultos) pueden aprender exactamente N observaciones distintas. A diferencia del pensamiento popular y la mayoría de implementaciones prácticas en las que los parámetros deben ser ajustados, en estas redes no es necesario. De hecho, resultados de simulaciones de grandes aplicaciones, tanto reales como artificiales, han demostrado que este método no sólo produce un aprendizaje más rápido sino que también se obtiene una mayor capacidad de generalización.

En el capítulo tercero de este proyecto se probará que los pesos y los sesgos de entrada de este tipo de redes pueden ser asignados aleatoriamente si la función de activación de la capa oculta es infinitamente diferenciable. Después de hacer esto, las redes SLFN pueden ser consideradas simplemente como un sistema lineal y los pesos de salida pueden ser analíticamente calculados a través de una simple operación inversa con las matrices de la capa de salida. Basado en este concepto, se trabajará con un simple algoritmo de aprendizaje

llamado Extreme Learning Machine (ELM) cuya velocidad de aprendizaje puede ser miles de veces mayor que la de los algoritmos tradicionales tales como Backpropagation (BP), mientras se obtiene mayor generalización. A diferencia de los algoritmos tradicionales de aprendizaje, el propuesto no sólo tiende al menor error de entrenamiento sino que también a la mínima norma de los pesos calculados. La teoría de Bartlett [11] sobre la generalización de redes neuronales de tipo feedforward expresa que cuanta más pequeña es la norma de los pesos, mejor capacidad de generalización tienden a tener las redes.

Por tanto, se introducirán los conceptos teóricos y se verá cómo el nuevo algoritmo propuesto es fácil de implementar, tiende al menor error de entrenamiento posible, obtiene la menor norma de los pesos y mejor generalización, ejecutándose extremadamente rápido.

Con los años se ha descubierto que un aspecto muy importante de los algoritmos de aprendizaje máquina es la capacidad de representación de los datos. Por esta razón, cada vez es más frecuente el desarrollo de estructuras de preprocesado y transformación de datos que den como resultado una representación que pueda soportar un aprendizaje efectivo. En este caso se estudiará la estructura de los Auto-Codificadores (*Auto Encoders)* y se comparará su funcionamiento con un algoritmo muy conocido como es el de *Descomposición en Valores Singulares (SVD)*, a la hora de representar grandes conjuntos de datos.

Los auto-codificadores juegan un papel fundamental en el aprendizaje no supervisado y en las arquitecturas de redes de aprendizaje profundo, llamadas “arquitecturas profundas”, y son usados para una gran variedad de tareas. A pesar del importante papel que tienen, sólo se han resuelto analíticamente auto-codificadores lineales sobre números reales. En este trabajo se presentará una estructura matemática general que permitiría el estudio de ambos tipos, lineales y no lineales [12]. Dicha estructura es válida para definir el concepto básico de aquella con la que se trabajará en este proyecto. En capítulos posteriores se usarán estos auto-codificadores para trabajar con lo que en español se llama Grandes Datos (*Big Data* en inglés), que como uno puede imaginarse es un término utilizado para sistemas que manejan grandes conjuntos de datos.

En el Capítulo 4 se presentará una nueva estructura para solventar algunos de los problemas que pueden surgir con las redes tradicionales a la hora de procesar Grandes Datos. Estas arquitecturas están orientadas al Aprendizaje Profundo o *Deep Learning* en inglés y son llamadas Redes Profundas. El aprendizaje profundo es un campo perteneciente a la *Inteligencia Artificial* que ha sido desarrollado recientemente. Intenta imitar el cerebro humano, que es capaz de procesar una gran cantidad de datos complejos de entrada, adquirir diferentes conocimientos rápidamente y solucionar de forma adecuada varios tipos de tareas complicadas. Combinando estas características del cerebro en un modelo de aprendizaje, se desea que éste pueda tratar con datos de grandes dimensiones, soportar un algoritmo rápido y eficiente, y que funcione correctamente ante complicadas tareas de IA como el reconocimiento de voz.

Sin embargo, en este caso, se aplicarán los conceptos y algoritmos estudiados al procesado de imágenes médicas. La imagen médica es el principal método de obtención de información del cuerpo humano. Desde la invención de los rayos-X en 1895, la investigación y desarrollo de la

imagen médica ha continuado obteniendo grandes aplicaciones tecnológicas. Se puede decir que en estos últimos veinte o treinta años casi todas las aplicaciones desarrolladas utilizan la imagen médica de distintas formas y con la ayuda de las redes neuronales en la última década se han abierto muchas puertas para los investigadores. Tanto es así que todos los artículos publicados recientemente muestran la necesidad del tratamiento con imágenes médicas para solucionar problemas de salud [13].

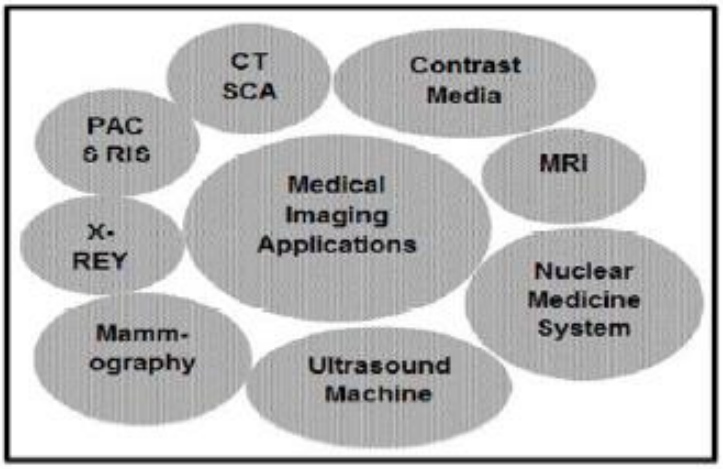


Figura 1.2: Aplicaciones de las imágenes médicas.

Idealmente, éste método consiste en monitorizar y observar el cuerpo de las personas para una posible enfermedad y/o lesión. Desde el punto de vista del paciente, el uso de imágenes médicas mejora la calidad de los servicios que les son proporcionados, y se mejora la toma de decisiones sobre un terreno que se basa en las evidencias.

Como se verá, el uso de las redes neuronales puede mejorar en gran medida el tratamiento de las imágenes médicas gracias a su buen entrenamiento y eficiencia, siempre y cuando éstas formen parte de las aplicaciones mencionadas. En resumen, la aplicación de redes neuronales para el procesamiento de imágenes médicas es uno de los temas más desarrollados en la última década.

En este proyecto se pretende trabajar con ecografías de la arteria carótida para buscar un método de predicción de la aterosclerosis. Esta enfermedad consiste en un engrosamiento de las paredes de las arterias. Aunque es posible que en estos últimos años no se hayan escuchado muchas noticias sobre este tema, la aterosclerosis es muy frecuente y puede ser la causa de infartos cerebrales, embolias o isquemia. Así, muchos estudios médicos se centran en la prevención de esta enfermedad. El grosor íntima-media (IMT) ha emergido como un indicador fiable de prevención de la aterosclerosis, haciendo posible evitar el empeoramiento de esta enfermedad y, por tanto, el riesgo cardiovascular [14].

Establecer un protocolo de medida del IMT es algo necesario en este campo, ya que hoy en día existen dos problemas importantes: el uso de diferentes protocolos de medida y la variabilidad entre los diferentes observadores del protocolo de medida manual. De esta forma, el método que se verá en el último capítulo podría proporcionar la repetitividad y reproducibilidad

necesarias como para que la medida del IMT sea considerado como un indicador fiable de aterosclerosis.

Por tanto, los objetivos de este proyecto se pueden resumir como:

1) En primer lugar, estudiar tanto el funcionamiento como la respuesta de las Máquinas de Aprendizaje Extremo (ELM) ante problemas de clasificación con Grandes Datos.

2) En segundo lugar, demostrar la buena capacidad de representación de un Auto-Codificador (*Autoencoder, AE*) basado en el algoritmo de las ELM, comparándolo con un método eficiente como es el de Descomposición en Valores Singulares de una matriz.

3) Después, se estudiará cómo a partir de la unión de varios Auto-Codificadores se puede construir un tipo de red multicapa llamada Máquina de Aprendizaje Extremo Multicapa (ML-ELM), capaz de mejorar los resultados obtenidos con las ELMs a la hora de procesar grandes cantidades de datos.

4) Finalmente, se pretende aplicar todo lo anterior al procesado de imágenes médicas, con el objetivo de medir el grosor de la capa íntima-media de la arteria carótida.

En busca de estos objetivos, la estructura del proyecto quedará de la siguiente forma: en el Capítulo 2 se continuará con una profundización teórica sobre las redes neuronales artificiales. Después se presentará la teoría relativa al algoritmo de aprendizaje en el que se centra este proyecto, viendo la base del funcionamiento de las Máquinas de Aprendizaje Extremo (*ELM*) y comprobando su funcionamiento ante un problema de clasificación. Será entonces cuando se muestre de forma detallada la estructura del Auto-Codificador (AE) y de las Máquinas de Aprendizaje Extremo Multicapa (ML-ELM), comparando éstas últimas con las ya mencionadas ELMs. Finalmente, se aplicarán todos estos conceptos sobre el procesado de imágenes médicas, en concreto, de ecografías con el objetivo de la medida del grosor íntima-media carotídeo. El trabajo se cerrará con un capítulo de conclusiones.

CAPÍTULO 2.

REDES NEURONALES ARTIFICIALES

Las actividades de investigación desarrolladas en torno al estudio de *redes neuronales artificiales*, simplemente redes neuronales, están motivadas a modelar la forma de procesamiento de la información en sistemas nerviosos bilógicos [15]. Especialmente, por la forma de funcionamiento del cerebro humano, que es completamente distinta al funcionamiento de un computador digital convencional. El cerebro humano corresponde al de un sistema altamente complejo, no lineal y paralelo. En término sencillos lo anterior equivale a decir que puede realizar muchas operaciones simultáneamente, a diferencia de los computadores comunes que son de tipo secuencial. En este sentido una red neuronal es un procesador de información, de distribución altamente paralela, constituido por muchas unidades sencillas de procesamiento llamadas neuronas. Se caracterizan principalmente por:

1) Tener una inclinación natural a adquirir el conocimiento a través de la experiencia, el cual es almacenado, al igual que en el cerebro, en el peso relativo de las conexiones interneuronales.
2) Tienen altísima plasticidad y adaptabilidad, son capaces de cambiar dinámicamente junto con el medio.
3) Poseen un alto nivel de tolerancia a fallos.
4) Tener un comportamiento no lineal puede permitirles procesar información procedente de otros fenómenos no lineales.

Entre las motivaciones principales, para el estudio del funcionamiento de las redes neuronales se encuentran los fenomenos neurológicos. Nuestro cerebro es un procesador de información muchísimo más eficiente que un computador. La clave de esto se encuentra en la inmensa plasticidad del cerebro. Existen tareas cotidianas para el cerebro que sería impensable realizar mediante computación tradicional. Un ejemplo de esto es la capacidad de reconocer a una persona en un tiempo de 100 a 200 ms. La plasticidad se percibe también en la capacidad de responder de forma correcta frente a un estímulo nunca antes recibido. Esa capacidad hace que cuando nos presentan por primera vez a alguien, sepamos que es una persona y no un objeto u otro ser biológico. Debido a estas características y muchas otras, las redes neuronales se han convertido en una gran ayuda en el procesamiento de datos experimentales de comportamiento complejo.

2.1. HISTORIA DE LAS REDES NEURONALES

En 1943, el neurobiólogo Warren McCulloch, y el estadístico Walter Pitss, publicaron el artículo "*A logical calculus of Ideas Imminent in Nervous Activity*". Este artículo constituyó la base y el inicio del desarrollo en diferentes campos como son los Ordenadores Digitales (John Von Neuman), la Inteligencia Artificial (Marvin Minsky con los Sistemas Expertos) y el funcionamiento del ojo (Frank Rosenblatt con la famosa red llamada Perceptrón).

Hebb definió en 1949 dos conceptos muy importantes y fundamentales que han pesado en el campo de las redes neuronales, basándose en investigaciones psicofisiológicas:

- El aprendizaje se localiza en las sinapsis o conexiones entre neuronas.
- LA información se representa en el cerebro mediante un conjunto de neuronas activas o inactivas.

En 1956, los pioneros de la Inteligencia Artificial, Minsky, McCarthy, Rochester, Shanon, organizaron la primera conferencia de Inteligencia Artificial que fue patrocinada por la Fundación Rochester. Esta conferencia se celebró en el verano de 1956 en la localidad inglesa de Darmouth y en muchos libros se hace referencia al verano de este año como la primera toma de contacto seria con las redes neuronales artificiales.

Nathaural Rochester del equipo de investigación de IBM presentó el modelo de una red neuronal que él mismo realizó y puede considerarse como el primer software de simulación de redes neuronales artificiales.

En 1957, Frank Rosenblatt publicó el mayor trabajo de investigación en computación neuronal realizado hasta esas fechas. Su trabajo consistía en el desarrollo de un elemento llamado “Perceptrón”.

El perceptrón es un sistema clasificador de patrones que puede identificar patrones geométricos y abstractos. El primer perceptrón era capaz de aprender algo y era robusto, de forma que su comportamiento variaba sólo si resultaban dañados los componentes del sistema. Además presentaba la característica de ser flexible y comportarse correctamente después de que algunas celdas fueran destruidas.

El perceptrón fue originalmente diseñado para el reconocimiento óptico de patrones. Una rejilla de 400 fotocélulas, correspondientes a las neuronas de la retina sensibles a la luz, recibe el estímulo óptico. Estas fotocélulas están conectadas a elementos asociativos que recogen los impulsos eléctricos emitidos desde las fotocélulas. Las conexiones entre los elementos asociativos y las fotocélulas se realizan de forma aleatoria.

Si las células presentan un valor de entrada superior a un umbral predeterminado entonces el elemento asociativo produce una salida. La figura inferior presenta la estructura de la red perceptrón.

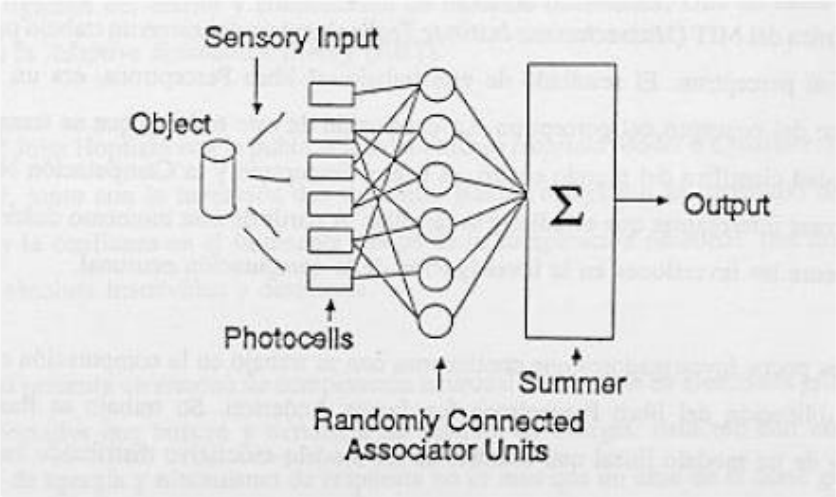


Figura 2.1: Red Perceptrón.

El perceptrón presenta algunas limitaciones debido a que se trataba de un dispositivo en desarrollo. La mayor limitación la reflejaron Minsky y Papert años más tarde, y ponían de manifiesto la incapacidad del perceptrón en resolver algunas tareas o problemas sencillos como por ejemplo la función lógica OR exclusivo.

Uno de los mayores cambios realizados en el perceptrón de Rossenblatt a lo largo de la década de los 60 ha sido el desarrollo de sistemas multicapa que pueden aprender y categorizar datos complejos.

En 1959, Bernard Widrow en Stanford desarrolló un elemento adaptativo lineal llamado "Adaline" (*Adaptive Linear Neuron*). La Adaline y una versión de dos capas, llamada "Madaline", fueron utilizadas en distintas aplicaciones como reconocimiento de voz y caracteres, predicción del tiempo, control adaptativo y sobre todo en el desarrollo de filtros adaptativos que eliminen los ecos de las líneas telefónicas.

A mediados de los años 60, Minsky y Papert pertenecientes al Laboratorio de Investigación de Electrónica del MIT (*Massachussets Institute Technology*) comenzaron un trabajo profundo de crítica al perceptrón. El resultado de este trabajo, el libro Perceptrons, era un análisis matemático del concepto del perceptrón. La conclusión de este trabajo, que se transmitió a la comunidad científica del mundo entero, es que el perceptrón y la Computación Neuronal no eran temas interesantes que estudiar y desarrollar. A partir de este momento descendieron drásticamente las inversiones en la investigación de la computación neuronal.

Uno de los pocos investigadores que continuaron con su trabajo en la computación neuronal tras la publicación del libro Perceptrons fue James Anderson. Su trabajo se basó en el desarrollo de un modelo lineal que consiste en un modelo asociativo distribuido basado en el principio de Hebb. Una versión extendida de este modelo lineal es el llamado modelo *Brain-State-in-a-Box* (BSB).

Teuvo Kohonen, de la Universidad de Helsinki, es uno de los mayores impulsores de la computación neuronal de la década de los 70. De su trabajo de investigación destacan dos aportaciones: la primera es la descripción y análisis de una clase grande de reglas adaptativas,

reglas en las que las conexiones ponderadas se modifican de una forma dependiente de los valores anteriores y posteriores de las sinapsis. Y la segunda aportación es el principio de aprendizaje competitivo en el que los elementos compiten por responder a un estímulo de entrada, y el ganador se adapta él mismo para responder con mayor efecto al estímulo.

Otro investigador que continuó con su trabajo de investigación en el mundo de la computación neuronal a pesar del mal presagio que indicaron Minsky y Papert fue Stephen Grossberg. Grossberg estaba especialmente interesado en la utilización de datos de la neurología para construir modelos de computación neuronal. La mayoría de sus reglas y postulados derivaron de estudios fisiológicos. Su trabajo ha constituido un gran impulso en la investigación del diseño y construcción de modelos neuronales. Una de estas clases de redes es la *Adaptive Resonance Theory* (ART).

Es en los años 80 cuando Rumelhart, McClelland y Hinton crearon uno de los mayores grupos de investigación de los últimos años, PDP (*Parallel Distributed Processing*), del que salieron los manuales con mayor influencia desde el trabajo descorazonador de Minsky y Papert.

En 1982 John Hopfield con la publicación del artículo *Hopfield Model* o *Crossbar Associative Network*, junto con la invención del algoritmo Backpropagation se consiguió devolver el interés y la confianza en el fascinante campo de la computación neuronal tras dos décadas de casi absoluta inactividad y desinterés.

Hopfield presenta un sistema de computación neuronal consistente en elementos procesadores interconectados que buscan y tienden a un mínimo de energía. Esta red con este tipo de función de energía y mecanismo de respuesta no es más que un caso de la clase genérica de redes que consideró Grossberg.

En 1984, Kohonen desarrolla los Mapas de Kohonen, redes basadas en aprendizaje competitivo, con una idea nueva basada en la biología: las unidades de procesos físicamente adyacentes aprenderán a representar patrones de entrada similares, así las neuronas de salida adyacentes identifican patrones similares.

Rumelhart de la Universidad de Stanford es uno de los principales impulsores de la red más utilizada en la mayoría de las aplicaciones, la famosa red neuronal Backpropagation. En la Universidad de Carnegie-Mellon, el grupo de investigación a la cabeza con McClelland destaca por el estudio de las posibles aplicaciones del Backpropagation. Y en la Universidad de Toronto, Hinton y Sejnowski desarrollaron en 1986 una máquina llamada Boltzman que consiste en la red de Hopfield con dos modificaciones significativas. Ya en 1987 Bart Kosko diseña una red llamada BAM (*Bidirectional Associate Memory*) basado en la red de Grossberg.

En la actualidad gracias a diversos grupos de investigación repartidos por universidades de todo el mundo las redes neuronales han alcanzado una madurez muy aceptable y se usan en todo tipo de aplicaciones [16].

2.2. LA NEURONA BIOLÓGICA

Como ya se ha mencionado en varias ocasiones el principal objetivo las redes neuronales es imitar el funcionamiento del cerebro, por lo que no se puede pasar por alto estudiar brevemente la neurona biológica.

Una neurona típica posee el aspecto que se muestra en la Figura 2.2. Sin embargo, debemos observar que el dibujo no está a escala, el axón alcanza una longitud típica de centímetros y a veces varios metros, las dendritas también y las terminales sinápticas son más largas y numerosas.

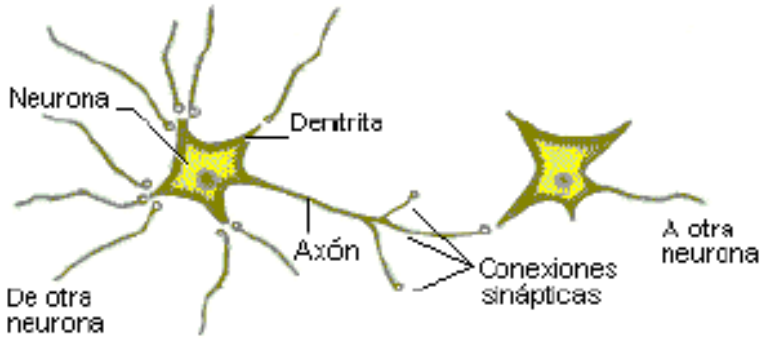


Figura 2.2: Neurona biológica.

Típicamente, las neuronas son 5 o 6 órdenes de magnitud más lentas que una compuerta lógica de silicio: en un chip de silicio los eventos toman alrededor de nanosegundos, mientras que en una neurona el tiempo es de milisegundos [15]. Sin embargo, el cerebro compensa en forma excepcional la lentitud relativa en el funcionamiento neuronal con un número inmenso de neuronas con interconexiones masivas entre ellas. Se estima que el número de neuronas en el cerebro es del orden de 10^{10}, y que el número de conexiones sinápticas es de $6 \cdot 10^{13}$. La red resultante que es el cerebro es una estructura enormemente eficiente. Específicamente, la eficiencia energética del cerebro es aproximadamente de 10^{16} J (operaciones/s), lo cual es del orden de 10^{10} veces mayor que la de los mejores computadores de actualidad.

La mayoría de las neuronas codifican sus salidas como una serie de breves impulsos periódicos, llamados *potenciales de acción*, que se originan cercanos al soma de la célula y se propagan a través del axón. Luego, este pulso llega a las sinapsis y de ahí a las dendritas de la neurona siguiente.

Una *sinapsis* es una interconexión entre dos neuronas. Un dibujo esquemático de ella se incluye en la Figura 2.3. En ella, el botón sináptico corresponde al término del axón de una neurona pre-sináptica, y la dendrita es la correspondiente a una neurona post-sináptica.

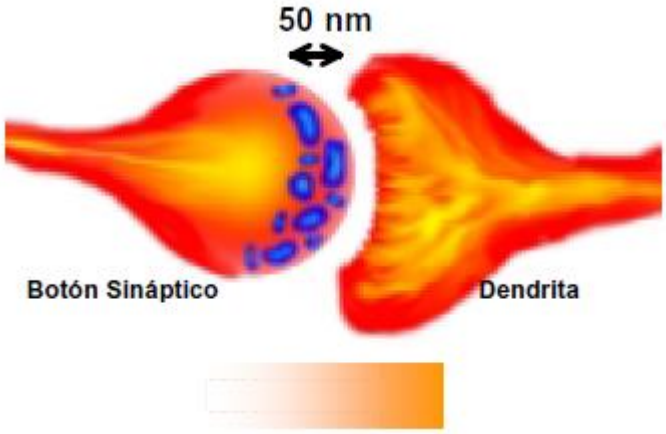


Figura 2.3: Sinapsis.

En la neurona hay dos comportamientos que son importantísimos para este estudio:

- El impulso que llega a una sinapsis y el que sale no son iguales en general. El tipo de pulso que saldrá depende muy sensiblemente de la cantidad de neurotransmisor. Esta cantidad de neurotransmisor cambia durante el proceso de aprendizaje, es aquí donde se almacena la información. Una sinapsis modifica el pulso, ya sea reforzándolo o debilitándolo.
- En el soma se suman las entradas de todas las dendritas. Si estas entradas sobrepasan un cierto umbral, entonces se transmitirá un pulso a lo largo del axón, en caso contrario no transmitirá. Después de transmitir un impulso, la neurona no puede transmitir durante un tiempo de entre 0,5 a 2 ms. A este tiempo se le llama periodo refractario.

En base a estas dos características se construirá el modelo de red neuronal.

2.3. LA NEURONA ARTIFICIAL

Las neuronas artificiales simulan el comportamiento de las descritas en el apartado anterior. Cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, la red neuronal.

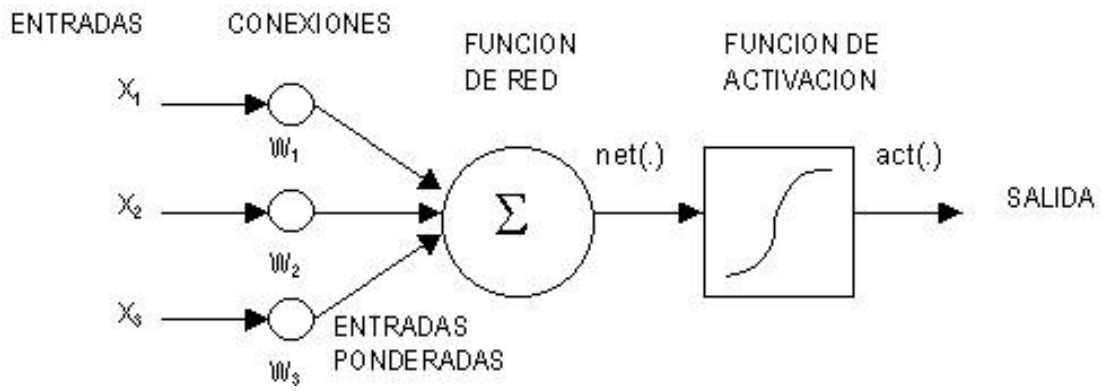


Figura 2.4: Neurona Artificial.

Como se ve en la ilustración superior, dicha unidad de proceso consta de una serie de entradas X_i que equivalen a las dendritas (de donde reciben la estimulación), que ponderadas por unos

pesos W_i (que representan cómo son evaluados los impulsos entrantes) y combinadas con la función de red o ponderación, se obtendrá el nivel de potencial de la neurona. La salida de la función de red es evaluada en la función de activación que da lugar a la salida de la unidad de proceso. Por tanto, se puede observar que se comporta como una neurona biológica pero de una forma muy simplificada.

Una representación vectorial del funcionamiento básico de una neurona artificial se indica según la siguiente expresión:

$$O = f(X * W)$$

Donde la función f puede ser una función lineal, una función umbral o una función no lineal que simule con mayor exactitud las características de transferencia no lineales de las neuronas biológicas.

Por las entradas X_i llegan unos valores que pueden ser enteros, reales o binarios. Estos valores podrían ser señales que enviarían otras neuronas. Los pesos que hay en las sinapsis W_i equivaldrían en la neurona biológica a los mecanismos que existen en las sinapsis para transmitir la señal. De esta forma la unión de estos valores equivale a las señales químicas inhibitorias y excitadoras que se dan en las sinapsis y que inducen a la neurona a cambiar su comportamiento.

Estos valores son la entrada de la función de red que los convierte en uno solo llamado típicamente el potencial, que en la neurona biológica equivaldría al total de señales que le llegan a la neurona por sus dendritas. Esta función suele ser una suma ponderada de las entradas y los pesos.

La salida de la función de red llega a la de activación que transforma este valor en otro en el dominio que trabajen las salidas de las neuronas. El valor de la salida cumpliría la función de la tasa de disparo de las neuronas biológicas.

A modo de resumen:

- Neuronas biológicas ⟹ Neuronas Artificiales.
- Conexiones sinápticas ⟹ Conexiones ponderadas.
- Efectividad de las sinapsis ⟹ Peso de las conexiones.
- Efecto excitador o inhibidor de una conexión ⟹ Signo del peso de una conexión.
- Efecto combinado de las sinapsis ⟹ Función de red.

2.4. FUNCIONES DE LAS NEURONAS ARTIFICIALES

El modelo de neurona que se ha descrito en el apartado anterior modela la neurona como una serie de funciones que se componen entre ellas. Como se ha visto, la función de ponderación o de red hace uso de los valores de entrada y los pesos de las sinapsis, y la función de activación toma este valor para transformarlo en el estado de la neurona.

2.4.1. Función de red o propagación

Esta función se encarga de transformar las diferentes entradas que provienen de la sinapsis en el potencial de la neurona. Normalmente se usa como función la suma ponderada de las entradas multiplicadas por los pesos, y se interpreta como un regulador de las señales que se emiten entre neuronas al ponderar las salidas de unas como entrada de la otra.

Otra regla usada es la distancia euclídea (como en los mapas de Kohonen). En ella, los pesos sinápticos funcionan de manera distinta al anterior ya que lo que hacen es aproximarse lo máximo posible al vector de entrada. Es utilizada en redes no supervisadas para que se ajuste a los patrones. Otra versión de ésta es la función de distancia de Manhattan, que en lugar de usar el cuadrado se usa el valor absoluto. Con esto se obtiene la medida del parecido entre el patrón de entrada X y los pesos W.

2.4.2. Función de activación

La función de activación combina el potencial postsináptico que nos proporciona la función anterior, con el estado actual de la neurona para conseguir el estado futuro de activación. Sin embargo, en muchos casos las redes no toman su propio estado como un parámetro y por tanto no se considera. Esta función es normalmente creciente monótona y las más usadas son:

- Lineal:

  Algunas redes neuronales (como Adaline) usan esta función de activación por su eficiencia y facilidad.

- Escalón:

  Esta función es la más usada para redes neuronales binarias que no es lineal y es muy simple. Algunas redes que la usan son el Perceptrón y Hopfield. Para redes que trabajan en el rango [-1,1] se usa la función signo.

- Hiperbólicas o tangenciales:

  Las redes con salidas continuas (como el perceptrón multicapa con retropropagación) usan esta función ya que su algoritmo de aprendizaje necesita una función derivable.

2.4.3. Función de salida

Esta función convierte el estado de la neurona en la salida hacia la siguiente neurona que se transmite por la sinapsis. Normalmente no se usa y se toma la identidad de manera que la salida es el propio estado de activación de la neurona. Existen algunas redes que transforman su estado de activación en una salida binaria y para eso usan la función escalón como salida. Otra opción es usar funciones probabilísticas como en la máquina de Boltzman, de tal forma que la red no tiene un comportamiento determinista.

2.5. CLASIFICACIÓN DE LAS RNA

En este apartado se abordarán tres clasificaciones de los distintos tipos de redes neuronales en función de sus características más notables.

2.5.1. Topología

Una primera clasificación de las redes de neuronas artificiales que se suele hacer es en función del patrón de conexiones que presenta. Así se definen dos tipos básicos de redes:

2.5.1.1. Redes de propagación hacia delante (feedforward)

Las redes alimentadas hacia delante –generalmente conocidas como redes feedforward- son aquellas en las que, como su nombre indica, la información se mueve en un único sentido, desde la entrada hacia la salida. Estas redes están clásicamente organizadas en "capas". Cada capa agrupa a un conjunto de neuronas que reciben sinapsis de las neuronas de la capa anterior y emiten salidas hacia las neuronas de la capa siguiente. Entre las neuronas de una misma capa no hay sinapsis [17].

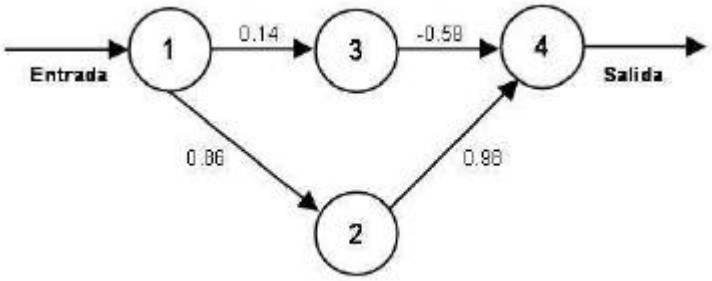


Figura 2.5: Red de tipo feedforward.

En este tipo de redes existe una capa de entrada, formada por las neuronas que reciben las señales de entrada a la red, y una capa de salida formada por una o más neuronas que emiten la respuesta de la red al exterior. Entre la capa de entrada y la de salida existen una o más capas intermedias, pudiendo distinguir por tanto entre dos tipos de redes feedforward:

- Monocapa: ejemplos de este tipo de redes son el Perceptrón y la red Adaline.

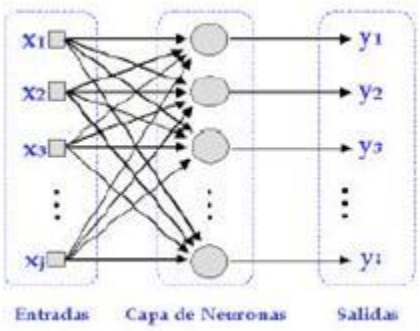


Figura 2.6: Red monocapa.

- Multicapa: un ejemplo es el Perceptrón Multicapa.

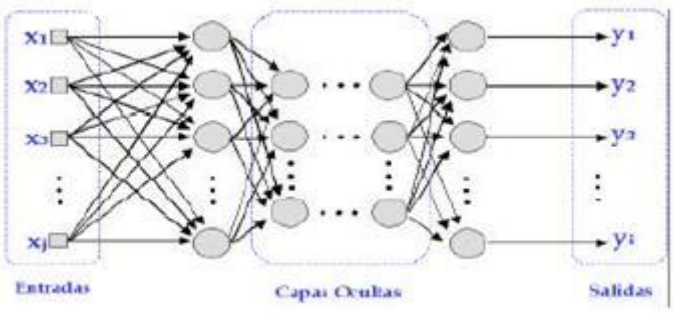


Figura 2.7: Red multicapa.

En redes así construidas es evidente que la información sólo puede moverse en un sentido: desde la capa de entrada hasta la de salida, atravesando todas y cada una de las capas intermedias una sola vez.

El hecho de que no haya conexión entre las neuronas de una misma capa hace que no haya tiempos de espera en los que las neuronas estén interactuando unas sobre las otras hasta que toda la capa adquiera un estado estable. Se trata por tanto de redes rápidas en sus cálculos.

2.5.1.2. Redes con Retroalimentación total o parcial

En este tipo de redes los elementos pueden enviar estímulos a neuronas de capas anteriores, de su propia capa o a ellos mismos, por lo que desaparece el concepto de agrupamiento de las neuronas en capas. Cada neurona puede estar conectada a todas las demás; de este modo, cuando se recibe información de entrada a la red, cada neurona tendrá que calcular y recalcular su estado varias veces hasta que todas las neuronas de la red alcancen un estado estable. Un estado estable es aquel en el que no ocurren cambios en la salida de ninguna neurona. No habiendo cambios en las salidas, las entradas de todas las neuronas serán también constantes, por lo que no tendrán que modificar su estado de activación ni en su respuesta, manteniéndose así un estado global estable [17].

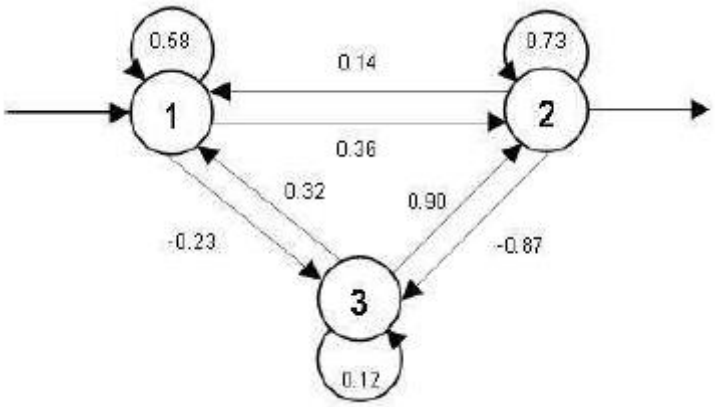


Figura 2.8: Red retroalimentada.

Las redes retroalimentadas emulan más fielmente la estructura del cerebro humano, en donde los fenómenos de retroalimentación son fundamentales. Algunos ejemplos de redes de este

tipo son la red Hopfield o Máquina de Boltzman. Sin embargo, en este proyecto no se trabaja con este tipo de redes por lo que no se profundizará en su funcionamiento.

2.5.2. Aprendizaje

Una segunda clasificación que se suele hacer es en función del tipo de aprendizaje de que es capaz (si necesita o no un conjunto de entrenamiento supervisado). Para cada tipo de aprendizaje encontramos varios modelos propuestos por diferentes autores:

2.5.2.1. Aprendizaje Supervisado

Con aprendizaje supervisado se refiere a todas aquellas aplicaciones o procesos en los que se dispone de información tanto de los valores de entrada del sistema como de los valores de salida deseados. De manera global, dos de los problemas típicos en el aprendizaje supervisado son el de clasificación y regresión. En clasificación, los valores deseados se corresponden con las etiquetas de cada caso (información cualitativa), mientras que en regresión, la información de salida es el valor real a estimar (información cuantitativa) [18].

Un problema de clasificación es, por ejemplo, el reconocimiento de caracteres, cuyo objetivo final es asignar a cada vector de entrada una de las categorías discretas. Expresado de otro modo, en la clasificación supervisada el mapeo de un conjunto de vectores de entrada ($x \in \mathbb{R}^d$, donde d es la dimensión del espacio de entrada) a un conjunto finito de etiquetas discretas ($y = 1 \dots C$, donde C es el número de clases existentes), donde dicho mapeo es modelado en términos de una función matemática $y = f(x, w)$, siendo w un vector de parámetros ajustables. El vector de parámetros se determina mediante un algoritmo de aprendizaje.

Si por el contrario, la salida deseada consiste en una o más de una variable continua, entonces se trata de un problema de regresión. Un ejemplo de un problema de este tipo es la predicción del rendimiento en un proceso de manufacturación químico, en el cual las entradas consisten en concentraciones de reactantes, valores de temperatura o valores de presión.

En la Figura 2.9 puede verse el esquema de un proceso genérico de aprendizaje supervisado. En este proceso, la primera fase es la de obtención de los datos que son necesarios para definir el modelo. Tras ésta, es muy frecuente encontrarse con la necesidad de transformar los datos obtenidos y llevarlos a otro espacio que pueda facilitar la creación de un modelo aprendido y con mayor precisión. Esta fase de preproceso suele denominarse fase de extracción y/o selección de características. Se ha de tener en cuenta que el conjunto de datos resultante de estas dos fases debe ser significativo y representativo; es decir, debe haber un número de ejemplos suficientes y diverso, en el cual todas las regiones significativas del espacio están suficientemente representadas para, de este modo, asegurar la aplicación general del modelo.

La siguiente fase es la más importante de todas, y es la denominada fase de aprendizaje. En esta fase se aplican los distintos algoritmos de aprendizaje y finaliza con la obtención del modelo o patrón. Esta fase se puede subdividir en tres subfases: separación de los datos de entrada, selección del algoritmo y entrenamiento. La primera subfase consiste en separar los datos de entrada en dos subconjuntos totalmente independientes: los datos de

entrenamiento, que se emplean durante el entrenamiento, y los datos de validación, que se utilizan para la validación posterior del modelo generado. La siguiente subfase es la elección del algoritmo apropiado para el problema planteado, teniendo en cuenta el tipo de datos del conjunto, los parámetros de entrada que son conocidos, etc. Existen un gran número de algoritmos que ofrecen distintas posibilidades, el objetivo de esta fase es el de escoger aquel que mejor se adapte a las necesidades del problema. La última subfase del proceso de aprendizaje es la de mayor relevancia; es la fase de entrenamiento y en ella se ejecuta el algoritmo escogido para los parámetros dados. Durante esta fase el algoritmo se ejecuta de manera iterativa obteniendo un error en cada iteración, siendo este error la diferencia entre el valor esperado y el obtenido por el modelo aprendido. Este modelo se ajustará en las sucesivas iteraciones en función de este error. Al final de esta fase se obtendrá un modelo aprendido ajustado a los datos de entrenamiento.

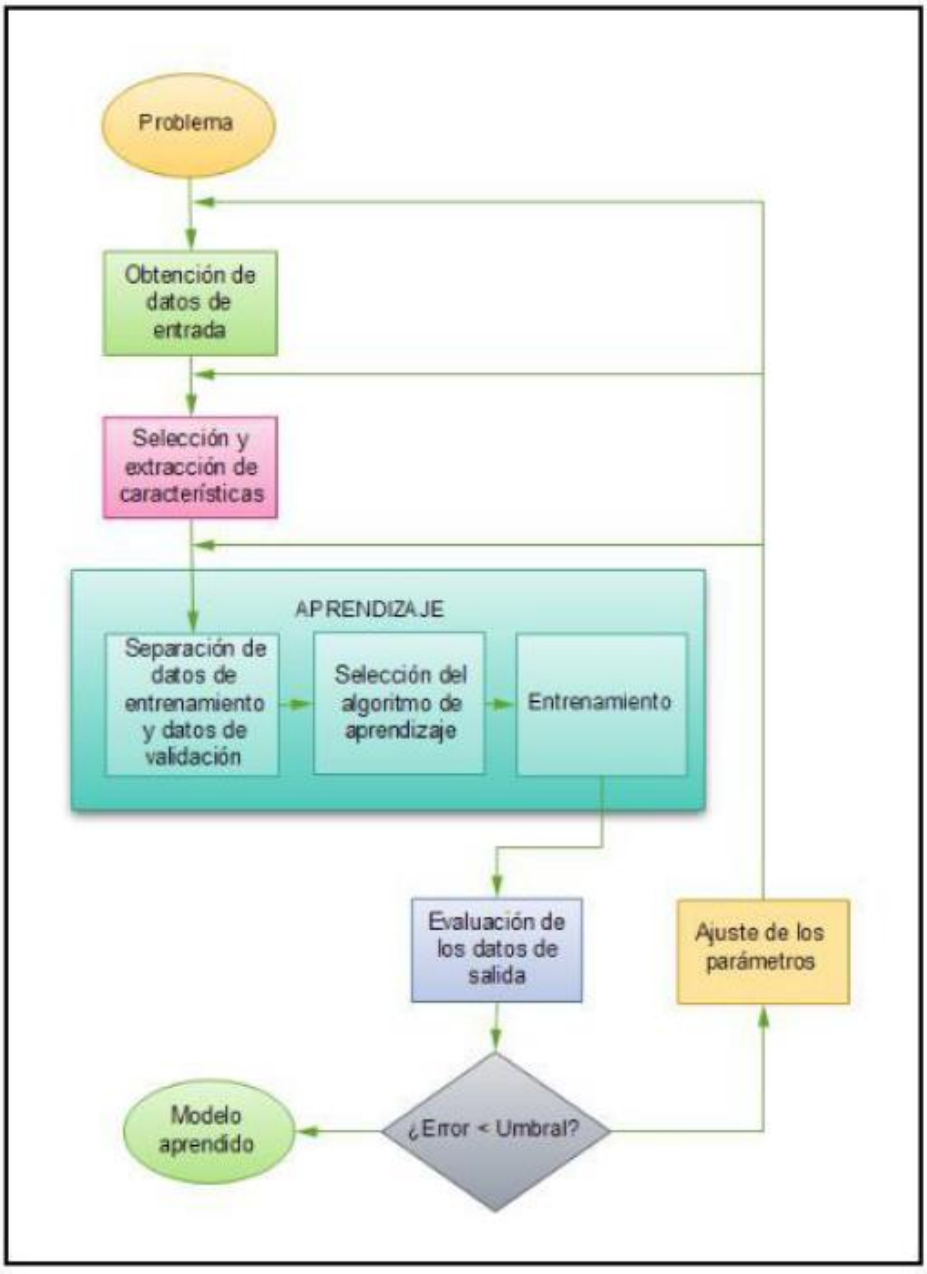


Figura 2.9: Proceso de Aprendizaje Supervisado.

Por último, se ha de comprobar que el modelo obtenido ofrece un resultado idóneo para el problema. Para ello se utiliza un nuevo conjunto de datos, los datos de validación. En esta etapa se comparan los resultados del modelo aprendido con los datos de validación, obteniendo así el error real del modelo. Si el error es inferior a un umbral determinado el proceso de aprendizaje se dará por concluido. Si es mayor, el diseñador del proceso podrá volver a alguna de las fases iniciales para así reajustar o rediseñar el proceso global. Estas fases serán repetidas hasta que el error final sea inferior al umbral definido.

Todas estas fases desempeñan un papel fundamental, y un fallo en alguna de ellas puede llevar a un fallo de clasificación o de generación del modelo. De entre ellas, la fase principal es la de aprendizaje, ya que ésta es la que terminará generando el patrón o modelo.

Son muchas las técnicas que han surgido dentro del aprendizaje supervisado. Una de las primeras técnicas fue K-vecinos más cercanos (KNN K-Nearest Neighbours) y posteriormente surgieron otros métodos como las redes neuronales, los árboles de decisión y las máquinas de vectores de soporte. Algunas redes que usan este tipo de aprendizaje son el Perceptrón, la red Adaline o la red Backpropagation.

2.5.2.2. Aprendizaje No Supervisado o Autoorganizado

En el lado contrario al aprendizaje supervisado se tienen los problemas en los que no se dispone de ningún tipo de información de salida sobre los datos, pero se desea organizar los datos de alguna manera para mejorar su comprensión. Este tipo de problemas son los denominados problemas de aprendizaje no supervisado [18].

Según muestra el libro "*Pattern Classification"* de *R. O. Duda, P. E. Hart y D. G. Stork*, existen cinco razones principales por las cuales el uso de técnicas de aprendizaje no supervisado resulta de interés para las aplicaciones:

- La recolección de datos y su posterior etiquetamiento en un conjunto de datos muy extenso puede suponer un coste muy elevado.
- También puede ser de interés actuar en sentido contrario; aprender con una gran cantidad de datos sin etiquetar, y sólo usar supervisión para etiquetar los distintos grupos encontrados.
- En muchas aplicaciones las características de los patrones cambian lentamente con el tiempo. Si estos cambios pueden rastrearse en un proceso de ejecución sin supervisar, se podrá obtener un resultado más idóneo.
- Se pueden usar métodos no supervisados para encontrar características que serán útiles para la categorización.
- En el comienzo de la investigación puede ser útil tener una visión general de la naturaleza y la estructura de los datos.

En la Figura 2.10 puede verse el esquema de un proceso genérico de aprendizaje no supervisado. Este proceso es similar al proceso de aprendizaje supervisado. Al igual que en el anterior, existen varias fases fundamentales. Las primeras fases, la de obtención de datos y su preproceso (selección y extracción de características), son idénticas a las fases del aprendizaje supervisado. En cambio, la siguiente fase, el aprendizaje, no es la misma. En este caso, al no

disponer de información acerca de la salida, en la fase de entrenamiento no se puede reajustar el modelo en base al error. Pero sigue siendo necesario separar los datos en datos de entrenamiento y datos de validación para decidir si el método es bueno o no. La fase de selección del algoritmo y el entrenamiento también se mantienen. En este caso, la posibilidad de validar si los resultados son correctos no es frecuente, puesto que no se dispone de información de la salida. La manera de decidir cuándo se ha aprendido es viendo si el sistema converge o estableciendo un criterio de parada como puede ser un número de iteraciones de funcionamiento máximo.

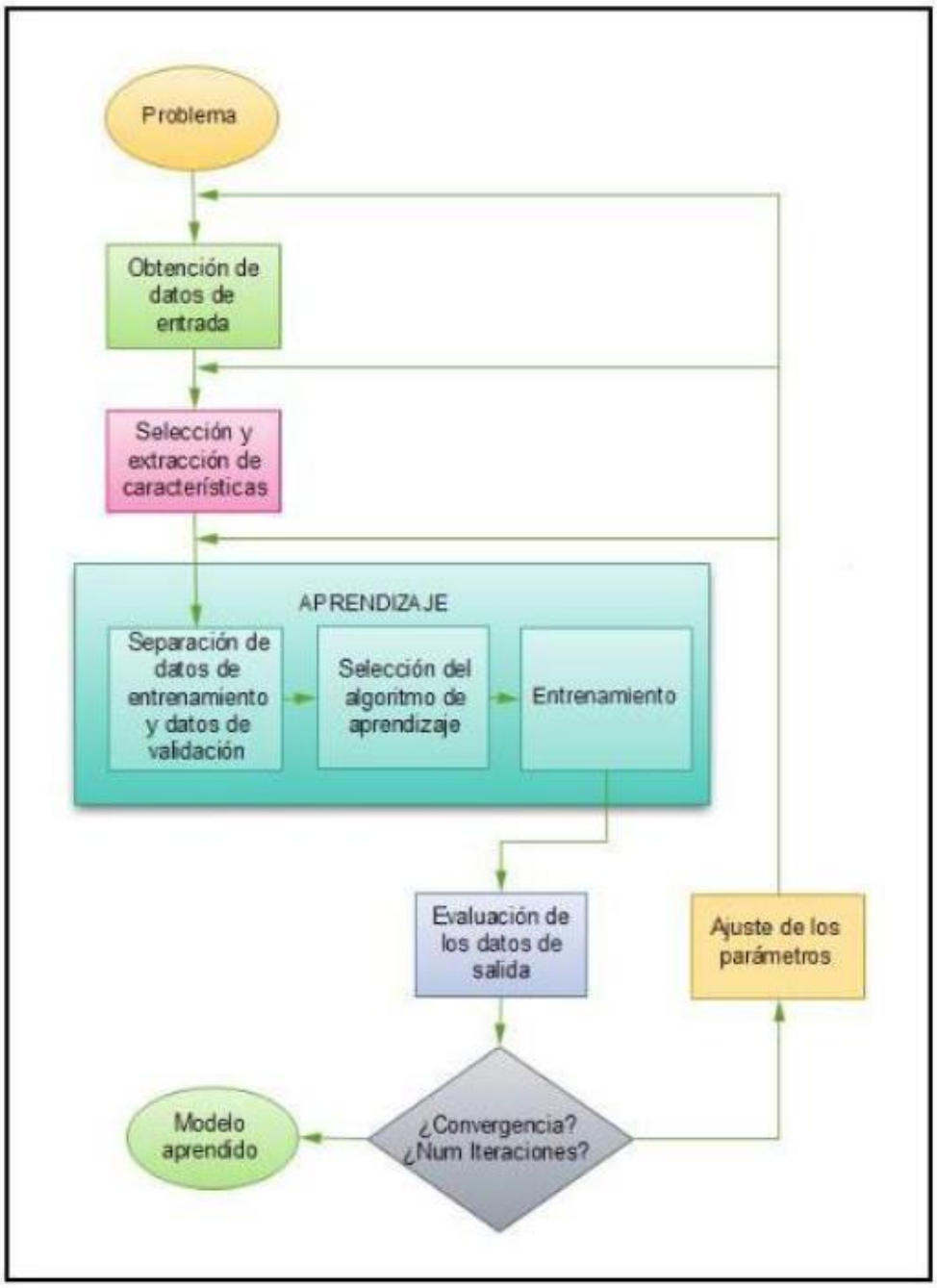


Figura 2.10: Proceso de Aprendizaje no Supervisado.

Existen dos tipos de técnicas fundamentales de aprendizaje no supervisado: agrupamiento o *clustering* y redes neuronales no supervisadas. Ejemplos de este tipo de redes son:

las memorias asociativas, las redes de Hopfield, la máquina de Boltzmann y la máquina de Cauchy, las redes de aprendizaje competitivo, las redes de Kohonen o mapas autoorganizados y las redes de resonancia adaptativa (ART).

2.5.2.3. Aprendizaje por Refuerzo

Otra técnica muy extendida dentro del aprendizaje automático es el aprendizaje por refuerzo. Este tipo de aprendizaje busca las acciones más apropiadas dada una situación concreta, de manera que aumente la recompensa por actuar de ese modo. El objetivo del aprendizaje por refuerzo es usar el paradigma premio-castigo para aprender una función, la cual permitirá tomar decisiones en el futuro a partir de la percepción del entorno. Es frecuente encontrar una secuencia de acciones y de estados por medio de los cuales el algoritmo interactúa con el entorno. En muchos casos, la acción actual no sólo afecta al beneficio o recompensa inmediatos, sino que también puede afectar al beneficio en otros momentos futuros [18].

Las aplicaciones del Aprendizaje por Refuerzo son múltiples, desde robots móviles que aprenden a salir de un laberinto, programas de ajedrez que aprenden cuáles son las mejores secuencias de movimientos para ganar un juego, o un brazo robótico que aprende cómo mover las articulaciones para lograr el movimiento final deseado.

2.5.2.4. Aprendizaje Semi-Supervisado.

El aprendizaje semi-supervisado es una combinación del aprendizaje supervisado y no supervisado. Puesto que asignar etiquetas o clases a los datos puede ser muy costoso, se puede recurrir a la opción de usar a la vez un conjunto de datos etiquetados de tamaño pequeño y un conjunto más extenso de datos no etiquetados, mejorando así la construcción de los modelos. Esta técnica es la usada por el aprendizaje semi-supervisado. En este método, se ha de tener en cuenta que no siempre los datos no etiquetados son de ayuda al proceso de aprendizaje. Por lo general, se asume que los datos no etiquetados siguen la misma distribución que los etiquetados para que el uso de datos sin etiquetar sea útil [18].

Existen numerosos métodos de aprendizaje semi-supervisado, los cuales ofrecen unas características determinadas. La manera de escoger el método más adecuado es viendo cuál de ellos se ajusta mejor a las necesidades del problema específico.

2.5.3. Tipo de Entrada

Finalmente también se pueden clasificar las RNAs según sean capaces de procesar información de distinto tipo en:

2.5.3.1. Las redes analógicas

Procesan datos de entrada de naturaleza analógica, valores reales continuos, habitualmente acotados y usualmente en el compacto [-1,1] o en el [0,1], para dar respuestas también continuas. Las redes analógicas suelen presentar funciones de activación continuas, habitualmente lineales o sigmoides. Entre estas redes neuronales destacan las redes de Backpropagation, la red continua de Hopfield, la Memoria Lineal Asociativa, la Brain-State-in-

Box, y los modelos de Kohonen (mapas auto-organizados (SOM) y Learning Vector Quantizer, (LVQ).

2.5.3.2. Las redes discretas (binarias)

Procesan datos de naturaleza discreta, habitualmente {0,1}, para acabar emitiendo una respuesta discreta. Entre las redes binarias destacan la Maquina de Boltzman, la Máquina de Cauchy, la red discreta de Hopfield, el Cognitrón y el Neogognitrón.

2.6. TIPOS DE REDES NEURONALES

Este capítulo del proyecto ha sido desarrollado con el objetivo de expandir el conocimiento de las redes neuronales y para aumentar las facilidades para programar el marco de trabajo.

Se pueden distinguir varios tipos de redes neuronales en función de la forma en que se organizan las neuronas, como aprenden o el número de capas. Aquí sólo se tratarán algunas de las más importantes y las que se consideran necesarias para el tema que concierne.

2.6.1. Perceptrón Simple

Como ya se ha comentado al principio del capítulo, en 1958 el psicólogo Frank Ronsenblant desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts y en una regla de aprendizaje basada en la corrección del error. A este modelo le llamó Perceptrón. El Perceptrón está constituido por conjunto de sensores de entrada que reciben los patrones de entrada a reconocer o clasificar y una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según que la salida de la misma sea 1 (activada) o 0 (desactivada).

Supongamos que se tiene una función f de R^n en {-1,1}, que aplica un patrón de entrada $\mathbf{x} = (x_1,x_2,\ldots,x_N)^T \in R^n$ en la salida deseada $t \in$ {-1,1}, es decir, $f(\mathbf{x}) = t$. La información de que se dispone sobre dicha función viene dada por p pares de patrones de entrenamiento:

$$\{x_1, t_1\}, \{x_2, t_2\}, \ldots, \{x_p, t_p\}$$

donde $\mathbf{x}_i \in R^n$ y $f(\mathbf{x}_i) = t_i \in$ {-1,1}, i=1,2,…,p. Dicha función realiza una partición en el espacio R^n de patrones de entrada; por una parte estarían los patrones con salida +1 y por otra parte los patrones con salida −1. Por lo tanto, se puede decir que la función f clasifica a los patrones de entrada en dos clases. Ejemplos de funciones f de este tipo son la función lógica OR o la función par.

El Perceptrón presenta dos capas de unidades procesadoras (PE) y sólo una de ellas presenta la capacidad de adaptar o modificar los pesos de las conexiones. La arquitectura del Perceptrón admite capas adicionales pero éstas no disponen la capacidad de modificar sus propias conexiones.

La Figura 2.11 muestra la unidad procesadora básica del Perceptrón. Las entradas x_i llegan por la parte izquierda, y cada conexión con la neurona j tiene asignada un peso de valor w_{ji}.

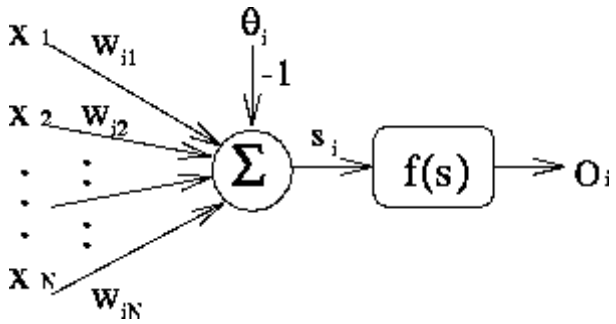


Figura 2.11: Perceptrón simple.

La unidad procesadora del Perceptrón realiza la suma ponderada de las entradas según la ecuación:

$$S_j = \sum x_i w_{ji}$$

Un aspecto común en muchas de las redes neuronales artificiales es la entrada especial llamada "bias" representada en la figura por θ_j. Esta entrada siempre representa un valor fijo y funciona como una masa en un circuito eléctrico donde no varía de valor.

El Perceptrón comprueba si la suma de las entradas ponderadas es mayor o menor que un cierto umbral Δ y genera la salida O_j según la siguiente ecuación:

$$f(s_j) = f(x_1, x_2, \dots, x_n) = \begin{cases} 1 & si \quad w_{j1}x_1 + w_{j2}x_2 + \cdots + w_{jN}x_N \geq \Delta \\ -1 & si \quad w_{j1}x_1 + w_{j2}x_2 + \cdots + w_{jN}x_N < \Delta \end{cases}$$

donde los parámetros $w_{j1}, w_{j2}, \dots, w_{jN}$ se llaman pesos sinápticos y son los pesos con los que se ponderan los valores de entrada $x_1, x_2, \dots, x_n$o argumentos de la función, y la suma ponderada $u = w_{j1}x_1 + w_{j2}x_2 + \cdots + w_{jN}x_N$ es el potencial sináptico del que se ha hablado anteriormente. También se puede expresar la función f mediante la función signo, es decir,

$$f(x_1, x_2, \dots, x_n) = sgn(u - \Delta)$$

Siendo la función signo:

$$sgn(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Y se dirá que en este caso la función de transferencia es la función signo. Cuando la salida de la unidad es igual a 1 se dice que la unidad de proceso está activada y presenta el estado 1, mientras que si su salida es igual a cero se dice que está desactivada, presentando el estado 0. Dos funciones que suelen usarse a la salida son:

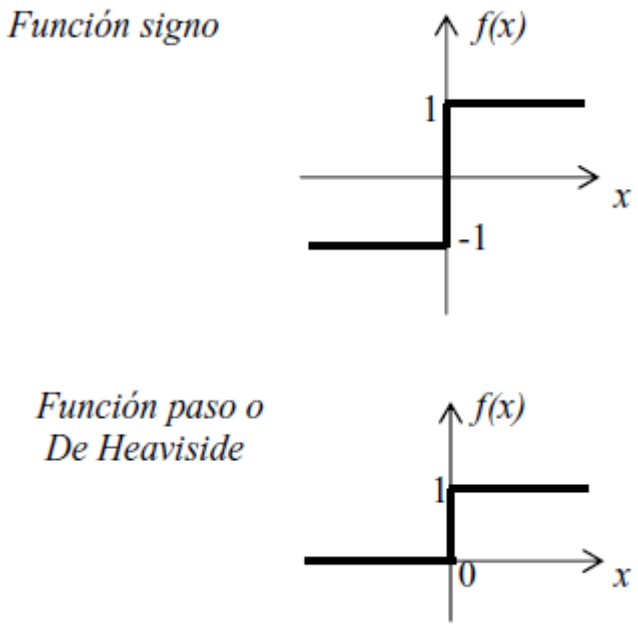


Figura 2.12: Función signo (superior) y Función De Heaviside (inferior).

Las redes Perceptrón de dos capas, representadas en la Figura 2.13 tienen una capa de entrada y una capa de unidades procesadoras que constituyen la capa de salida.

A lo largo de los años 50 y 60 se desarrollaron muchos tipos de topologías de redes basadas en la arquitectura del perceptrón. Las topologías con tres o más capas se caracterizan porque la regla de aprendizaje del perceptrón sólo adapta los pesos o valores de las conexiones de una capa. Una aplicación típica de un sistema de tres capas es la que muestra la Figura 2.14 donde la entrada es la imagen de la letra E y la salida es la categorización de la entrada en dos clases.

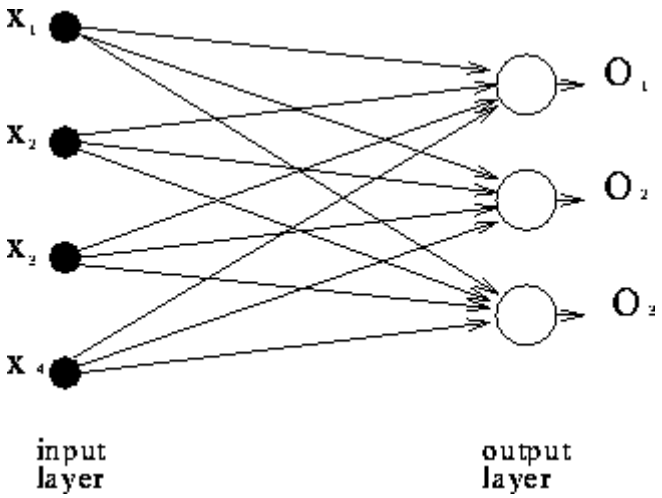


Figura 2.13: Red Perceptrón de dos capas.

El entrenamiento del Perceptrón consiste en presentar a la red todos los elementos del conjunto de entrenamiento constituido por parejas de vectores de forma secuencial.

El objetivo del entrenamiento es llegar a un conjunto de valores de los pesos de la red de forma que responda correctamente a todo el conjunto de entrenamiento. Después del

entrenamiento los pesos no son ya modificados y la red está ya en disposición de responder adecuadamente a las entradas que se le presenten.

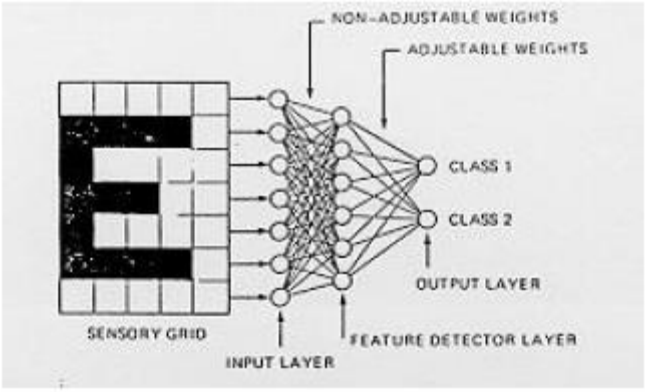


Figura 2.14: Red Perceptrón de tres capas.

La adaptación de los pesos se puede realizar mediante diferentes reglas. Una de las más simples se indica a continuación:

$$w_{ji}(k+1) = w_{ji}(k) + C\ (t_j * O_j)\ x_i$$

Siendo t_j el valor de la salida deseada, O_j el valor de la salida producida por la unidad procesadora, x_i el valor de la entrada i, y C el coeficiente de aprendizaje.

En todo proceso de entrenamiento el comportamiento de la red inicialmente va mejorando hasta que llega a un punto en el que se estabiliza y se dice que la red ha convergido. Esta convergencia tiene dos posibilidades, la primera consiste en que la red hay aprendido correctamente el conjunto de entrenamiento o la segunda se trata de que la red no ha aprendido todas las respuestas correctas.

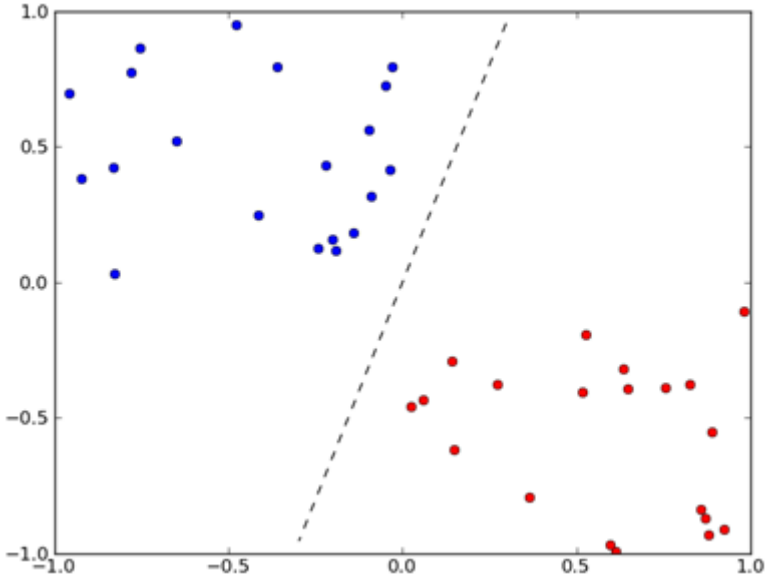


Figura 2.15: Separabilidad lineal.

El mayor inconveniente del Perceptrón, a pesar del éxito que ha tenido en muchas aplicaciones de clasificación de patrones es la imposibilidad de adaptar los pesos de todas las capas. En los años en los que se realizó el Perceptrón, los investigadores no fueron capaces de diseñar un algoritmo que propagara las correcciones de los pesos a través de redes multicapa.

La principal limitación funcional es que una unidad de salida sólo puede clasificar patrones inicialmente separables. La Figura 2.15 ilustra el concepto general de Separabilidad Lineal, es decir, las clases de patrones que pueden separarse en dos clases mediante una línea. Este concepto puede extenderse a más dimensiones separando mediante planos e hiperplanos [16].

2.6.2. Perceptron Multicapa

El Perceptrón multicapa es una red de alimentación hacia adelante (feedforward) compuesta por una capa de unidades de entrada (sensores), otra capa de unidades de salida y un número determinado de capas intermedias de unidades de proceso, también llamadas capas ocultas porque no se ven las salidas de dichas neuronas y no tienen conexiones con el exterior. Cada sensor de entrada está conectado con las unidades de la segunda capa, y cada unidad de proceso de la segunda capa está conectada con las unidades de la primera capa y con las unidades de la tercera capa, así sucesivamente. Las unidades de salida están conectadas solamente con las unidades de la última capa oculta, como se muestra en la Figura 2.16.

Con esta red se pretender establecer una correspondencia entre un conjunto de entrada y un conjunto de salidas deseadas, de manera que:

$$(x_1, x_2, \dots, x_N) \in \mathbb{R}^N \rightarrow (y_1, y_2, \dots, y_M) \in \mathbb{R}^M$$

Para ello se dispone de un conjunto de p patrones de entrenamiento, de manera que se sabe perfectamente la salida correspondiente a cada patrón de entrada. Así, el conjunto de entrenamiento será:

$$\{(x_1^k, x_2^k, \dots, x_N^k) \rightarrow \ (y_1^k, y_2^k, \dots, y_M^k) : k = 1, 2, \dots, p\}$$

Para implementar dicha relación, la primera capa (sensores de entrada) tendrá tantos sensores como componentes tenga el patrón de entrada, es decir, N; la capa de salida tendrá tantas unidades de proceso como componentes tengan las salidas deseadas, es decir, M, y el número de capas ocultas y su tamaño dependerán de la dificultad de la correspondencia a implementar.

Con esta arquitectura se consiguieron solventar las limitaciones de las redes de una sola capa, consiguiendo implementar cualquier función con el grado de precisión deseado, es decir, que las redes multicapa fuesen aproximadores universales. Por ejemplo, si deseamos implementar la función XOR basta con utilizar dos unidades de proceso en la capa oculta, puesto que hay sólo dos patrones de entrada a los que les asignan una salida igual a 1. Al igual que las funciones booleanas, se puede implementar con esta red cualquier función continua de $\mathbb{R}^N$ en $\mathbb{R}^M$, con la diferencia de que las entradas serán número reales (señales analógicas) y las unidades de proceso continuas.

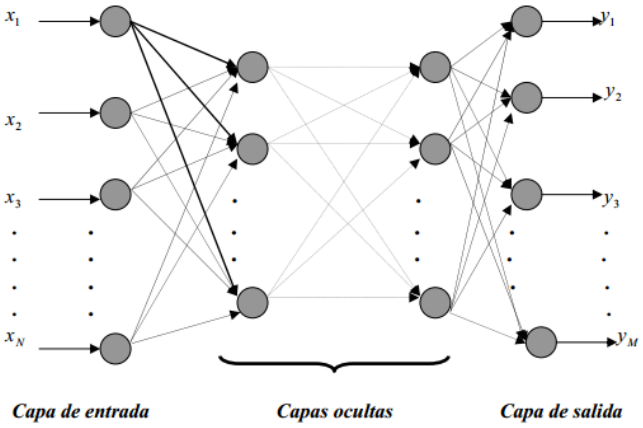


Figura 2.16: Perceptrón Multicapa.

2.6.3. Redes de Funciones de Base Radial (RBF)

Las funciones radiales son simplemente una clase de funciones. En principio, podían ser empleadas en cualquier tipo de modelo lineal o no lineal, y en cualquier tipo de red, monocapa o multicapa. Sin embargo, desde el estudio de Broomhead y Lowe en 1988, las redes de funciones de base radial (redes RBF) han sido asociadas tradicionalmente a redes compuestas por una sola capa oculta, tal y como muestra la Figura 2.17.

Como se puede observar en ella, cada vector de entrada X llega a m funciones base cuyas salidas se combinan linealmente con los pesos $\{w_j\}_{j=1}^{m}$ hacia la salida de la red $f(X)$.

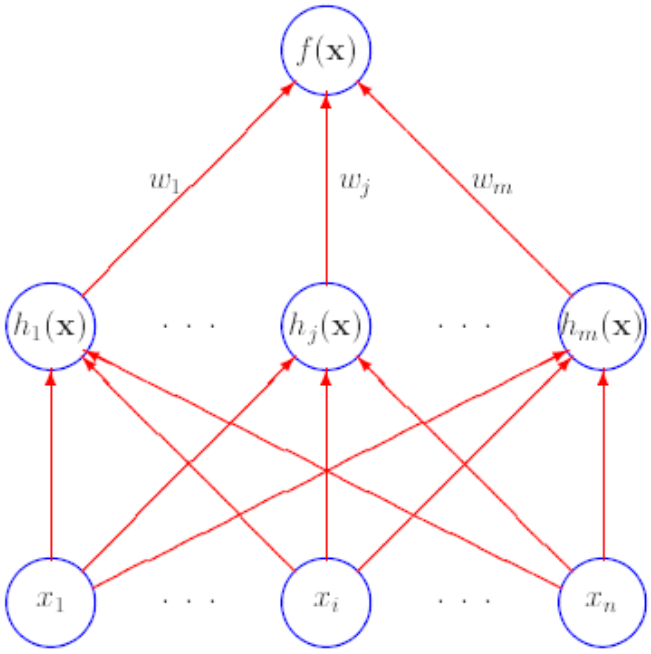


Figura 2.17: Red de Función de Base Radial (RBF).

Una red RBF es no lineal si las funciones base pueden cambiar de tamaño o si posee más de una capa oculta. El caso habitual es el de una red de una sola capa oculta cuyas funciones están fijas en posición y tamaño. Además, suelen usarse algoritmos de optimización no lineal

pero sólo para la regularización de los parámetros en el caso de regresión y para la elección del número óptimo de funciones base (que equivale a decir el número óptimo de nodos ocultos) en selección hacia delante. Por el contrario, en su entrenamiento se evitan el tipo de algoritmos no lineales de descenso por gradiente (tales como el gradiente conjugado o métodos de métrica variable) que son empleados en redes no lineales específicas, ya que manteniéndose firme en el mundo del álgebra lineal se consigue un análisis más sencillo y de menor coste computacional [19].

Por tanto las características principales de este tipo de redes son:

- Las redes de base radial tienen sus orígenes a finales de los años 80.
- Son redes de tipo multicapa que tienen conexiones hacia delante y que sólo tienen una capa oculta. Mientras que las neuronas ocultas poseen carácter local, las neuronas de salida realizan una combinación lineal de las activaciones de las neuronas ocultas.
- Este tipo de redes construyen aproximaciones que son combinaciones lineales de múltiples funciones locales no lineales.
- Entre sus aplicaciones se encuentran análisis de series temporales, procesamiento de imágenes, reconocimiento automático del habla, diagnósticos médicos, etc.

Como ya se ha dicho, las funciones radiales son una clase especial de funciones. La función de base radial es una función que calcula la distancia euclídea de un vector de entrada X respecto de un centro c, de tal manera que resulta la siguiente función:

$$h(x) = (\|x - c_i\|)$$

A cada neurona le corresponde una función de base radial y un peso de salida. El patrón de salida ingresa a una neurona de salida que suma las entradas y da como resultado una salida de la red. De esta forma, la función general de una red RBF final se puede escribir:

$$f(x) = \sum_{j=1}^{m} w_j \, \emptyset(\|x - c_i\|)$$

Una característica importante de estas funciones es que su respuesta decrece (o crece) monótonamente con la distancia del punto central. El centro, la escala de distancias y la forma precisa de la función son los parámetros del modelo, todos fijos si es lineal.

Una función radial típica es la Gaussiana que, en el caso de entradas escalares, es:

$$h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Sus parámetros son su centro c y si radio r. La Figura 2.18 (izquierda) muestra una RBF Gaussiana con centro $c = 0$ y radio $r = 1$.

Otra función típica es la Gaussiana decreciente con la distancia al centro. En contraste con la anterior, es una RBF multicuadrática en la que, para el caso de entradas escalares, decrece con la distancia, tal y como puede verse en la Figura 2.18 (derecha).

$$h(x) = \frac{\sqrt{r^2 + (x - c)^2}}{r}$$

Las funciones de tipo Gaussiana son locales (tienen una respuesta significante sólo cerca del centro) y se usan mucho más frecuentemente que las de tipo multicuadrático, las cuales tienen una respuesta global. Además, son biológicamente recomendables ya que poseen una respuesta finita.

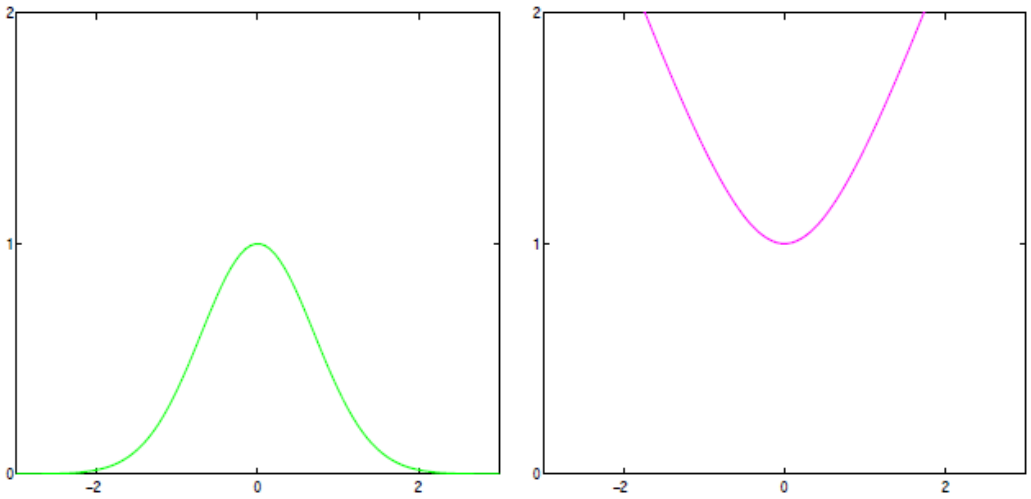


Figura 2.18: Función Gausiana (izquierda) y Multicuadrática (derecha).

CAPÍTULO 3.

MÁQUINA DE APRENDIZAJE EXTREMO (ELM)

Las técnicas de inteligencia computacional han sido usadas para gran variedad de aplicaciones. Entre las numerosas técnicas, las redes neuronales y las máquinas de vectores de soporte (SVM) han jugado un papel dominante. Sin embargo, se sabe que ambos se enfrentan a algunas cuestiones difíciles de resolver tales como su lenta velocidad de aprendizaje.

Recientemente, las Máquinas de Aprendizaje Extremo o *Extreme Learning Machines* (ELM) han atraído la atención de más y más investigadores como una técnica emergente que resuelve algunos problemas a los que se enfrentan otras técnicas. ELM trabaja en redes monocapa hacia delante o de tipo "feedforward" (SLFNs). La esencia del ELM es que la capa oculta de la red no necesita ser entrenada. Comparado con las técnicas de inteligencia computacional tradicionales, ELM mantiene la capacidad de generalización y un entrenamiento mucho más rápido con menor intervención humana. En este capítulo se estudiará en profundidad la teoría relacionada con el ELM para poder usarlo en el siguiente como base para el Aprendizaje Profundo. Esta teoría está fundamentada en el artículo [20] de la bibliografía.

3.1. INTRODUCCIÓN

Existen muchos tipos de redes neuronales (como se ha podido ver en el capítulo anterior). Sin embargo, las redes de tipo "feedforward" son unas de las más populares. A modo de resumen, consisten en una capa de entrada que recibe el estímulo procedente del ambiente externo, una o más capas ocultas, y una capa de salida que envía la salida al exterior.

Normalmente se usan tres aproximaciones principales para el entrenamiento de este tipo de redes:

1) *Método basado en descenso por gradiente*, como por ejemplo Backpropagation para redes multicapa hacia delante. En estos casos se suelen usar nodos ocultos de tipo aditivo, con funciones de activación $g(x)$: $R \rightarrow R$, como:

$$g(x) = \frac{1}{1 + \exp(-x)}$$

De esta forma, la función de salida del nodo i en la capa oculta l viene dada por:

$$G\left(\boldsymbol{a}_i^{(l)}, b_i^{(l)}, \boldsymbol{x}^{(l)}\right) = g\left(\boldsymbol{a}_i^{(l)} \cdot \boldsymbol{x}^{(l)} + b_i^{(l)}\right), b_i^{(l)} \in R$$

donde $\boldsymbol{a}_i^{(l)}$ es el vector de pesos que conecta la capa $(l-1)$ del nodo i con la capa l, y $b_i^{(l)}$ es el sesgo o "bias" del nodo i en la capa l. Los algoritmos de aprendizaje basados en descenso por gradiente son en general mucho más lentos de lo esperado.

2) *Método basado en optimización estándar,* como las máquinas de vectores de soporte (SVM) para un tipo específico de redes monocapa hacia delante: las llamadas redes de vectores de soporte.
Rosenblatt estudió un mecanismo de aprendizaje donde sólo se ajustaban los pesos de la última capa oculta y la capa de salida. Después de haber fijado todos los pesos los datos de entrada son transformados a un espacio de características Z de la última capa oculta (Figura 3.1). En este espacio de características se construye una función de decisión lineal:

$$f(x) = sign\left(\sum_{i=1}^{L} \beta_i z_i(x)\right)$$

Donde $\boldsymbol{\beta_i}$ son los pesos de la capa de salida entre el nodo de salida y la neurona i de la última capa oculta de un Perceptron, y $z_i(x)$ es la salida de la neurona i en la última capa oculta del Perceptron.

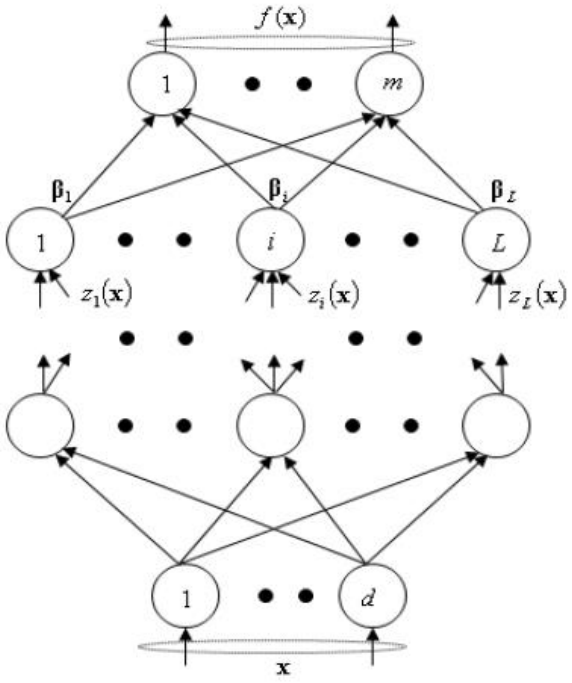


Figura 3.1: Red de tipo feedforward multicapa.

Para encontrar una solución alternativa de $z_i(x)$, en 1995 Cortes y Vapnik [21] propusieron la máquina de soporte de vectores que mapea los datos del espacio de entrada a un espacio de características Z de mayor dimensión, a través de un mapeo no lineal escogido a priori. Los métodos de optimización son usados para encontrar el hiperplano que maximiza los márgenes de separación de las dos clases en el espacio de características.

3) *Método basado en mínimos cuadrados,* como por ejemplo las redes RBF, cuyos nodos ocultos tienen una función de activación $g(x)$: $R \rightarrow R$, como la función Gaussiana:

$$g(x) = \exp(-x^2)$$

De esta forma:

$$G(\boldsymbol{a_i}, b_i, \boldsymbol{x}) = g(b_i \|\boldsymbol{x} - \boldsymbol{a_i}\|), \qquad b_i \in R^+$$

donde $\boldsymbol{a_i}$ y b_i son el centro y factor de impacto del nodo oculto i. R^+ indica el conjunto de todos los números reales positivos. La red RBF es un caso especial de las redes SLFN con nodos RBF en su capa oculta (Figura 3.2). Cada nodo tiene sus propios parámetros $\boldsymbol{a_i}$ y b_i, y su salida viene dada por una función simétrica en función de la distancia entre la entrada y el centro.

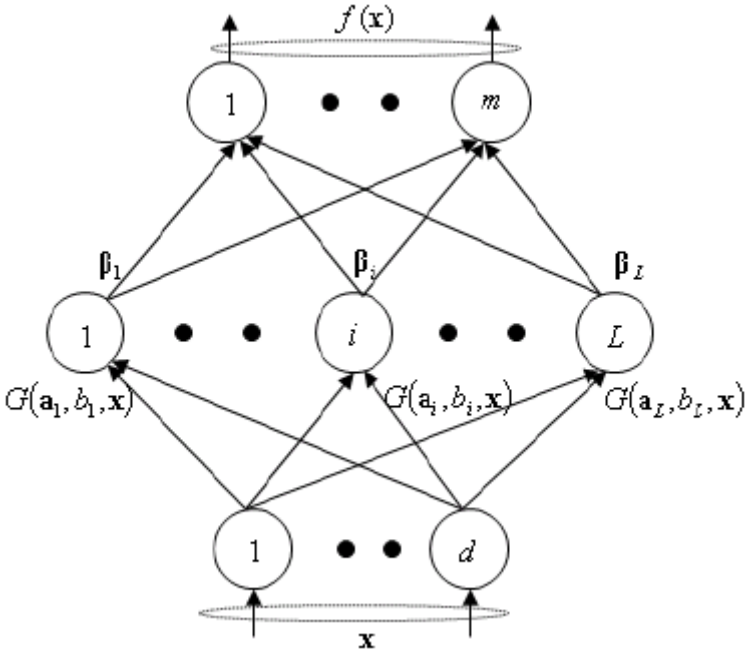


Figura 3.2: Red de tipo feedforward monocapa.

En la implementación de estas redes, según Lowe [22], los centros de los nodos ocultos pueden ser seleccionados aleatoriamente de los datos de entrada o de la región de entrenamiento en lugar de ser entrenados, y todos los factores de impacto b_i se establecen con el mismo valor. Después de fijar estos parámetros, el vector de pesos de salida $\boldsymbol{\beta}_i$ es el único parámetro desconocido que se puede resolver a través del método de mínimos cuadrados.

Las máquinas de aprendizaje extremo fueron desarrolladas para redes hacia delante de una sola capa oculta y después se extendieron a redes "generalizadas" hacia delante monocapa (SLFNs). La esencia del ELM es que los pesos de la capa oculta no necesitan ser calculados, es decir, dicha capa no necesita ser entrenada. Una de las implementaciones típicas del ELM consiste en usar nodos calculados aleatoriamente en la capa oculta, que puede ser independiente de los datos de entrenamiento. Acorde con la teoría de redes neuronales, en redes con propagación hacia adelante se llega al menor error de entrenamiento cuanto menor sea la norma de los pesos de salida, y la red tiende a tener mayor generalización. Por tanto los nodos ocultos no necesitan ser entrenados y los parámetros de la capa oculta pueden fijarse para después calcular los pesos de salida mediante el método de mínimos cuadrados.

3.2. TEORÍA DE APRENDIZAJE DE LAS MÁQUINAS DE APRENDIZAJE EXTREMO

Es importante mencionar que las capacidades de interpolación y de aproximación universal de las Máquinas de Aprendizaje Extremo han sido principalmente investigadas por Huang en [11-23].

La función de salida de una red SLFN con L nodos ocultos puede ser representada por:

$$f_L(x) = \sum_{i=1}^{L} \boldsymbol{\beta_i} g_i(x) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a_i}, b_i, \boldsymbol{x}), \boldsymbol{x} \in R^d, \boldsymbol{\beta_i} \in R^m$$

donde g_i denota la función de salida $G(\boldsymbol{a_i}, b_i, \boldsymbol{x})$ del nodo oculto i. Para nodos aditivos con función de activación g, $g_i$es definida como:

$$g_i = G(\boldsymbol{a_i}, b_i, \boldsymbol{x}) = g(\boldsymbol{a_i} \cdot \boldsymbol{x} + b_i), \boldsymbol{a_i} \in R^d, b_i \in R$$

Y para nodos RBF con función de activación g, $g_i$es definida como:

$$g_i = G(\boldsymbol{a_i}, b_i, \boldsymbol{x}) = g(b_i \|\boldsymbol{x} - \boldsymbol{a_i}\|), \boldsymbol{a_i} \in R^d, b_i \in R^+$$

En las últimas dos décadas, la capacidad de interpolación y aproximación universal de las redes SLFN han sido investigadas profundamente. Se probó que N muestras distintas pueden ser aprendidas por este tipo de redes con exactamente N nodos ocultos. Además, este estudio dio una respuesta más completa sobre la capacidad de interpolación de las SLFNs y probó que una red con, como mucho, N nodos ocultos y con una función de activación arbitraria no lineal y acotada puede aprender cualquier conjunto de N muestras distintas sin error. Tales funciones de activación incluyen la escalón, rampa y la sigmoidal así como las de base radial, funciones coseno cuadrado y muchas no regulares [20].

Muchos investigadores han demostrado rigurosamente en la teoría que dada una función de activación $g(x)$ que satisface ciertas condiciones, existe una secuencia de funciones de red $\{f_L\}$ que aproximan cualquier función continua f dada como objetivo con un error esperado $\epsilon > 0$. En todas estas teorías convencionales, todos los parámetros de cualquier f_L de la secuencia de red (por ejemplo los parámetros de la capa oculta $(\boldsymbol{a_i}, b_i)$ y los pesos de salida $\boldsymbol{\beta_i}$ necesitan ser ajustados libremente. De acuerdo con estas teorías convencionales, los parámetros $(\boldsymbol{a_i}, b_i)$ necesitan ser entrenados para encontrar los valores apropiados para la función objetivo f.

En las últimas dos décadas, se ha tratado de minimizar el esfuerzo realizado en ajustar los parámetros mencionados. En lugar de ajustar todos los parámetros de la capa oculta en toda la secuencia de red f_L, algunos investigadores sugirieron métodos en los que se añadían nodos ocultos y entonces se fijaban sus parámetros después de un entrenamiento. Los parámetros de los nodos existentes permanecerán fijos y nunca se modificarán en los demás procedimientos.

Los parámetros de la capa oculta en aquellos modelos convencionales de aprendizaje necesitan ser ajustados al menos una vez basándose en las muestras de entrenamiento. En contraste, todos los parámetros en las ELM no necesitan ser entrenados y pueden ser independientes de las muestras de entrenamiento. Una de las implementaciones típicas de las máquinas de aprendizaje extremo es que los parámetros de los nodos ocultos $(\boldsymbol{a_i}, b_i)$ se pueden generar de una forma aleatoria. La capacidad de aprendizaje de estas máquinas ha sido estudiada desde dos puntos de vista: capacidad de interpolación y capacidad de aproximación universal.

3.2.1. Teorema de Interpolación

Para N muestras distintas $(\boldsymbol{x_i}, \boldsymbol{t_i}) \in R^d \times R^m$, las redes monocapa de tipo "feedforward" con L nodos ocultos son matemáticamente modeladas como:

$$\sum_{i=1}^{L} \boldsymbol{\beta_i} g_i(\boldsymbol{x_j}) = \sum_{i=1}^{L} \boldsymbol{\beta_i} G(\boldsymbol{a_i}, b_i, \boldsymbol{x_j}) = \boldsymbol{o_j}, j = 1, \dots, N$$

Estas SLFNs pueden aproximar estas N muestras con un error medio nulo $\sum_{j=1}^{L} \|\boldsymbol{o_j} - \boldsymbol{t_j}\| = 0$, es decir, existen $(\boldsymbol{a_i}, b_i)$ y $\boldsymbol{\beta_i}$ tal que:

$$\sum_{i=1}^{L} \boldsymbol{\beta_i} G(\boldsymbol{a_i}, b_i, \boldsymbol{x_j}) = \boldsymbol{t_j}, j = 1, \dots, N$$

Las N ecuaciones anteriores pueden ser escritas de forma vectorial como:

$$\boldsymbol{H\beta} = \boldsymbol{T}$$

Donde,

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(x_1) \\ \vdots \\ \boldsymbol{h}(x_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a_1}, b_1, \boldsymbol{x_1}) & \cdots & G(\boldsymbol{a_L}, b_L, \boldsymbol{x_1}) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a_1}, b_1, \boldsymbol{x_N}) & \cdots & G(\boldsymbol{a_L}, b_L, \boldsymbol{x_N}) \end{bmatrix}_{\boldsymbol{N \times L}}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{y} \quad \boldsymbol{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N \times m}$$

$\boldsymbol{H}$ es la matriz de salida de la capa oculta de la red donde la columna i de la misma, es la salida del nodo oculto i con respecto a las entradas $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$. $\boldsymbol{h}(\boldsymbol{x})$ es el mapeo de características de la capa oculta. La fila i de $\boldsymbol{H}$ es el mapeo de características de la capa oculta con respecto de la entrada $\boldsymbol{x}_i$: $\boldsymbol{h}(\boldsymbol{x_i})$.

Se ha probado que desde un punto de vista de capacidad de interpolación, si la función de activación g es infinitamente diferenciable en cualquier intervalo, los parámetros de la capa oculta pueden ser generados aleatoriamente.

Teorema 3.2.1. *Dado un valor positivo pequeño $\epsilon > 0$, una función de activación $g: R \to R$ que sea infinitamente diferenciable en cualquier intervalo, y N muestras arbitrarias y distintas*

$(\boldsymbol{x_i}, \boldsymbol{t_i}) \in R^d \times R^m$, *existe* $L \leq N$ *tal que para cada* $\{\boldsymbol{a_i}, b_i\}_{i=1}^{L}$ *generados aleatoriamente de un intervalo de* $R^d \times R$, *de acuerdo a una función de distribución continua con probabilidad uno,* $\|\boldsymbol{H}_{N\times L}\boldsymbol{\beta}_{L\times m} - \boldsymbol{T}_{N\times m}\| < \epsilon$.

Desde un punto de vista de interpolación, el máximo número de nodos necesario no debe ser mayor que el número de muestras de entrenamiento. De hecho, si $L = N$, el error de entrenamiento puede ser cero.

Teorema 3.2.2. *Dada una función de activación* $g: R \to R$ infinitamente diferenciable en cualquier intervalo y *N muestras arbitrarias distintas* $(\boldsymbol{x_i}, \boldsymbol{t_i}) \in R^d \times R^m$, *para cualquier* $\{\boldsymbol{a_i}, b_i\}_{i=1}^{N}$ *generados aleatoriamente de un intervalo* $R^d \times R$, *de acuerdo a una función de distribución continua, con probabilidad uno,* $\|\boldsymbol{H}_{N\times N}\boldsymbol{\beta}_{N\times m} - \boldsymbol{T}_{N\times m}\| = 0$.

Desde un punto de vista de interpolación, se han usado un amplio rango de funciones en las ELM, lo que incluye las funciones sigmoidales, las funciones de base radial, seno, coseno, exponencial, y muchas otras no regulares.

Puede ser demasiado estricto exigir que las funciones de activación de los nodos ocultos sean infinitamente diferenciables. Por ejemplo, con esto no se incluirían algunas funciones importantes como la función escalón. Las redes con estas funciones son muy populares en las aplicaciones reales, especialmente en la implementación con hardware digital. Sin embargo, como la función escalón no es diferenciable, los investigadores no han intentado encontrar algoritmos eficientes directos para este tipo de redes. El teorema de interpolación mencionado arriba, referente a la capacidad universal de aproximación, puede ser extendido a cualquier tipo de función de activación a trozos incluyendo la función escalón, y así se puede aplicar un algoritmo de aprendizaje eficiente directo a aquellos casos que no pueden ser manejados por otras técnicas de aprendizaje pasadas.

3.2.2. Teorema de Aproximación Universal

Huang, en [24] demostró en la teoría que las redes SLFNs con nodos RBF o aditivos generados aleatoriamente, pueden aproximar cualquier función continua objetivo sobre cualquier subconjunto $X \in R^d$.

Se define $L^2(X)$ como un espacio de funciones f en un subconjunto compacto X en el espacio Euclídeo d-dimensional tal que $|f|^2$ son integrables, es decir, $\int_X |f(x)|^2 dx < \infty$. Se denota $L^2(R^d)$ como L^2. Para $u, v \in L^2(X)$, el producto escalar se define como:

$$\langle u, v \rangle = \int_X u(x)v(x)dx$$

El módulo de $L^2(X)$ en el espacio es denotado por $\|\cdot\|$, y la distancia más corta entre la función de red f_L y la función objetivo f se mide a través de:

$$\|f_L - f\| = \left[\int_X |f_L(x) - f(x)|^2 dx\right]^{1/2}$$

Definición 3.2.1. *Una función $g(x)$: $R \to R$ se dice que es continua a trozos si tiene sólo un número finito de discontinuidades en cualquier intervalo, y sus límites por la derecha e izquierda son definidos (no necesariamente igual) en cada discontinuidad.*

Definición 3.2.2. *Un nodo es llamado nodo aleatorio si sus parámetros $(\boldsymbol{a}, b)$ son generados aleatoriamente basándose en una función distribución de probabilidad continua.*

A diferencia de la aleatoriedad mencionada en otros métodos de aprendizaje, todos los parámetros de los nodos ocultos $(\boldsymbol{a_i}, b_i)$ en las máquinas de aprendizaje extremo pueden ser independientes de las muestras de entrenamiento y pueden ser generados antes de observarlas.

Definición 3.2.3. *La secuencia de la función $g_i = G(\boldsymbol{a_i}, b_i, \boldsymbol{x})$ se dice que es aleatoria si los parámetros correspondientes $(\boldsymbol{a_i}, b_i)$ son generados aleatoriamente de $R^d \times R$ o $R^d \times R^+$ basándose en una función de distribución de probabilidad continua.*

Lema 3.2.1. *Dada g: $R \to R$, $span\{g(\boldsymbol{a} \cdot \boldsymbol{x} + b) : (\boldsymbol{a}, b) \in R^d \times R\}$ es denso en L^p para todo $p \in [1, \infty)$ si y sólo si no es un polinomio (casi siempre).*

Lema 3.2.2. *Se define $k : R^d \to R$ como una función acotada integrable tal que k es continua (casi siempre) y $\int_{R^d} k(x)dx \neq 0$. Entonces, $span\left\{k\left(\frac{\boldsymbol{x}-\boldsymbol{a}}{\boldsymbol{b}}\right) : (\boldsymbol{a}, b) \in R^d \times R^+\right\}$ es denso en L^p para todo $p \in [1, \infty)$.*

Los lemas 3.2.1 y 3.2.2 muestran que las redes neuronales hacia delante con nodos aditivos o de tipo RBF pueden aproximar cualquier función objetivo que sea continua y que los parámetros de los nodos son entrenados y fijados a un valor adecuado. Estos lemas sólo muestran la capacidad universal de aproximación de estas redes, sin embargo, cómo encontrar los valores adecuados de los parámetros mencionados es una cuestión que permanece abierta, y para ello se han sugerido muchos algoritmos de aprendizaje.

Huang, demostró en [24] que dada una función de activación continua a trozos no constante y acotada g: $R \to R$ para nodos aditivos, o una función continua integrable a trozos g: $R \to R$ para nodos del tipo RBF, la capa oculta de la red SLFN no necesita ser entrenada, de hecho, todos los nodos ocultos pueden establecerse aleatoriamente. Las redes SLFN con nodos aleatorios puede aproximar cualquier función objetivo.

Se define $e_L \equiv f - f_L$ como la función de error residual para la actual f_L con *L* nodos ocultos, donde $f \in L^2(X)$ es la función objetivo. La capa de salida puede tener más de un nodo, $m > 1$, es decir, la función f es una función con múltiples salidas: $f = \left[f^{(1)}, \dots, f^{(m)}\right]^T$. La función de salida correspondiente de la red con L nodos ocultos es $f_L = \left[{f_L}^{(1)}, \dots, {f_L}^{(m)}\right]^T$. Se define $\beta_L^{(j)}$ como el peso de salida entre el nodo oculto L y el nodo de salida j, y ${e_L}^{(j)} \equiv f^{(j)} - {f_L}^{(j)}$ como la función de error residual del nodo de salida j con respecto a los L nodos ocultos, $j = 1, \dots, m$. Teóricamente:

Teorema 3.2.3. *Dada una función continua a trozos no constante y acotada g: $R \to R$ para nodos aditivos o una función continua integrable a trozos g: $R \to R$ y $\int_R g(x)dx \neq 0$ para*

nodos de tipo RBF, para una función objetivo continua f y una secuencia de la función generada aleatoriamente $\{g_L\}$, el límite $lim_{L\to\infty}\|f - f_L\| = 0$ se mantiene con probabilidad uno si

$$\beta_L^{(j)} = \frac{\langle e_{L-1}^{(j)}, g_L \rangle}{\|g_L\|^2}, j = 1, \dots, m.$$

El Teorema 3.2.3 puede ser extendido más allá de nodos aditivos o RBF para generalizar sobre redes SLFN. Dado un tipo de nodos ocultos de cómputo a trozos (posiblemente no como los nodos neuronales), si las redes SLFN pueden trabajar como aproximadores universales con parámetros ocultos ajustables, desde un punto de vista de aproximación de funciones, los parámetros de los nodos ocultos de estas redes "generalizadas" SLFN pueden en realidad ser ajustadas aleatoriamente a partir de una distribución continua. En teoría, los parámetros de estas redes pueden ser analíticamente determinados a través de ELM en lugar de ser entrenados. El entrenamiento, en realidad, no es necesario en estas redes "generalizadas" que incluyen redes sigmoidales, redes RBF, redes trigonométricas, redes umbral (función escalón), redes complejas, de orden superior, etc.

Teorema 3.2.4. *Dada una función continua a trozos no constante g: $R \to R$,* si abarcar *$\{G(\boldsymbol{a}, b, \boldsymbol{x}) : (\boldsymbol{a}, b) \in R^d \times R\}$ es engorroso en L^2, para una función objetivo continua f y una secuencia de función $\{g_L(\boldsymbol{x}) = G(\boldsymbol{a_L}, b_L, \boldsymbol{x})\}$ generada aleatoriamente en base a una distribución continua, $lim_{L\to\infty}\|f - f_L\| = 0$ se mantiene con probabilidad unidad si los pesos de salida $\boldsymbol{\beta}_i$ son determinados por mínimos cuadrados para minimizar $\left\|f(x) - \sum_{i=1}^{L} \beta_i g_i(x)\right\|$.*

El Teorema 3.2.4 supone que ELMs con arquitecturas de red fijas donde los parámetros de salida son determinados a través de mínimos cuadrados, pueden trabajar como aproximadores universales si la función de activación g es no continua a trozos y abarcar $\{G(\boldsymbol{a}, b, \boldsymbol{x}) : (\boldsymbol{a}, b) \in R^d \times R\}$ *es engorroso en L^2*.

3.3. MÁQUINA DE APRENDIZAJE EXTREMO (ELM)

La esencia de la máquina de aprendizaje extremo reside en:

1) La capa oculta de la red no necesita ser calculada iterativamente.
2) Acorde con la teoría sobre redes hacia delante, el error de entrenamiento $\|H\beta - T\|$ y la norma de los pesos $\|\beta\|$ necesitan ser minimizados.
3) La capa oculta necesita satisfacer la condición universal de aproximación (Teoremas 3.2.3 y 3.2.4).

Acorde con los Teoremas 3.2.1 y 3.2.4 los pesos de los nodos ocultos pueden ser generados aleatoriamente, de tal forma que los únicos parámetros desconocidos serán los vectores de pesos de salida $\boldsymbol{\beta}_i$ entre la capa oculta y la de salida, que pueden ser calculados directamente mediante mínimos cuadrados.

3.3.1. ELM básica

Los parámetros de los nodos $(\boldsymbol{a_i}, b_i)$ permanecen fijos una vez que se han establecido aleatoriamente. Entrenar la red SLFN simplemente es equivalente a encontrar la solución por mínimos cuadrados del sistema lineal $\boldsymbol{H\beta} = \boldsymbol{T}$:

$$\left\|\boldsymbol{H}\widehat{\boldsymbol{\beta}} - \boldsymbol{T}\right\| = \min_{\beta} \|\boldsymbol{H\beta} - \boldsymbol{T}\|$$

Si el número L de nodos ocultos es igual al número de muestras de entrenamiento N, $L = N$, acorde con el Teorema 3.2.1, la matriz $\boldsymbol{H}$ es cuadrada e invertible cuando los parámetros $(\boldsymbol{a_i}, b_i)$ son elegidos aleatoriamente, y así las redes SLFN pueden aproximar dichas muestras de entrenamiento con un error nulo.

Sin embargo, en la mayoría de los casos el número de nodos es mucho menor que el número de muestras de entrenamiento, $L \ll N$, por lo que no $\boldsymbol{H}$ es cuadrada y pueden no existir $\boldsymbol{a_i}, b_i, \boldsymbol{\beta}_i$ $(\mathrm{i} = 1, \ldots, \mathrm{L})$ tal que $\boldsymbol{H\beta} = \boldsymbol{T}$. La solución que minimiza por mínimos cuadrados el sistema de ecuaciones superior es:

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}$$

Donde $\boldsymbol{H}^{\dagger}$ es la *inversa generalizada Moore-Penrose* de la matriz $\boldsymbol{H}$.

Así, el ELM puede resumirse como:

Algoritmo ELM: dado un conjunto de entrenamiento $\mathcal{N} = \{(\boldsymbol{x_i}, \boldsymbol{t_i}) |\ \boldsymbol{x_i} \in \boldsymbol{R}^d, \boldsymbol{t}_i \in \boldsymbol{R}^m, i = 1, \ldots, N\}$, una función de salida de los nodos ocultos $G(\boldsymbol{a_i}, b_i, \boldsymbol{x})$, y un número de nodos ocultos L,

- *Paso 1*: Generar de forma aleatoria los parámetros de los nodos ocultos $(\boldsymbol{a_i}, b_i)$.
- *Paso 2*: Calcular la matriz de salida de la capa oculta $\boldsymbol{H}$.
- *Paso 3*: Calcular los pesos de la capa de salida $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T}$$

El algoritmo ELM puede trabajar con muchos tipos de funciones de activación. Muchos algoritmos de aprendizaje no están preparados para trabajar con redes umbral directamente y en su lugar usan redes analógicas para aproximar las redes umbrales, tales como el método de descenso por gradiente. Sin embargo, una ventaja del ELM es que puede ser usado para entrenar este tipo de redes directamente sin necesidad de recurrir a otros algoritmos.

Se pueden usar diferentes métodos para obtener la inversa generalizada Moore-Penrose de una matriz: proyección ortogonal, método de ortogonalización, métodos iterativos, y descomposición en valores singulares (Apéndice B).

3.3.2. ELM basado en el mapeo aleatorio de la capa oculta

El método de proyección ortogonal puede ser eficientemente utilizado en el ELM: $\boldsymbol{H}^{\dagger} = \left(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\boldsymbol{T}}$ si $\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}$ es no singular o $\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\boldsymbol{T}}\left(\boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}}\right)^{-1}$ si $\boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}}$ es no singular. Acorde con la

teoría de la regresión, se sugirió la adición de un valor positivo $1/\lambda$ a la diagonal de $\boldsymbol{H^T H}$ o $\boldsymbol{HH^T}$ en el cálculo de los pesos de salida $\boldsymbol{\beta}$. La solución resultante es estable y tiende a una mayor generalización en el entrenamiento.

Es decir, para mejorar la generalización del ELM se tiene

$$\boldsymbol{\beta} = \boldsymbol{H^T}\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{HH^T}\right)^{-1}\boldsymbol{T}$$

y por tanto la función de salida es

$$f(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{H^T}\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{HH^T}\right)^{-1}\boldsymbol{T}$$

O se puede también calcular

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H^T H}\right)^{-1}\boldsymbol{H^T T}$$

Teniendo como función de salida del ELM:

$$f(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{x})\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H^T H}\right)^{-1}\boldsymbol{H^T T}$$

Huang muestra en [25] que las soluciones anteriores consiguen minimizar $\|\boldsymbol{H\beta} - \boldsymbol{T}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$, que es el objetivo esencial del ELM tal y como ya se ha mencionado.

Así, el algoritmo de la máquina de aprendizaje extremo puede ser reescrito como:

Algoritmo ELM: dado un conjunto de entrenamiento $\mathcal{N} = \{(\boldsymbol{x_i}, \boldsymbol{t_i}) |\ \boldsymbol{x_i} \in \boldsymbol{R}^d, \boldsymbol{t}_i \in \boldsymbol{R}^m, i = 1, \dots, N\}$, una función de salida de los nodos ocultos $G(\boldsymbol{a_i}, b_i, \boldsymbol{x})$, y un número de nodos ocultos L,

- *Paso 1*: Generar de forma aleatoria los parámetros de los nodos ocultos $(\boldsymbol{a_i}, b_i)$.
- *Paso 2*: Calcular la matriz de salida de la capa oculta $\boldsymbol{H}$.
- *Paso 3*: Calcular los pesos de la capa de salida $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \boldsymbol{H^T}\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{HH^T}\right)^{-1}\boldsymbol{T} \quad ; \quad \boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H^T H}\right)^{-1}\boldsymbol{H^T T}$$

En estas implementaciones, la condición del número de nodos ocultos puede ser suavizada, y no depende fuertemente del número de muestras de entrenamiento N. Esto ocurre para ambos casos $L < N$ o $L \geq N$, y marca la diferencia con el teorema de interpolación en el que era necesario que $L \leq N$ (Teorema 3.2.1), pero se mantiene el teorema de aproximación universal consistentemente (Teorema 3.2.4).

La Figura 3.3 muestras una clasificación para un caso binario. En este caso el número de nodos ocultos se mucho mayor que el de muestras de entrenamiento.

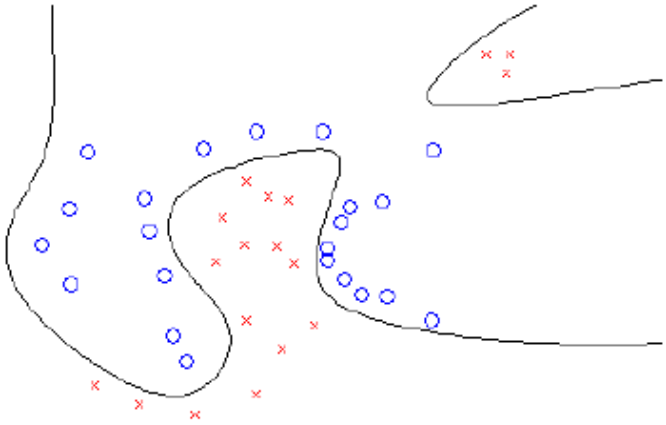


Figura 3.3: Frontera de clasificación para un caso binario.

Toh y Deng estudiaron en [26] las redes SLFNs de tipo sigmoidal aditivas. Deng y Man se centraron en obtener la solución analítica de $f(\boldsymbol{x})$ basándose en métodos de optimización. Toh propuso una tasa de error total basada en la solución del ELM para multi-clase (TER-ELM). Después Huang en [25] amplió el estudio del ELM para las redes SLFN generalizadas con diferentes tipos de nodos ocultos así como núcleos, y mostró que para casos de regresión, clasificación binaria y multi-clase se puede usar el algoritmo ELM estudiado.

3.4. CONJUNTOS ELM

La idea del conjunto de redes neuronales fue propuesta por Hansen y Salamon [27] en 1990. Su trabajo mostró que el entrenamiento de una red mono-capa puede ser mejorado usando un conjunto de redes neuronales con un esquema determinado. Esta técnica ha sido extendida en gran medida después de este descubrimiento.

Además, Sun propuso en [28] una integración de varias ELMs es un aplicación que consistía en predecir la cantidad de ventas en un futuro. Varias redes ELM fueron conectadas en paralelo y el promedio de las salidas fue lo que se usó para la predicción. El conjunto resultante obtenía una mayor capacidad de generalización.

Heeswijk investigó en [29] el modelo de conjuntos ELM adaptativo en la aplicación de "*predicción de un paso adelante en series de tiempo estacionario"*. Con esto verificó que el método trabajaba en series de tiempo estacionario y se probó la capacidad del método en series de tiempo no estacionario. Los estudios empíricos mostraron que el modelo del conjunto adaptativo consigue un error de prueba aceptable y buena adaptabilidad. Fue el mismo Heeswijk quien estudió el conjunto de ELM para aplicaciones de regresión a gran escala.

Además, los conjuntos de redes son métodos potencialmente importantes para entrenar en aprendizaje secuencial. El conjunto de red consiste en unas pocas redes que pueden tener diferentes capacidades de adaptación a los nuevos datos. Algunas pueden adaptarse mejor y más rápido que otras, lo que puede hacer que la red en conjunto supere el problema de que una sola no se adapte bien a los datos de entrada. Lan, en [30] propuso una estructura de red integrada, la cual llamó Conjunto de Máquinas de Aprendizaje Extremo Secuenciales en Línea

(EOS-ELM en inglés). El valor promedio de las salidas de cada ELM en el conjunto era usado como resultado final. Los resultados de la simulación probaron que la red EOS-ELM es más estable que una simple red ELM para cada uno de los problemas [20].

3.5. DISEÑO Y ENTRENAMIENTO DE UN MLP ENTRENADO CON ELM

Como ya se ha mencionado en el capítulo introductorio del trabajo, en este proyecto se pretende demostrar que con una nueva estructura multicapa que se introducirá más adelante, se consiguen mejorar tanto la capacidad de generalización como la de clasificación de otras redes, en este caso la Máquina de Aprendizaje Extremo.

Dicha estructura estará basada en el algoritmo ELM expuesto, por lo que lo primero que se hará es trabajar con redes SLFN (Figura 3.4) para comprobar cuáles son los resultados que se obtienen al entrenarlas con este algoritmo. Es decir, simplemente se pondrá en práctica toda la teoría expuesta a lo largo de este capítulo. Para ello, dejando a un lado el procesamiento de imágenes médicas que se verá al final del trabajo, como primera aplicación se trabajará con el conjunto de datos conocido MNIST.

Como se explica en el Apéndice A, estos datos, correspondientes a imágenes de números escritos a mano, estarán divididos en matrices de 28x28, por lo que se tienen unos datos con una dimensión de entrada de 784. El conjunto de entrenamiento estará formado por 60000 muestras, es decir:

$$\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_{60000}}\} \; con \; \boldsymbol{x_i} = \left(x^1, x^{\mathbf{2}}, \dots, x^{784}\right)$$

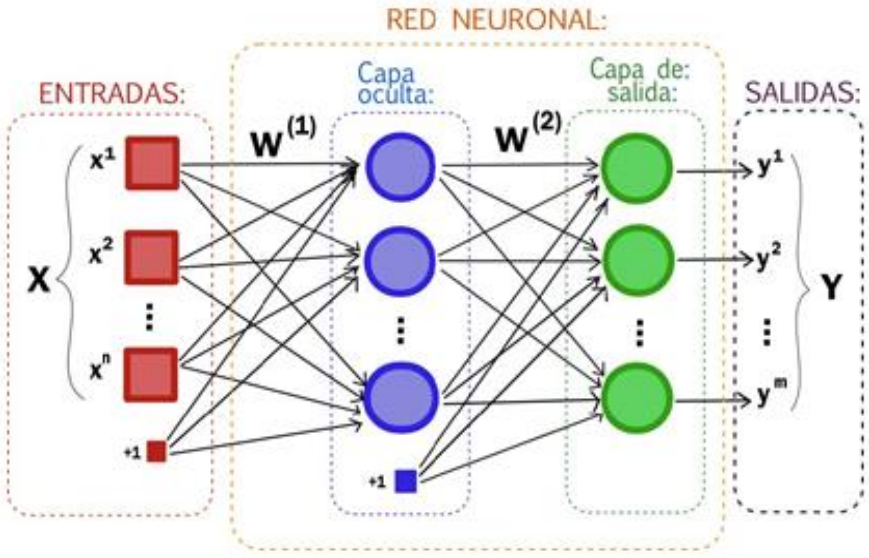


Figura 3.4: Esquema de red de trabajo.

Además los *targets* $\boldsymbol{t_i}$ se presentarán en codificación 1-de-10, donde la clase 1 será el valor $\boldsymbol{t_i} = (10 \dots 0)$ y la clase 10 será la correspondiente al valor $\boldsymbol{t_i} = (0 \dots 01)$. Por tanto:

$$\mathcal{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_{60000}}\} \; con \; \boldsymbol{y_i} = \left(y^1, y^2, \dots, y^{10}\right)$$

es decir, como es lógico las salidas tendrán la misma dimensión que los t*argets*.

Respecto al tipo de nodos de la capa oculta, se trabajará con nodos RBF, cuya función de activación será una Gaussiana con media 0 y varianza 1:

$$g(x) = \exp(-\frac{x^2}{2})$$

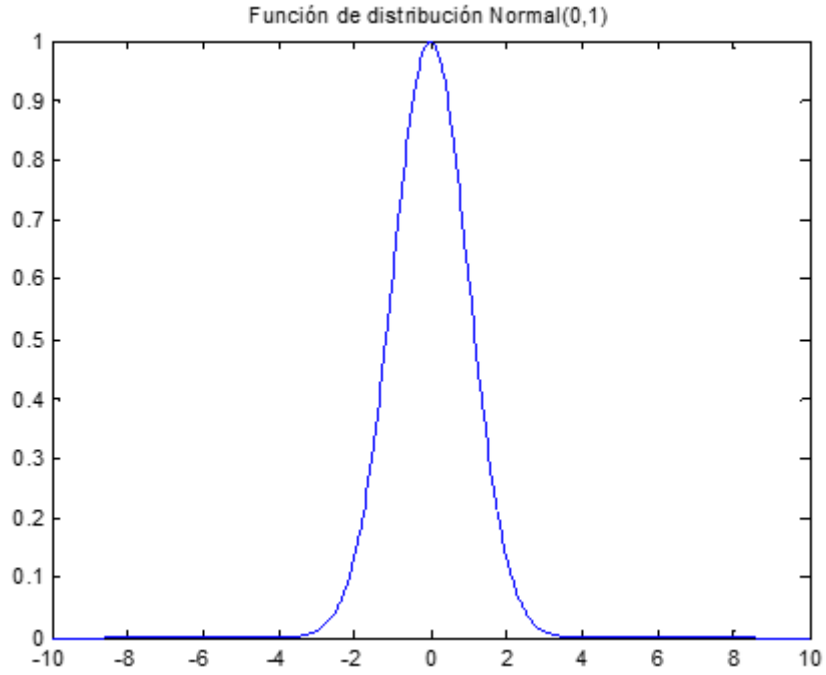


Figura 3.5: Función Gausiana media 0 y varianza 1.

Los principales pasos a realizar para el entrenamiento de la red serán los que se han explicado en el punto 3.3.2:

- *Paso 1*: Inicializar de forma aleatoria los parámetros de los nodos ocultos $(\boldsymbol{a_i}, b_i)$. Para ello se elegirán números aleatorios de una distribución Gaussiana como la de la Figura 3.5 usando el comando "*randn*" de MATLAB. Estos parámetros, por tanto, serán variables aleatorias. A su vez, cumplirán:

  $$\boldsymbol{a}^T\boldsymbol{a} = \boldsymbol{I} \quad \text{Y} \quad b^T b = 1$$

  En el siguiente capítulo se explicará el procedimiento con el que se consigue que cumplan estas dos condiciones.

- *Paso 2*: Calcular la matriz de salida de la capa oculta $\boldsymbol{H}$. Para ello no hay más que pasar los datos de entrada por la función de los nodos ocultos (Figura 3.5).

- *Paso 3*: Calcular los pesos de la capa de salida $\boldsymbol{\beta}$. En este caso se usa una de las fórmulas con las que, tal y como se ha explicado, se consigue mayor generalización y una solución más robusta:

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{T}$$

Llegado a este punto sólo falta resolver dos cuestiones: ¿cuántos nodos de la capa oculta se necesitan?, y ¿qué valor hay que darle al parámetro λ? Para ello se realizará validación cruzada, con el objetivo de evitar el O*verfitting* (sobreajuste).

El sobreajuste es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado (aprendizaje supervisado). El algoritmo de aprendizaje debe alcanzar un estado en el que será capaz de predecir el resultado en otros casos a partir de lo aprendido con los datos de entrenamiento. A esto se le llama capacidad de generalización. Generalizar es poder resolver situaciones distintas a las acaecidas durante el entrenamiento. Sin embargo, cuando un sistema se entrena demasiado (se sobreentrena) o se entrena con datos extraños, el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo (Figura 3.6, izquierda). Durante la fase de sobreajuste, el éxito al responder las muestras de entrenamiento sigue incrementándose mientras que su actuación con muestras nuevas va empeorando (Figura 3.6, derecha).

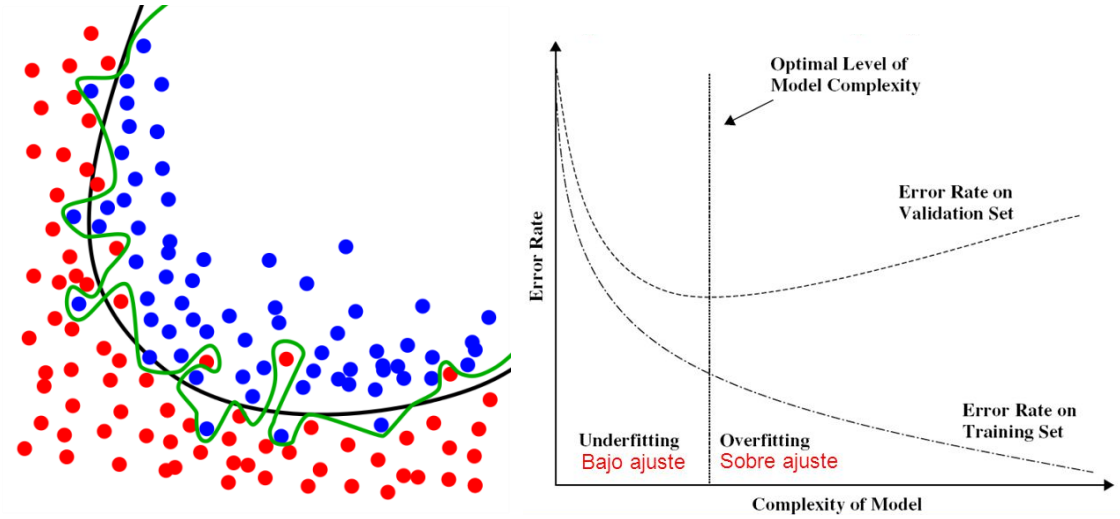


Figura 3.6: Frontera sobreajustada (izquierda). Evolución del error en el sobreajuste (derecha).

El procedimiento de validación cruzada que evitará el sobreajuste consistirá en dividir el conjunto de datos total (formado por 60000 muestras) en dos: un conjunto de entrenamiento más pequeño (36000 muestras, un 60% del total) y otro llamado conjunto de validación (24000 muestras, el 40% restante), asegurando que ambos conjuntos estén balanceados, es decir, haya aproximadamente el mismo porcentaje de muestras pertenecientes a cada una de las clases. Con el primer conjunto se entrenará y con el segundo se comprobará la capacidad de generalización de la red midiendo el error de clasificación cometido.

El siguiente paso es proporcionar un rango de valores para los parámetros λ y L y hacer un barrido sobre ellos buscando la pareja (L, λ) para la que la red obtiene mayor capacidad de generalización. En el caso de L, se le darán valores de entre 20 y 1020 nodos ocultos con intervalos de 100 nodos. Y en el caso de λ, el rango será de $(2^{-5}, 2^{-4}, \dots, 2^{4}, 2^{5})$. Por tanto, en el entrenamiento se usarán 11 valores diferentes de λ y 11 de L. Para cada valor de L se realizará el barrido sobre λ, resultando un total de 121 parejas de (L, λ).

Es importante aclarar que sería posible encontrar un valor mucho más preciso de los parámetros, pero para ello sería necesario hacer un barrido bastante más extenso y con intervalos más pequeños. Esto no fue posible debido a las limitaciones del ordenador utilizado, con tan sólo 4GB de memoria RAM.

Como ya se ha dicho, los pesos de los nodos ocultos $(\boldsymbol{a_i}, b_i)$ son variables aleatorias, y se sabe que para cada valor de los mismos se obtienen unos $\boldsymbol{\beta}$ diferentes. Por tanto, se decidió que el mejor procedimiento para buscar el valor de los pesos $(\boldsymbol{a_i}, b_i)$ con los que obtener una mayor capacidad de generalización de la red podía ser:

1) Para cada pareja (L, λ) se realizan 5 simulaciones diferentes. En cada una de ellas, se entrena la red (los tres pasos anteriores) con el conjunto de entrenamiento formado por 36000 muestras. Entonces, a través del conjunto de validación, se calcula su error de clasificación correspondiente. Con esto se obtendrá una matriz de errores de clasificación tridimensional como la de la Figura 3.7:

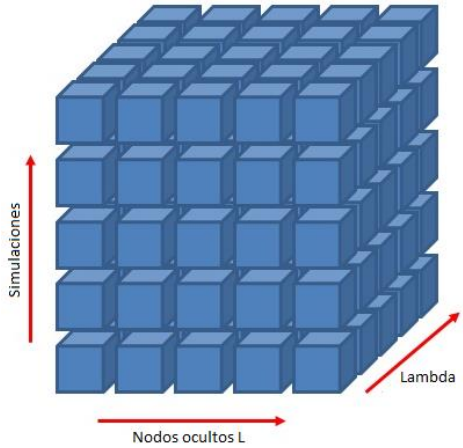


Figura 3.7: Matriz de errores tridimensional.

2) Se calcula la media de los errores cometidos en las 5 simulaciones realizadas para cada pareja (L, λ).
3) Se eligen los valores de (L, λ) para los que la media del error de clasificación sea menor.
4) Se elegirán los valores de $(\boldsymbol{a_i}, b_i)$ con los que se obtiene un error menor, de entre las cinco simulaciones con las que se ha obtenido el menor error medio.

En la imagen inferior se puede ver un diagrama de flujo que explica el entrenamiento total de la red ELM de forma detallada. Se puede ver de una forma más clara cómo en cada una de las 5 simulaciones realizadas se deben llevar a cabo los tres pasos principales expuestos con anterioridad, con el objetivo de obtener la matriz de errores tridimensional mencionada.

Una vez acabado este proceso, se habrá conseguido establecer los valores de todos los parámetros de la red obteniendo una buena capacidad de generalización y gran robustez.

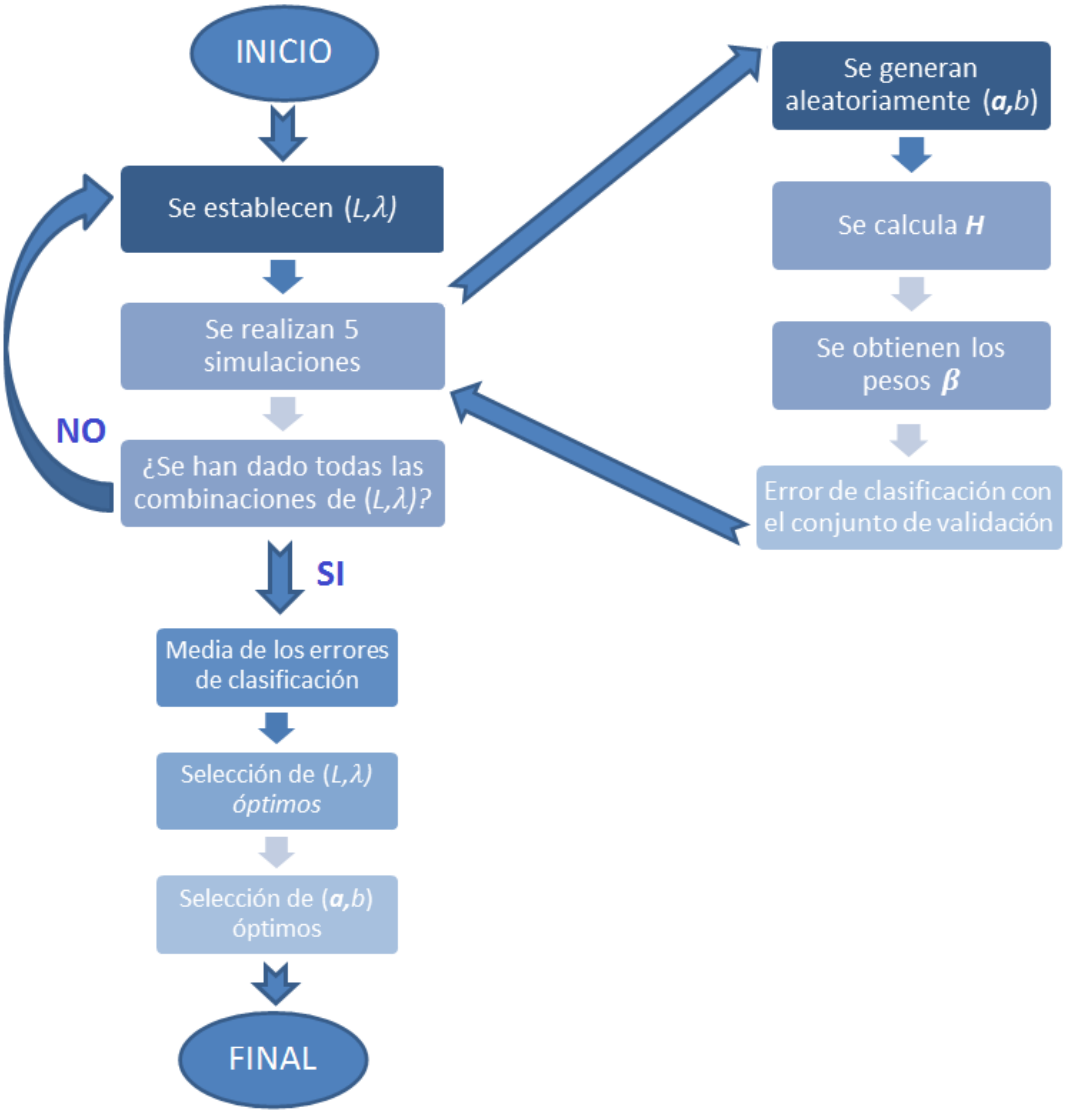


Figura 3.8: Diagrama de flujo del entrenamiento de ELM.

3.6. RESULTADOS

Como se ha comentado en el punto anterior, en primer lugar se realizó la etapa de entrenamiento. En la Figura 3.9 se presenta la evolución del error medio en función de los valores de los parámetros (L, λ).

Como se puede observar, el error medio disminuye conforme aumentan tanto L como λ. Esto lleva a pensar que si se hubiera tenido la opción de trabajar con un ordenador más potente y se hubiese realizado un barrido con rangos más amplios, hubiese sido posible encontrar un error medio menor que el que aquí se ha obtenido y, por tanto, una mayor precisión en el entrenamiento. Los valores óptimos en este caso son:

- Número de nodos de la capa oculta $L = 1020$.
- Parámetro $\lambda = 32$.

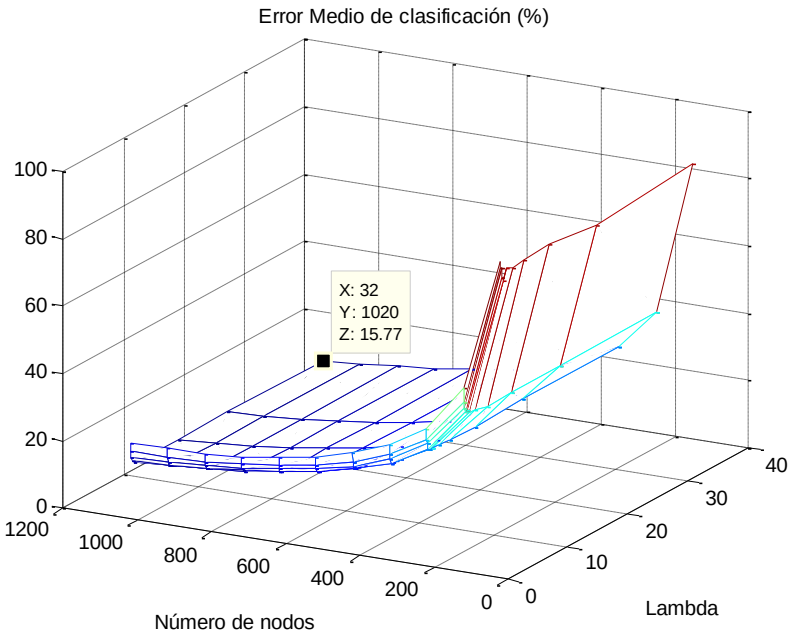


Figura 3.9: Evolución del error medio de clasificación ELM.

Para acabar, se realiza la última etapa de testeo, cuyo conjunto de datos está formado por 10000 muestras/patrones (base de datos MNIST). Para ello, simplemente se pasa el conjunto por la red y se mide la precisión obtenida a la salida. Esto se calcula mediante la umbralización de los datos de salida, es decir, tomando como un 0 los valores menores de 0.5, y un 1 los mayores de este umbral. Estos resultados umbralizados se compararán con las etiquetas de los datos para obtener el porcentaje de errores cometidos y, por tanto, la precisión de la red.

En la siguiente tabla se presentan los resultados obtenidos para diferentes valores de (L, λ) durante el entrenamiento. En ella se pueden apreciar diferentes resultados de la precisión de la red en función del número de nodos de la capa oculta, así como la influencia del valor del parámetro λ cuando el número de nodos es el mismo.

L	λ	Precisión (%)
20	0.0313	9.19
120	0.25	55.89
520	0.0313	74.54
520	32	79.84
820	1	83.60

Tabla 3.1: Resultados de entrenamiento.

Como ya se ha comentado, conforme aumenta L se crea una red más potente, y con el aumento de λ lo hace el nivel de regularización. Ambos casos provocan una mejoría del funcionamiento de la red.

L óptima	λ óptima	Máxima Precisión (%)
1020	32	85.10

Tabla 3.2: Resultado final.

CAPÍTULO 4.

APRENDIZAJE DE REPRESENTACIÓN CON MÁQUINAS DE APRENDIZAJE EXTREMO PARA GRANDES DATOS

4.1. INTRODUCCIÓN

La capacidad de generalización de un algoritmo de aprendizaje máquina depende, entre otras cosas, de las características del conjunto de datos con el que se trabaja. Por tanto es importante trabajar estas características para representar la estructura de los datos. Sin embargo, este proceso requiere un conocimiento del dominio así como el ingenio humano para poder generar las características apropiadas. Hinton demostró [31] que las Máquinas de Boltzman Restringidas (Restricted Boltzman Machines, RBM) y los Auto-Codificadores (Auto Encoders, AE) son una buena forma de trabajar estas características y a su vez se pueden usar para entrenar redes multicapa. A estas redes multicapa se les llama Redes Profundas, y existen dos tipos basados en las RBM a las que se les llama Redes de Creencia Profunda (Deep Belief Networks, DBN) y Máquinas de Boltzman Profundas (Deep Boltzman Network, DBM). Hay dos tipos de Auto-Codificadores basados en redes profundas, que son: Auto-Codificadores Apilados (Stacked Auto Encoders, SAE) y Auto-Codificadores Apilados de Eliminación de Ruido (Stacked Denoising Auto Encoders, SDAE). Las DBN y DBM son creadas mediante el apilamiento de RBMs, mientras que los SAE y SDAE se crean apilando AEs. Las redes profundas superan a las redes multicapa tradicionales, a las SLFNs y a las SVMs cuando se trata de trabajar con grandes datos, pero tienen el problema de que la velocidad de entrenamiento es lenta [32].

En estos últimos años, el concepto de las arquitecturas profundas ha despertado un creciente interés dentro de la comunidad de la inteligencia artificial y el aprendizaje automático. Estas arquitecturas profundas para modelado pueden considerarse como una tercera generación de

redes neuronales, las cuales se caracterizan por estar formadas por varias capas de neuronas ocultas que realizan una abstracción de la información.

Aunque estos modelos ya eran conocidos en la década de los 80, en la práctica se había visto que estas redes profundas no proporcionaban ventajas importantes sobre las habituales redes neuronales de una única capa oculta. Sin embargo, avances científicos recientes han demostrado que, si se utilizan algoritmos de construcción apropiados, las redes profundas sí son capaces de obtener niveles de rendimiento fuera del alcance de las redes clásicas.

En este capítulo se verá, en primer lugar, la estructura de los Auto-Codificadores (Auto Encoders, AE), cuyo funcionamiento se basa en obtener a la salida la información que se recibe a la entrada.

Las Máquinas de Aprendizaje Extremo Multicapa (Multi Layer Extreme Learning Machine, ML-ELM) con las que se trabajará después, son similares a las redes profundas de las que se ha hablado. Estas ML-ELM se forman apilando AEs y realizan aprendizaje no supervisado capa por capa.

4.2. AUTO-CODIFICADORES

Los Auto-Codificadores son un simple circuito de aprendizaje cuyo objetivo es transformar su entrada en una salida con la menor cantidad de distorsión posible. Aunque conceptualmente parece algo muy simple, tienen un papel muy importante en el aprendizaje máquina. Fueron tratados por primera vez en 1980 por Hinton y el grupo PDP para superar el problema de "backpropagation sin un profesor", usando el conjunto de datos de entrada como profesor. Estas estructuras satisfacen uno de los paradigmas fundamentales para el aprendizaje no supervisado y desde el inicio trataron el misterio de cómo los cambios sinápticos inducidos por eventos locales pueden ser coordinados de una manera auto-organizada para producir un comportamiento inteligente [12].

Más recientemente, los auto-codificadores han tomado un papel importante en las arquitecturas profundas, particularmente en las RBMs donde son combinados y entrenados de abajo hacia arriba de manera no supervisada, seguido por una fase de aprendizaje supervisado en la capa más alta de la red. Estas arquitecturas han sido usadas en problemas tanto de clasificación como de regresión.

Básicamente, el funcionamiento de un auto-codificador consiste en replicar a su salida la información que le llega a la entrada. De esta forma, como se puede apreciar en la arquitectura básica de este tipo de red (Figura 4.1), la capa de salida tiene el mismo tamaño que la de entrada. Esta idea a priori puede parecer simple, pero lo importante es que con esto se consigue que la red aprenda en su capa intermedia una representación de los datos de entrada sobre un espacio de diferente dimensión. El auto-codificador presenta dos grandes ventajas que hacen de él una de las redes neuronales más estudiadas en estos últimos años.

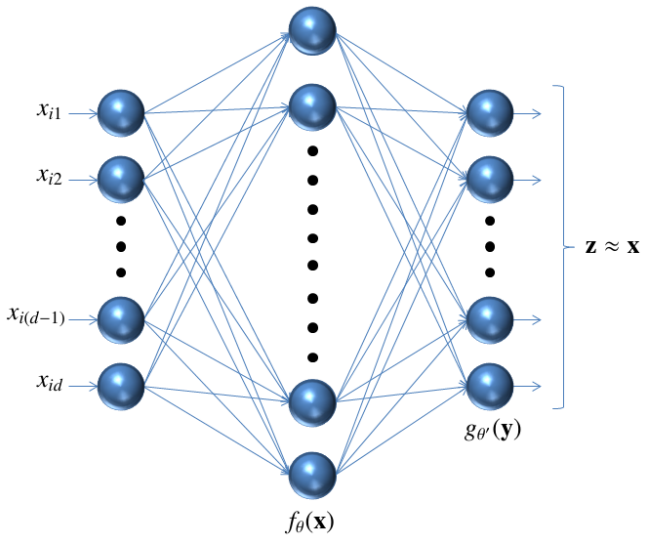


Figura 4.1: Auto-Codificador de imágenes.

Así, se pueden distinguir tres posibles situaciones en función del número de nodos ocultos:

- En primer lugar, si el tamaño de esta capa es menor que el de entrada, el auto-codificador es comúnmente denominado compresivo, ya que realiza una compresión de la información que le llega. Este procedimiento ha sido ampliamente estudiado y comparado con diferentes técnicas de reducción de dimensionalidad de los datos. En este trabajo se compara con una de las más conocidas Descomposición en Valores Singulares.
- Por otro lado, podría darse el caso de que la capa oculta tuviera el mismo número de nodos que las de entrada y salida. En este caso, se ha demostrado que la red tiende a aprender la matriz identidad y es algo que no aporta nada nuevo a la información que se tiene a priori.
- Finalmente, se destaca el caso en el que la red mapea el conjunto de datos de entrada a un espacio de dimensión mayor. En este caso, mediante el apilado de varios auto-codificadores (procedimiento desarrollado en este trabajo), se puede hacer más separable un problema de clasificación al tratarlo en otro espacio de dimensión.

Una vez visto, a grandes rasgos, el funcionamiento de este tipo de redes, es el momento de presentar una estructura general de la misma.

4.2.1. Estructura General de un Auto-Codificador

Una estructura general de este tipo de arquitecturas es la mostrada en la Figura 4.2. Un auto-codificador $n|p|n$ viene definido por: $n, p, m, \mathbb{F}, \mathbb{G}, \mathcal{A}, \mathcal{B}, \mathcal{X}, \Delta$ donde:

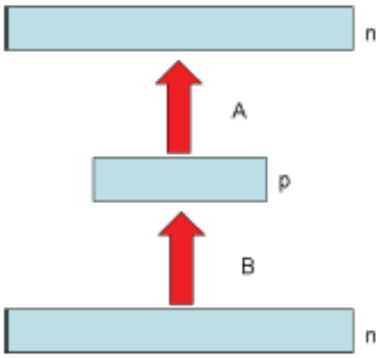


Figura 4.2: Estructura general de un Auto-Codificador.

1. $\mathbb{F}$ y $\mathbb{G}$ son conjuntos.
2. n y p son enteros positivos. En la imagen se ve el caso en que $0 < p < n$.
3. $\mathcal{A}$ es una clase de funciones de $\mathbb{G}^p$ a $\mathbb{F}^n$.
4. $\mathcal{B}$ es una clase de funciones de $\mathbb{F}^n$ a $\mathbb{G}^p$.
5. $\mathcal{X} = \{\boldsymbol{x_1}, \dots, \boldsymbol{x_m}\}$ es un conjunto de m vectores de entrenamiento en $\mathbb{F}^n$. Si existen Targets, se define $\mathcal{T} = \{\boldsymbol{t_1}, \dots, \boldsymbol{t_m}\}$ como el correspondiente conjunto de vectores en $\mathbb{F}^n$.
6. Δ es una función distorsión definida sobre $\mathbb{F}^n$.

Para cualquier $A \in \mathcal{A}$ y $B \in \mathcal{B}$, el auto-codificador transforma un vector de entrada $x \in \mathbb{F}^n$ en un vector de salida $A\ o\ B(x) \in \mathbb{F}^n$. El problema del auto-codificador es encontrar $A \in \mathcal{A}$ y $B \in \mathcal{B}$ que minimicen la función de distorsión:

$$\min E(A,B) = \min_{A,B} \sum_{t=1}^{m} E(x_t) = \min_{A,B} \sum_{t=1}^{m} \Delta(A\ o\ B(x_t), x_t)$$

En el caso donde se proporcionan los targets:

$$\min E(A,B) = \min_{A,B} \sum_{t=1}^{m} E(x_t, t_t) = \min_{A,B} \sum_{t=1}^{m} \Delta(A\ o\ B(x_t), t_t)$$

Obviamente, esta estructura tan general puede modificarse para muchos tipos de auto-codificadores, dependiendo de, por ejemplo, los conjuntos $\mathbb{F}$ y $\mathbb{G}$, las clases de funciones de $\mathcal{A}$ y $\mathcal{B}$, la función Δ o por la presencia de restricciones adicionales tales como la regularización. Cuando se comenzó a estudiar este tipo de redes, fue precisamente un caso especial de esta estructura, en el que todos los vectores pertenecían a $\mathbb{F} = \mathbb{G} = \mathbb{R}$ y tanto $\mathcal{A}$ como $\mathcal{B}$ eran matrices multiplicativas seguidas de una transformación sigmoidal no lineal (esta función es muy usada en problemas de regresión para la capa oculta). Éste es, precisamente, el caso que se usa en este proyecto para procesar las características de las imágenes. Actualmente ya está establecida y se ha estudiado la teoría de los auto-codificadores lineales con valores complejos, pero eso es algo en lo que no se va a entrar aquí [12].

4.2.2. Diseño y Entrenamiento de un Auto-Codificador

Los auto-codificadores con los que se trabaja en este proyecto se caracterizan por representar las características de los datos de entrada a su salida, basándose en los valores singulares de la matriz de entrada. Como ya se ha visto en el Capítulo 3, el ELM es un aproximador universal, y en este capítulo, se va a poder ver cómo la estructura del Auto-Codificador basada en el algoritmo ELM (ELM-AE) también lo es [32].

El método de Descomposición en Valores Singulares (Singular Value Descomposition, SVD) de una matriz se usa muy a menudo para el procesamiento de imágenes. Este método se basa en la factorización de dicha matriz en tres diferentes, con el objetivo de obtener una imagen lo más parecida posible a la original pero reduciendo su tamaño. Se puede ver de forma más detenida en qué consiste este procedimiento en el Apéndice B. En esta parte del trabajo se pretende comparar estas estructuras llamadas auto-codificadores con el método tradicional SVD a la hora de representar las características de un conjunto de píxeles.

¿Qué son los valores singulares de una matriz? Esta cuestión se puede resolver rápidamente sabiendo que cualquier matriz A de $m \times n$, la matriz $A^T A$ de $n \times n$ es simétrica y por tanto puede ser diagonalizada ortogonalmente. Los autovalores de $A^T A$ serán reales y no negativos [33-34]. Esto es sabido ya que para cualquier autovector u, se tiene que

$$A^T A u = \lambda u$$

Si se multiplica a ambos lados por u, se obtiene la igualdad

$$u^T A^T A u = \lambda u^T u$$

Que indica que $\|Au\|^2 = \lambda \|u\|^2$, lo que conlleva que $\lambda \geq 0$. Por tanto, tiene sentido tomar las $\sqrt{\lambda_i}$, si $\lambda_i, i = 1 \dots n$, son los autovalores de $A^T A$. En resumen:

Definición 4.2.1. *Si A es una matriz $m \times n$, los valores singulares de A son las raíces cuadradas de los autovalores de $A^T A$, y se denotan mediante $\sigma_1, \dots, \sigma_n$. Es una convención acomodar los valores singulares de modo que $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.*

Existen tres formas significativas de representar las características de entrada por parte de los auto-codificadores [32]:

- *Representación Comprimida.*

Se representan desde un espacio de los datos de entrada de mayor dimensión a un espacio dimensional menor.

- *Representación Dispersa.*

Se representan desde un espacio de los datos de entrada de menor dimensión a un espacio dimensional mayor.

- *Representación de igual dimensión.*

Como su nombre indica, la dimensión del espacio de los datos de entrada es la misma que la de los datos de salida.

De nuevo, dejando un poco de lado la aplicación final de segmentación de imágenes médicas, en este apartado se realizarán todos los experimentos con la base de datos MNIST (Apéndice A). Anteriormente, en la Figura 4.2, se ha mostrado la estructura general de un auto-codificador, pero es importante centrarse en la que será objeto de estudio. Ésta se puede ver en la Figura 4.3.

Tanto los pesos de la capa oculta como los sesgos o "bias" no sólo serán inicializados aleatoriamente, sino que después se modificarán de tal forma que se consiga que sean ortogonales entre sí, cumpliendo, como en el caso del ELM:

$$\boldsymbol{a}^T\boldsymbol{a} = \boldsymbol{I} \quad \text{Y} \quad b^T b = 1$$

Para ello, en primer lugar se iniciarán los valores de forma aleatoria de entre una distribución $N(0,1)$ y después se usará el método de ortogonalización de Gran Schmidt sobre $\boldsymbol{a}$ para obtener una base ortonormal. Con respecto a b, sólo es necesario normalizar.

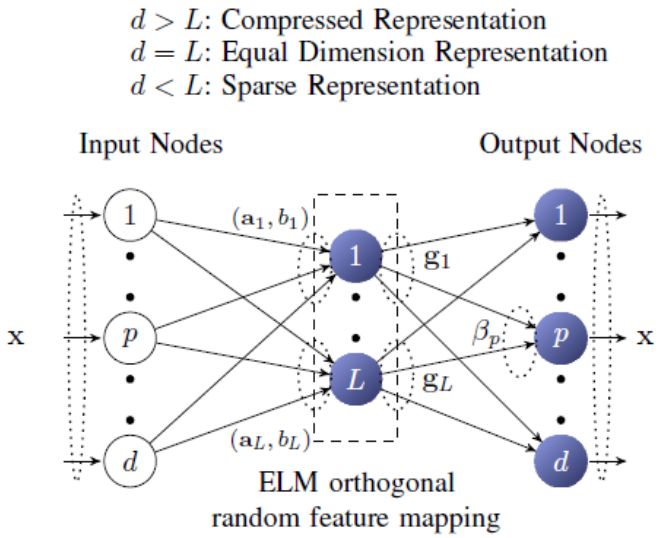


Figura 4.3: Estructura del Auto-Codificador de trabajo.

El método de Gram-Schmit es un algoritmo para obtener, a partir de una base, otra que genere el mismo espacio y además sea ortogonal. Matemáticamente es muy sencillo: si se tiene que a_k son cada uno de los vectores que forman la matriz $\boldsymbol{a}$, se pueden calcular nuevos vectores u_k que forma la base ortogonal de la forma:

$$u_k = a_k - \sum_{i=1}^{k-1} \frac{\langle u_j, a_k \rangle}{\langle u_j, u_j \rangle} u_j$$

A partir de las propiedades del producto escalar, es sencillo probar que el conjunto de vectores $u_1, \dots, u_n$ es ortogonal. Para hallar la base ortonormal basta con dividir entre la norma de cada vector de la base hallada:

$$e_k = \frac{u_k}{\|u_k\|} = \frac{u_k}{\sqrt{\langle u_k, u_k \rangle}}$$

De nuevo, para evitar sobreajuste (overfitting) se realizará el proceso de validación cruzada en el entrenamiento, dividiendo los datos en conjunto de entrenamiento (60% del total de patrones) y en conjunto de validación (40%). Una representación de los primeros 1000 patrones del conjunto de entrenamiento se pueden ver en la Figura 4.4.

La diferencia principal en el entrenamiento con respecto al caso del ELM será la elección de los valores óptimos de λ y L, que se hará tratando de minimizar la función de error *Mínimos Cuadrados Regularizados* (Regularized Least Squares, RLS) en lugar del error de clasificación como en el caso anterior. Esto es así porque no se trata de un problema de clasificación mediante aprendizaje supervisado, sino de un problema de regresión mediante aprendizaje no supervisado, por lo que en lugar de medir probabilidades de error se intenta minimizar una función de error.

- Mínimos cuadrados regularizados: $\|\boldsymbol{H\beta} - \boldsymbol{X}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$

Como se puede observar, en esta función de error aparece $\boldsymbol{\beta}$. El objetivo del entrenamiento es encontrar estos pesos de la capa de salida para los que mi red minimice dicha función de error. Se ha visto en el Apartado 3.3.2 que la fórmula que lo consigue es:

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H^T H}\right)^{-1} \boldsymbol{H^T X}$$

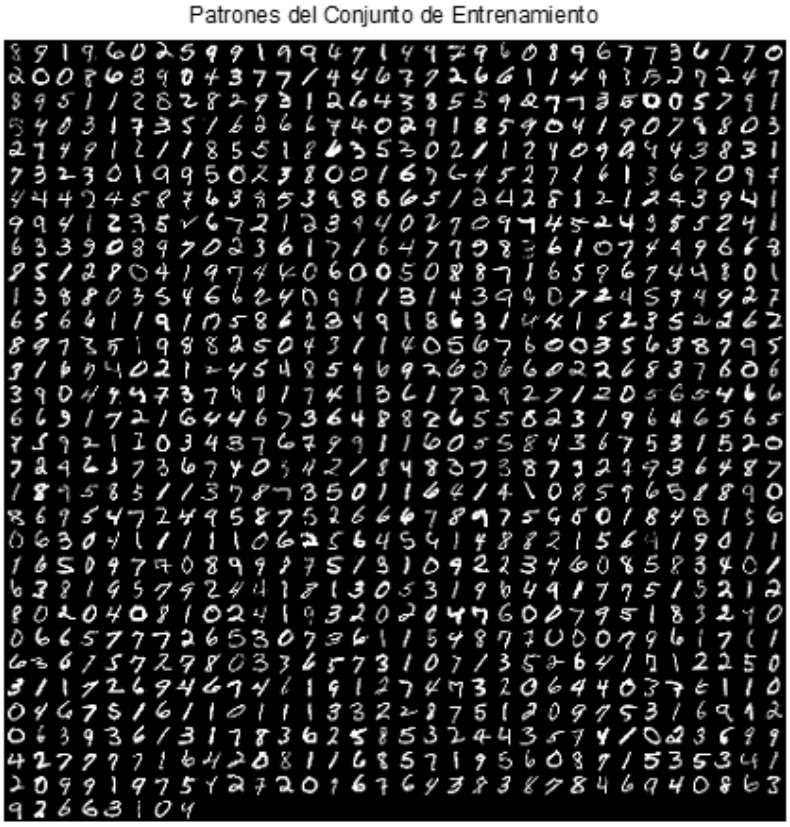


Figura 4.4: 1000 primeros patrones de entrenamiento.

De una forma resumida, el entrenamiento realizado:

1) Se realiza un barrido en λ $(2^{-5}, 2^{-4}, \dots, 2^{4}, 2^{5})$ y L $(20{:}\,100{:}\,1020)$. Para cada pareja de valores (L, λ) se ejecuta 5 veces lo siguiente:
   - Se generan $(\boldsymbol{a}, b)$.
   - Se calcula la matriz de salida de la capa oculta $\boldsymbol{H}$.
   - Se obtienen los pesos de la capa de salida $\boldsymbol{\beta}$.
   - Con el conjunto de validación se obtiene el error por Mínimos Cuadrados Regularizados.
2) Se eligen los valores óptimos de los parámetros (L, λ) que serán aquellos para los que se obtiene el mínimo error medio calculado a partir de la función anterior.
3) Se elegirán los valores de $(\boldsymbol{a}, b)$ con los que se obtiene un error menor, de entre las cinco simulaciones con las que se ha obtenido el menor error medio.

Por último, se debe aclarar que la función de activación de los nodos ocultos en este caso será una sigmoide:

$$g(x) = \frac{1}{1 + \exp(-x)}$$

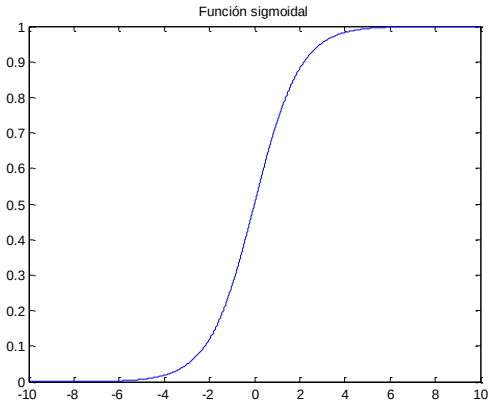


Figura 4.5: Función sigmoidal.

4.2.3. Resultados

Como en el caso relativo a las ELMs, lo primero que se hace es entrenar la red. Para ello se sigue el procedimiento descrito en el apartado anterior. Si se representa la evolución del error medio cometido en función de los valores de (L, λ), teniendo en cuenta la función de error anterior:

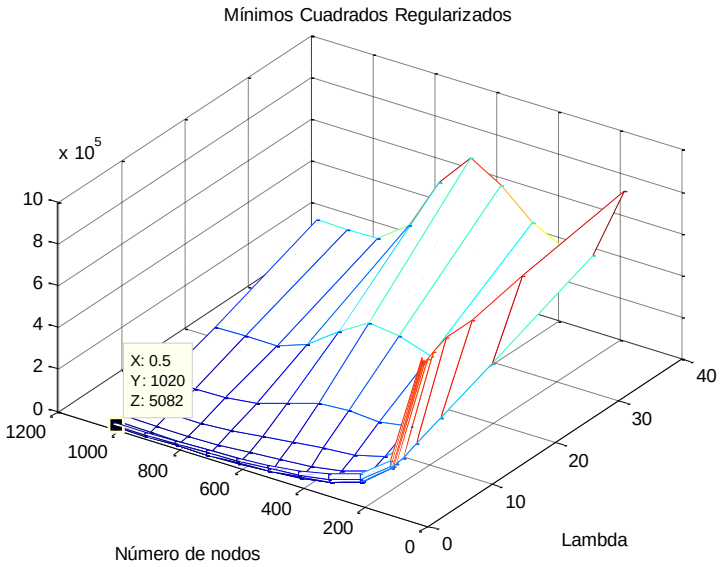


Figura 4.7: Error medio por Mínimos Cuadrados Regularizados.

Como se aprecia en las figuras superiores, se obtienen unos valores óptimos de:

- Nodos ocultos $L = 1020$.
- Parámetro $\lambda = 0.5$.

El siguiente paso es el testeo de la red. Para ello, se sigue el mismo procedimiento propuesto en el estudio [32]: crear un conjunto de test formado por 20 patrones pertenecientes a cada una de las clases. Como se tienen 10 clases, relativas a cada uno de los dígitos manuscritos (del 0 al 9), el conjunto de test estará formado por un total de 200 patrones. Dichos patrones son elegidos de entre el total de 1000 que forman el conjunto de test de la base de datos MNIST. A continuación se puede ver el conjunto comentado.

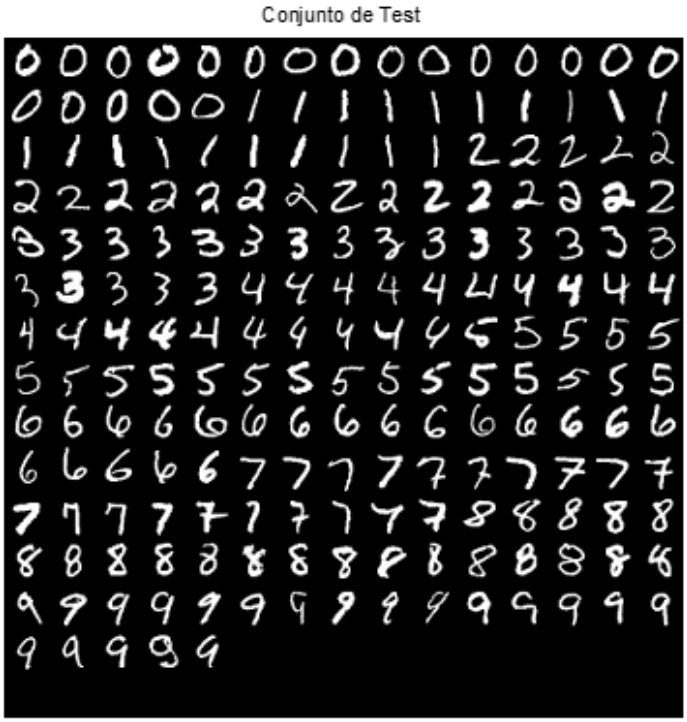


Figura 4.8: Conjunto de test.

Como se va a ver a continuación, el buen funcionamiento de un auto-codificador depende del número de nodos de la capa oculta L y del parámetro λ, mientras que la precisión del procedimiento SVD depende del número de valores singulares significativos que se tengan en cuenta. Por tanto, para comparar la capacidad de representación de ambos métodos, se procede a representar los resultados obtenidos por el auto-codificador para diferentes valores de (L, λ), para después ver lo que se obtiene con el método de Descomposición en Valores Singulares.

4.2.3.1. Auto-Codificador

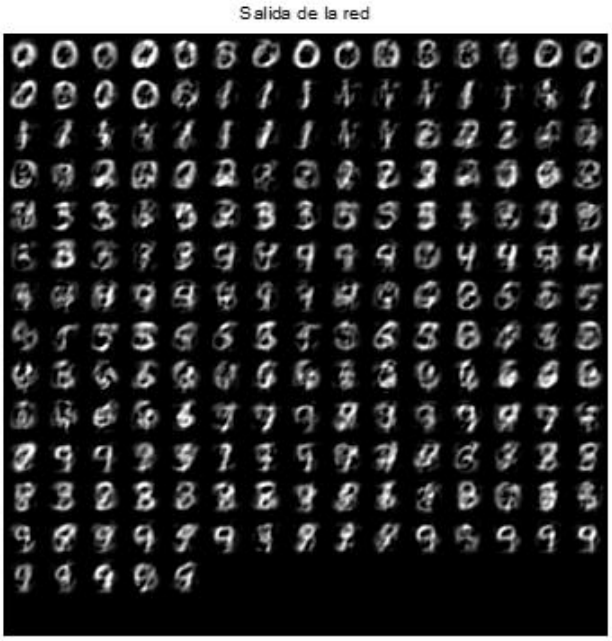


Figura 4.13: Solución para $L = 20$ y $\lambda = 1$.

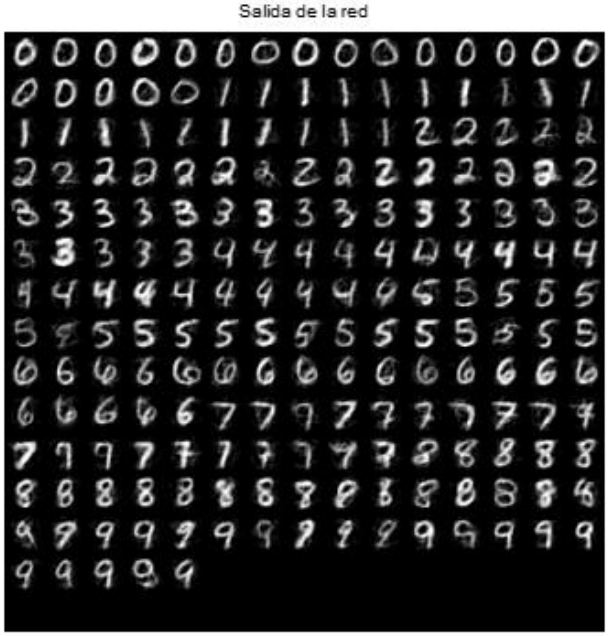


Figura 4.14: Solución para $L = 120$ y $\lambda = 1$.

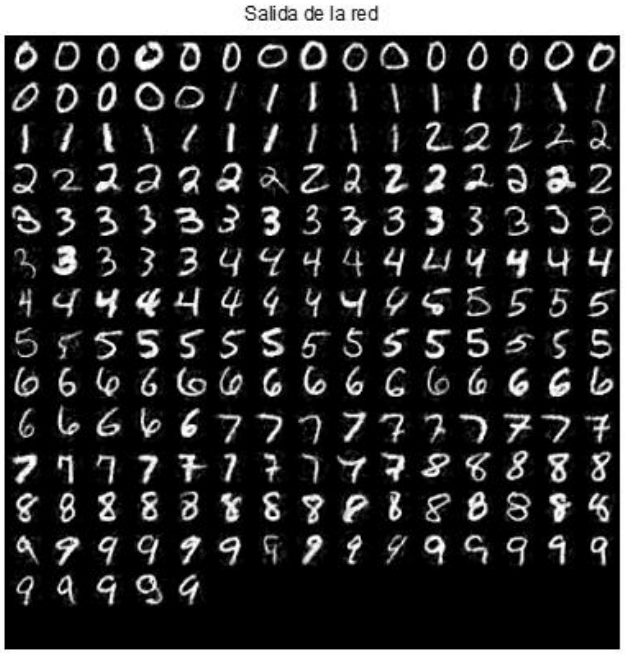


Figura 4.11: Solución para $L = 120$ y $\lambda = 16$.

Figura 4.12: Solución para los valores óptimos $L = 1020$ y $\lambda = 0.5$.

Con estos resultados se concluye que la principal ventaja de los auto-codificadores es que se puede conseguir una red con un número pequeño de nodos en su capa oculta que es capaz de representar a la salida las características de los datos de entrada con una precisión bastante buena, y a su vez posee una excelente capacidad de generalización. Se aprecia que para un

número de 120 nodos ocultos se consigue reducir en gran medida la dimensión de los datos de entrada y obtener una versión comprimida de los mismos de $200\ x\ 120 = 24000$ datos.

Por otro lado, si se eligen los valores para los que el error cometido es menor se puede decir que, a simple vista, se consigue reproducir a la salida la imagen que se introduce por la red.

4.2.3.2. Descomposición en Valores Singulares

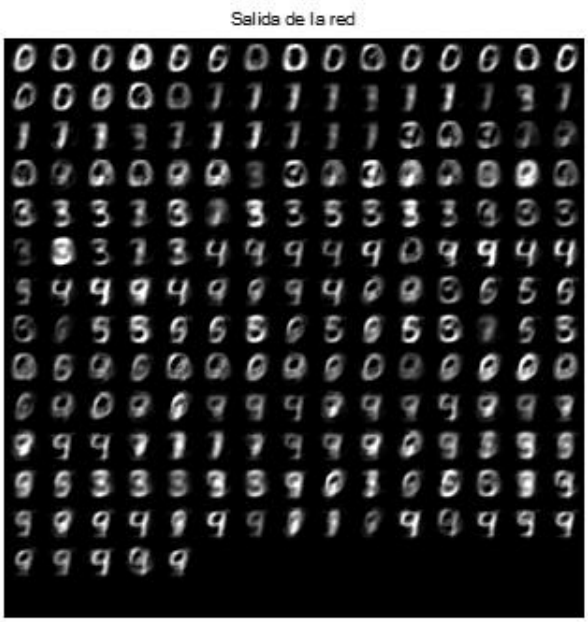


Figura 4.13: Solución para 5 valores singulares más significativos.

Figura 4.14: Solución para 25 valores singulares más significativos.

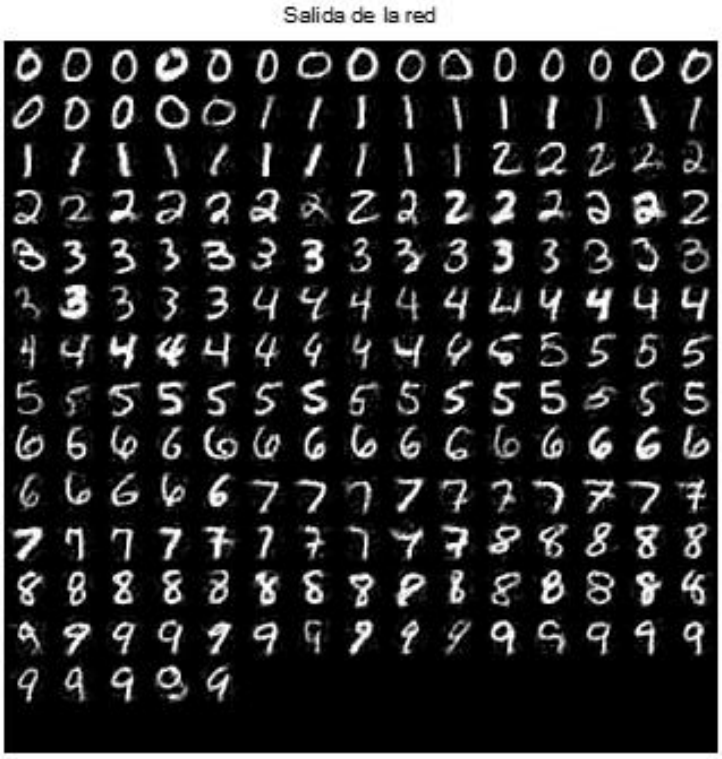


Figura 4.15: Solución para 40 valores singulares más significativos.

Siguiendo el procedimiento descrito en el Apéndice B, en este caso se necesitan $200 + 784 + 1 = 985$ datos para crear cada una de las matrices correspondientes a cada modo. Cada una de estas matrices será de $200\ x\ 784$ datos, lo que significa que no se reduce la dimensión de los datos. Además, teniendo en cuenta los 25 modos más significativos ($25\ x\ 985 = 24625$ datos), se observa que, en el caso anterior, con 120 nodos ocultos se obtienen mejores resultados y a su vez se consigue una mayor compresión de la matriz de datos. Finalmente, se puede comprobar que para obtener unos resultados que se acerquen a los del AE se necesitan tomar los 40 valores singulares más significativos de la matriz de entrada, 39400 datos, casi el doble de los necesarios con el primer método.

4.3. MÁQUINAS DE APRENDIZAJE EXTREMO MULTICAPA (ML-ELM)

En la práctica, los problemas de clasificación complejos requieren la contribución de varias redes neuronales para alcanzar una solución óptima. De acuerdo con el principio "*divide y vencerás*", una tarea compleja se resuelve dividiendo ésta en varias tareas más sencillas y combinando sus soluciones. En la terminología de las redes neuronales, esto se traduce en dividir la tarea de aprendizaje entre varios expertos [35]. Sin embargo, se va a presentar una forma que permite solucionar estos problemas de clasificación sin la necesidad de recurrir a más de una red. Sólo será necesario aumentar el número de capas ocultas de la red de trabajo.

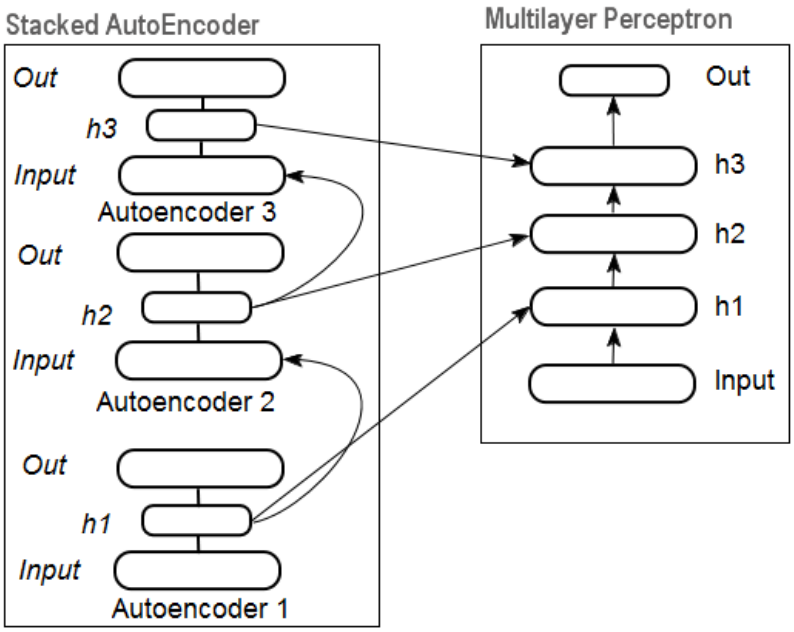


Figura 4.16: Auto-Codificadores apilados.

Como se ha comentado anteriormente, un auto-codificador aprende, mediante el mapeo a un espacio de dimensión distinto, una representación de la información que le llega a su entrada. Esto puede ser muy útil para la resolución de problemas de clasificación como los que se tratan en este proyecto. Sin embargo, se ha indicado en varias ocasiones que el entrenamiento de este tipo de redes es no supervisado, por tanto, ¿cómo se afrontan estos problemas de clasificación? La respuesta es muy sencilla, se aprovecha la gran capacidad de representación

de los AEs para que, mediante su apilado, se pueda crear una red profunda capaz de mejorar los resultados que se obtienen con un simple Perceptrón Multicapa de una capa oculta. Este procedimiento se apoya en numerosos estudios que demuestran que, si un simple AE puede aprender las características simples de los datos de entrada, mediante el apilado de varios de ellos las características aprendidas pueden ser más complejas. En el siguiente punto se presentad de una forma más detallada esta técnica.

4.3.1. Diseño y Entrenamiento de una Máquina de Aprendizaje Extremo Multicapa

Muchos autores han estudiado que las redes neuronales multicapa funcionan de una forma muy pobre cuando éstas son entrenadas simplemente con el algoritmo *Backpropagation*. En una red de arquitectura profunda los pesos de la capa oculta se inicializan usando un correcto entrenamiento no supervisado y finalmente toda la red se entrena, además, con *Backpropagation*. De forma parecida, los pesos de la capa oculta de la red ML-ELM son inicializados usando un ELM-AE, que a su vez es entrenado mediante aprendizaje no supervisado tal y como ya se ha visto. Sin embargo, en contraste con las redes profundas, en el ML-ELM no es necesaria una fase de ajuste [32].

Por tanto, el procedimiento de construcción de esta red multicapa será muy sencillo. Como en los casos anteriores se tendrá un conjunto de entrenamiento, otro de validación y finalmente uno de test. En primer lugar, se usan los conjuntos de entrenamiento y validación para entrenar un auto-codificador, obteniéndose los pesos de la capa de salida $\boldsymbol{\beta}$ correspondientes. El procedimiento de entrenamiento es el mismo que se ha descrito en el Apartado 4.2.2 de este capítulo. Es importante aclarar que los barridos sobre los parámetros del auto-codificador son, en este caso, de: λ $(2^{-10}, 2^{-9}, \dots, 2^{9}, 2^{10})$ y L $(20{:}\,200{:}\,1020)$. Así, los $\boldsymbol{\beta}$ obtenidos serán usados como pesos de la primera capa oculta de la red en construcción multicapa.

Para continuar añadiendo capas a esta red, lo siguiente que hay que hacer es pasar los conjuntos iniciales de entrenamiento y validación por la misma. De esta forma, se obtendrán $\boldsymbol{H}$, que será la entrada al siguiente auto-codificador, y el conjunto de validación para el entrenamiento del siguiente auto-codificador. Con éste, se obtendrán los pesos de la segunda capa oculta de la red multicapa y, por consiguiente, la entrada al tercer auto-codificador, y así sucesivamente. Se podría decir que los conjuntos iniciales de entrenamiento y validación se deben ir propagando por la red ML-ELM para entrenar el auto-codificador con el que se obtendrá la siguiente capa.

Esto, que así a simple vista puede resultar algo engorroso, se puede describir matemáticamente con una simple fórmula:

$$\boldsymbol{H}^k = g\left(\left(\boldsymbol{\beta}^k\right)^T \boldsymbol{H}^{k-1}\right)$$

Donde $\boldsymbol{H}^k$ es la matriz de salida de la capa oculta k-ésima de la red ML-ELM. La capa de entrada puede ser considerada como la capa oculta 0, donde $k = 0$ (Figura 4.17).

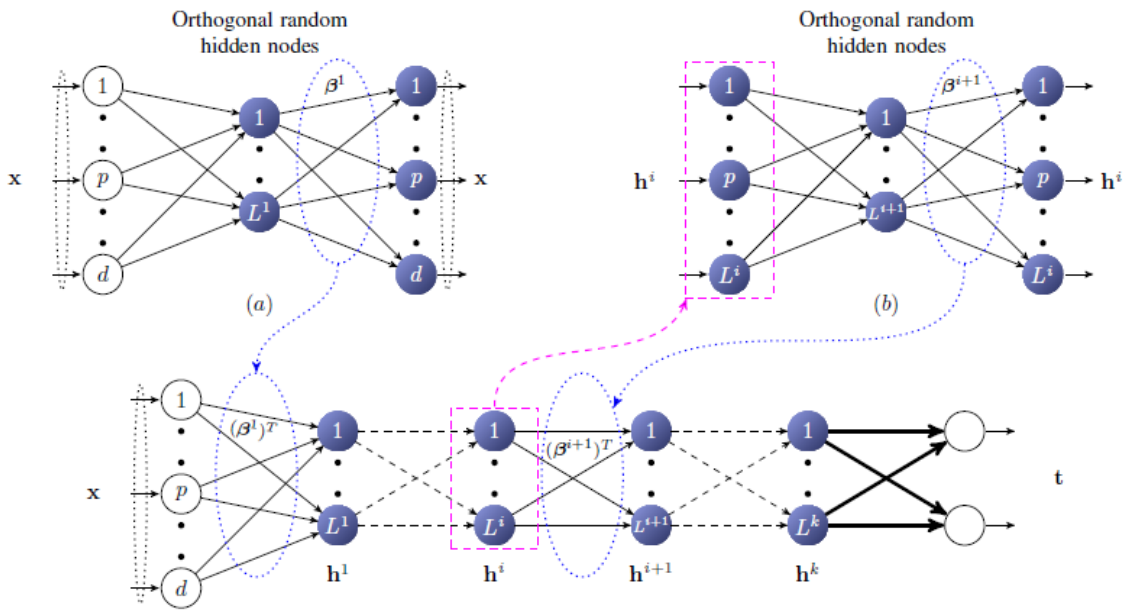


Figura 4.17: Creación de la red Multicapa.

Acorde con la teoría, las funciones de activación de las capas ocultas de la red ML-ELM pueden ser lineales o no lineales a trozos. Si el número de nodos L^k en la capa oculta k-ésima es igual al de la capa $(k-1)$-ésima, la función de activación puede ser elegida como una función lineal y, en otro caso, puede ser no lineal a trozos [32]. No obstante, en el código realizado se ha usado en todo momento la función sigmoidal como función de activación de los nodos ocultos, tanto de los auto-codificadores como de la red ML-ELM.

Finalmente, la parte de la obtención de los pesos de la capa de salida es la relativa al entrenamiento supervisado. Para ello, simplemente se pasa por la red el conjunto de entrenamiento y se entrena realizando un barrido sobre λ $(2^{-10}, 2^{-9}, \dots, 2^{9}, 2^{10})$ en busca de los pesos que minimizan la función Mínimos Cuadrados Regularizados, es decir, usando:

$$\widehat{\boldsymbol{\beta}}_{\boldsymbol{out}} = \left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{T}$$

Donde $\boldsymbol{H}$ es la salida de la última capa oculta de la red ML-ELM.

Pero, ¿cuándo se debe parar de añadir capas ocultas a la red multicapa? Éste es un aspecto importante a tener en cuenta. Para ello se ha seguido un criterio de parada muy sencillo: cada vez que se añade una nueva capa, se calculan los pesos de la capa de salida como se ha comentado y se usa el conjunto de validación para comprobar el funcionamiento de la red. Esto no es más que pasar el conjunto de datos por la red y calcular el error de clasificación cometido. De esta forma, si en algún momento la adición de una nueva capa conlleva que el error aumente, será el momento de parar el entrenamiento y quedarse con la estructura obtenida antes de añadir esta última capa. A continuación se presenta un diagrama que resume el procedimiento de entrenamiento descrito:

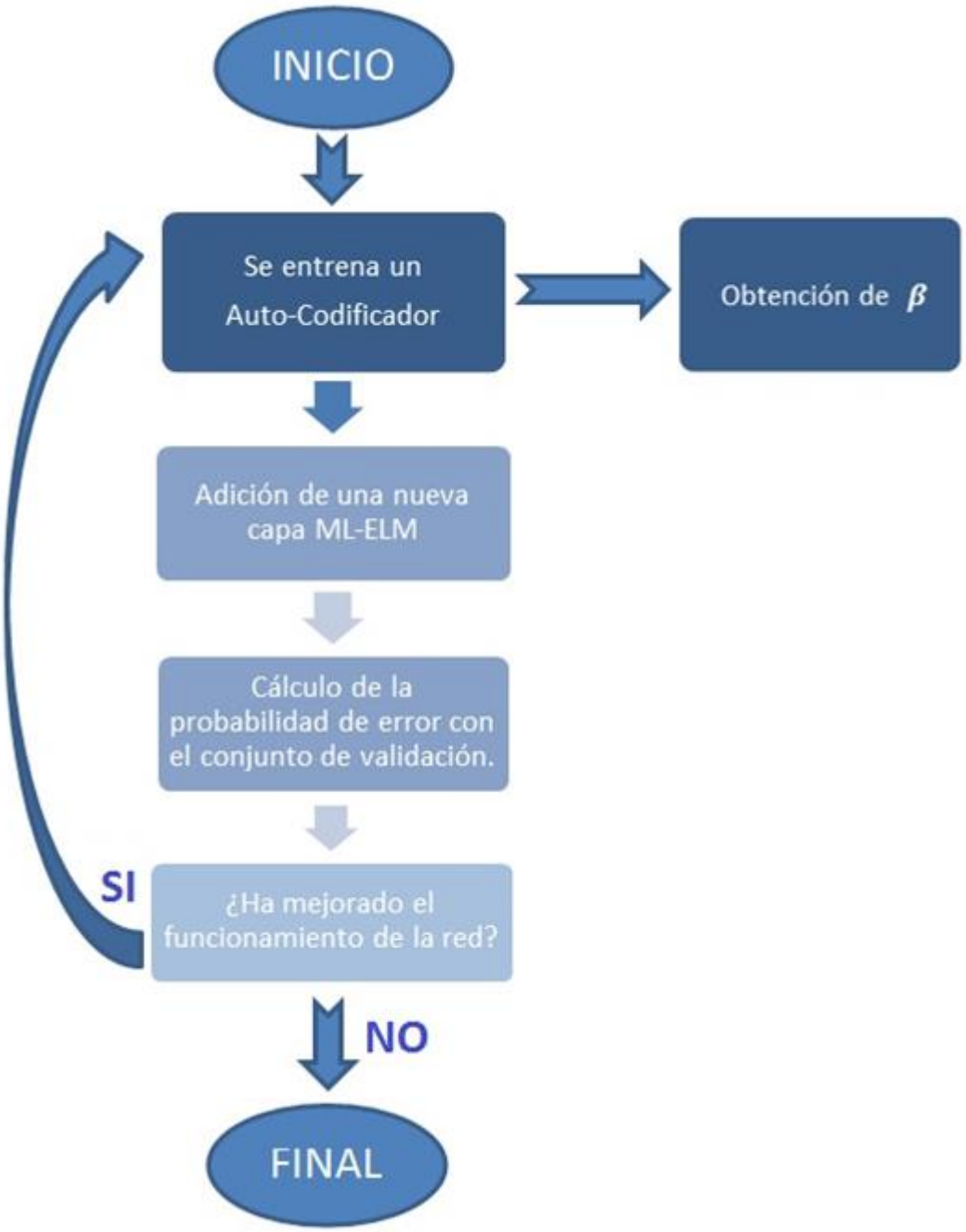


Figura 4.18: Diagrama de flujo del entrenamiento de ML-ELM.

4.3.2. Resultados

Seguido el procedimiento ya explicado, se va creando la red capa a capa y calculando la precisión en cada caso. En la siguiente tabla se aprecia la evolución obtenida usando los conjuntos de entrenamiento y validación:

Número de capas de la red	Probabilidad de acierto de entrenamiento (%)
1	84.78
2	87.75
3	88.85
4	89.47
5	90.19
6	90.93
7	90.03

Tabla 4.1: Resultados de entrenamiento.

Por tanto, la estructura obtenida a través del entrenamiento es de 784-1020-1020-1020-1020-1020-1020-10.

Una vez entrenada la red, se realiza la etapa de testeo con el conjunto formado por 10000 patrones perteneciente a la base de datos MNIST. Esta etapa es la más importante ya que es la que realmente indica el buen funcionamiento de la red y demuestra que ésta ha adquirido una buena capacidad de generalización. Por tanto este resultado es el que demuestra que se ha conseguido mejorar otras estructuras como la de las ELMs estudiadas en el capítulo anterior. Queda demostrado en la Tabla 4.2 que con el incremento del número de capas se van obteniendo mejores resultados en el problema de clasificación, hasta llegar al 91.38% de probabilidad de acierto obtenido con el número óptimo de 6 capas intermedias.

Número de capas de la red	Probabilidad de acierto de test (%)
1	85.72
2	88.33
3	89.50
4	89.98
5	90.80
6	91.38

Tabla 4.2: Resultados finales.

CAPÍTULO 5.

MEDIDA DEL GROSOR ÍNTIMA-MEDIA CAROTÍDEO CON REDES NEURONALES

Después de trabajar con la base de datos MNIST, para acabar con este proyecto se pretende aplicar todos los conocimientos expuestos sobre la medida del grosor de las capas íntima-media de la carótida a partir de imágenes por ultrasonido de la arteria, más conocidas como ecografías.

5.1. INTRODUCCIÓN

Según los últimos datos de la Organización Mundial de la Salud, las enfermedades cardiovasculares (CVDs) se sitúan en el tercer puesto de las causas de mortalidad en los países occidentales. Además de un correcto estilo de vida, la prevención es la clave para reducir estos datos tan abultados. Por tanto, el uso de algún mecanismo que indique el nivel de riesgo de enfermedad es de suma importancia en la práctica clínica. Uno de los métodos más usados como marcador de riesgo de CVDs es el grosor íntima-media carotídeo (IMT), el cual posee muchas ventajas [36].

La manifestación más temprana del posible comienzo de una enfermedad cardiovascular es la aterosclerosis [36]. La ateroesclerosis es un síndrome caracterizado por el depósito e infiltración de sustancias lipídicas en las paredes de las arterias de mediano y grueso calibre. Es la forma más común de arteriosclerosis.

En la Figura 5.1 se puede ver la anatomía de la carótida a lo largo del cuello (izquierda). A la derecha se muestra la arteria carótida común (CCA), que en su parte superior se bifurca en las arterias carótida interior y exterior.

Las imágenes por ultrasonido, también denominadas exploración por ultrasonido o ecografía, son las herramientas más usadas para diagnosticar CVDs. Involucran el uso de un pequeño transductor (sonda) y un gel para ultrasonido para la exposición del cuerpo a ondas acústicas de alta frecuencia. El ultrasonido es seguro y no doloroso, y produce imágenes del interior del organismo usando ondas de sonido. Las examinaciones por ultrasonido no utilizan radiación ionizante (como se usa en los rayos X). Debido a que las imágenes por ultrasonido se capturan en tiempo real, pueden mostrar la estructura y el movimiento de los órganos internos del cuerpo, como así también la sangre que fluye por los vasos sanguíneos.

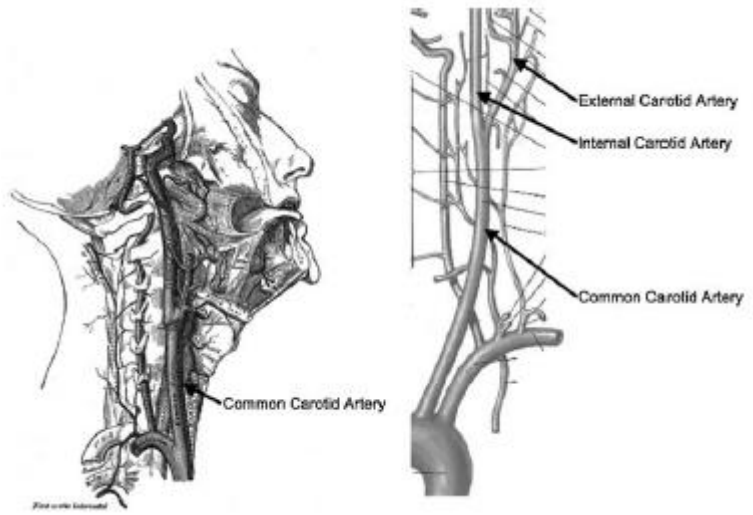


Figura 5.1: Anatomía de la arteria carótida por el cuello.

Un ultrasonido de las dos arterias carótidas del cuerpo, que se hallan a cada lado del cuello y transportan sangre desde el corazón hasta el cerebro, ofrece imágenes detalladas de estos vasos sanguíneos e información sobre la sangre que fluye a través de ellas. La meta principal del ultrasonido de la arteria carótida es buscar en los pacientes una obstrucción o estrechamiento de sus paredes (o estenosis), que, de estar presente, puede aumentar el riesgo de sufrir un derrame cerebral. Si se detecta un estrechamiento significativo, se podría iniciar un tratamiento completo.

La relación entre el IMT arterial y el aumento del riesgo de sufrir una CVD ha sido estudiada por muchos autores. Estos estudios confirman que un incremento por encima del 0.9-1.0 mm del IMT indica un alto riesgo de CVD. Es sabido que aunque el IMT aumenta con la edad, el proceso de la aterosclerosis comienza en la infancia, aunque entre los 5 y los 20 años el IMT sólo puede aumentar suavemente [36].

Por tanto, una forma de controlar el IMT es a través de ecografías en modo-B, un método no invasivo y de bajo coste que permite un examen del paciente en un corto periodo de tiempo.

Esta técnica tiene muchas ventajas entre las que se destacan:

- Usa radiación no ionizante.
- Permite un examen rápido y seguro del paciente.
- El equipamiento está disponible en muchos centros.
- Permite visualizar las paredes de la arteria.

A pesar de estas ventajas, las ecografías presentan una relación señal-ruido (SNR) menor que otras técnicas tales como MRI o CT. Además, el uso de diferentes protocolos y la variedad de observadores son problemas recurrentes en el procedimiento. La repetitividad y reproducibilidad del proceso son de gran importancia para analizar el IMT [37]. Aun así, la evaluación de imágenes por ultrasonidos ha sido la base de muchos estudios clínicos.

Las paredes de la carótida están formadas por tres capas: la más interna es la íntima, la del medio es la media y la más externa es la adventicia. La segmentación de la arteria carótida

común consiste en trazar los perfiles de las interfaces más importantes: la interfaz lumen-íntima (LI) y la media-adventicia (MA). Para poder medir el grosos íntima-media (IMT) se necesita la vista longitudinal de la arteria [36].

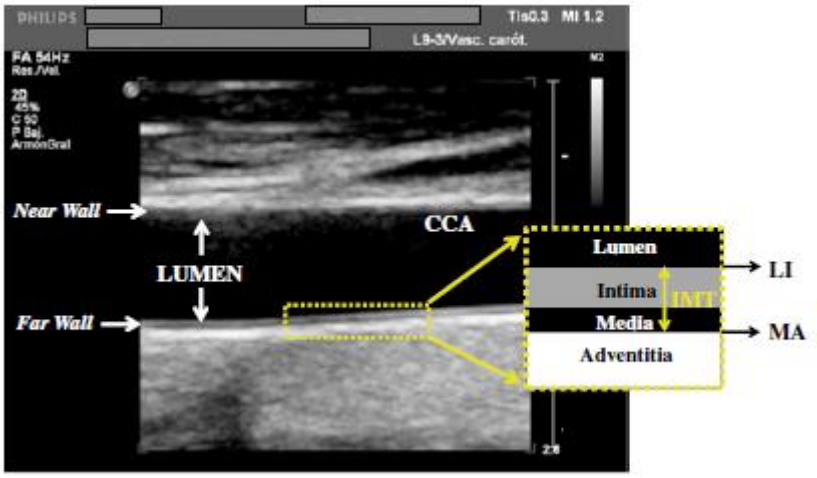


Figura 5.2: Vista longitudinal de la CCA en una ecografía.

Una imagen longitudinal de la arteria muestra la pared cercana a la superficie del cuello y la pared lejana, que es la más interna (Figura 5.2). Tradicionalmente, el lugar donde se debe medir el IMT es en la pared interna, en una sección de aproximadamente 1 cm de longitud y próxima a la zona donde se bifurca. Es ahí donde se corresponde con un patrón claro-oscuro-claro de las capas íntima-media-adventicia de la pared (Figura 5.2). El médico toma tres medidas diferentes del segmento mencionado y se escoge la mayor como el valor final del IMT [37].

En esta parte final del trabajo, se va a ver cómo es posible reducir la subjetividad que conlleva la medida manual del IMT por parte del doctor realizando la medida a lo largo de toda la pared mostrada en la ecografía. Esto permite resultados más precisos y la posibilidad de extraer estadísticas de interés por parte del especialista [37]. Para ello se realizará el procesamiento de la imagen médica a través de las arquitecturas de redes ya estudiadas: ELMs y ML-ELMs basadas en AEs.

Antes de eso, es necesario mostrar el procedimiento por el cual se procesan las imágenes. Primero, la ecografía se pre-procesa para detectar la región de interés (ROI), que estará en la pared interna. Entonces, con las redes neuronales se conseguirá una clasificación de los píxeles de esta región (ROI) en función de su pertenencia al IMT o no, resultando una imagen binaria. Finalmente, esta imagen binaria se post-procesa para extraer el contorno de las interfaces LI y MA [37].

5.2. PROCESAMIENTO DE IMÁGENES Y OBTENCIÓN DE PATRONES

Se ha de aclarar que esta parte del trabajo pertenece a la Tesis Doctoral realizada por Dña. Rosa María Menchón Lara bajo la supervisión del Dr. D. José Luis Sancho Gómez, en la Universidad Politécnica de Cartagena. La información se obtiene del estudio [37].

5.2.1. Adquisición de imágenes

En validación de la técnica de segmentación se ha usado un conjunto de 60 ecografías longitudinales de la CCA en formato DICOM. Todas ellas las proporcionó el Departamento de Radiología del *Hospital Universitario Virgen de la Arrixaca* (Murcia, España). Los sujetos fueron 15 mujeres y 15 hombres de entre 25 y 79 años, con una media de edad de 57 años. Una muestra de las imágenes utilizadas se puede ver en la Figura 5.2.

Todas estas imágenes se obtuvieron con *Sistema de Ultrasonido Philips iU22* y guardadas en escala de grises de 256 niveles. Su resolución espacial va desde 0.029 hasta los 0.081 mm/pixel, con una media y desviación típica de 0.052 y 0.015 mm/pixel, respectivamente. Los parámetros del escáner se ajustaron en cada caso por parte del radiólogo para obtener imágenes con suficiente calidad como para visualizar los límites del IMT. En el estudio se usan imágenes borrosas y afectadas por el ruido, así como algunas en las que solo se aprecia el límite del IMT parcialmente.

Como se ha mencionado, el objetivo es medir de forma automática el IMT. Se aconseja que esta medida se realice en la pared interna y en aquellas regiones que no están afectadas por placas. Por este motivo, en el estudio no se incluyen imágenes con placas visibles.

5.2.2. Contorno manual

Para poder evaluar la precisión del método de segmentación y de la medida realizada, es necesario comparar los resultados con algunos valores de referencia (*Targets*). Aunque no es posible definir la segmentación perfecta, se usa la diferencia de cuatro segmentaciones manuales para realizar la comparación. En particular, dos radiólogos expertos dibujaron el contorno de cada una de las 60 imágenes dos veces, con un tiempo entre medio de dos meses. Así, se obtuvieron un total de 240 segmentaciones manuales. Estos valores de referencia permitirán el estudio de las diferencias entre ambos métodos (manual y automático), así como ver la variabilidad que conlleva una medida manual. Por supuesto, cada segmentación manual incluye la traza de las interfaces lumen-íntima y media-adventicia. Por medio de estas aplicaciones, es posible marcar con el ratón del ordenador todos los puntos que se quiera sobre la imagen. El resto de puntos para completar el contorno se realizarán por interpolación.

5.2.3. Método de segmentación y obtención de patrones

En primer lugar, para cualquier imagen de la CCA, se realiza una etapa de pre-procesado para detectar la región de interés (ROI). Después, mediante un proceso de enventanado se construyen los patrones correspondientes a cada píxel. De esta forma, quedan preparados los

conjuntos de datos para una posterior etapa de clasificación usando tanto las ELMs como ML-ELMs ya vistas. Finalmente nos quedarán unos resultados de clasificación que se pueden pasar por una última etapa de post-procesado y extraer el contorno final de las interfaces LI y MA, aunque ésta no se realizará en este trabajo.

5.2.3.1. Etapa de pre-procesado

En las ecografías el lumen corresponde a una región oscura (Figura 5.2) delimitada por las paredes de la arteria. Se puede apreciar, sobre este lumen, el eco correspondiente a la pared más superficial al cuello. La pared donde se realiza la medida se encuentra debajo del lumen, y con esta etapa de pre-procesado se localiza de forma automática. Los pasos a seguir, a gran escala, se muestran en la Figura 5.3. Por otro lado, se pueden apreciar las respectivas operaciones necesarias de una forma más específica en la Figura 5.4. Aquí no se va a entrar en qué consiste cada una de dichas operaciones, sólo importa que una vez realizados todos los pasos se obtendrá una imagen binaria como la mostrada en la Figura 5.5 (izquierda) en la que se distingue fácilmente el lumen.

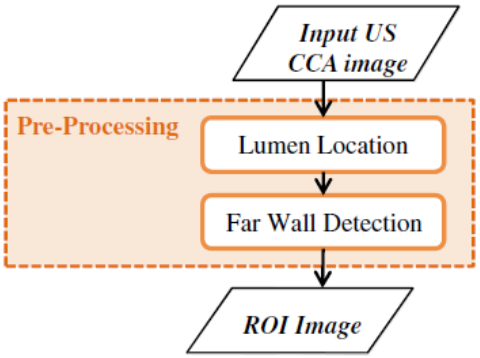


Figura 5.3: Etapa de pre-procesado.

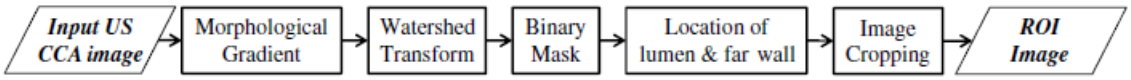


Figura 5.4: Operaciones del pre-procesado.

Una vez localizado el lumen, simplemente hay que centrarse en el límite correspondiente a la pared interna de la CCA, y se establecen los límites de la región de interés ROI. Para el límite superior se traza una línea 0.6 mm por encima del punto más alto de la pared de interés, y el límite inferior se fija a 1.5 mm por debajo del punto más bajo (Figura 5.5 derecha). Así, el tamaño de la ROI está relacionado con la forma de la arteria en la ecografía.

5.2.3.2. Obtención de patrones

Ya que la segmentación de imágenes puede ser considerada como una clasificación de píxeles, normalmente este problema se trata como uno de reconocimiento de patrones y para ello se usan gran variedad de técnicas. Además, las técnicas de reconocimiento de patrones proporcionan la flexibilidad y automatización que se necesitan en este caso para el procesamiento de las imágenes médicas.

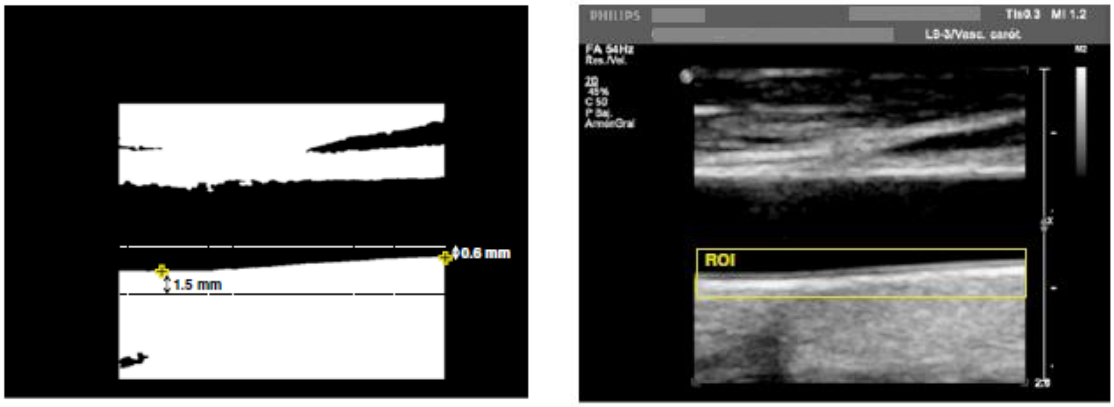


Figura 5.5: Imagen binaria tras el pre-procesado (izquierda) y establecimiento del límite de la ROI (derecha).

Por tanto, como información de entrada a la red que va a realizar la clasificación se tomarán los valores de intensidad de los píxeles que se encuentren en un vecindario del píxel que se pretende clasificar. Para ello, se realizará una ventana cuadrada de tamaño $W \times W$ que se irá desplazando píxel a píxel sobre la imagen de entrada (en este caso, la imagen ROI). Esta ventana proporciona información sobre el patrón de los valores de intensidad en el vecindario del pixel central. De esta forma, si por ejemplo se tiene una imagen de entrada de $730 \times 110 = 80300$ píxeles, se realizaría un barrido por cada uno de ellos y en función del tamaño de la ventana W se obtendría una matriz de datos de entrada de tamaño 80300×9, 80300×49 y 80300×121, para $W = 3$, $W = 7$ y $W = 11$, respectivamente (Figura 5.6). Así, cada dato de entrada a la red sería cada una de las filas de la matriz resultante.

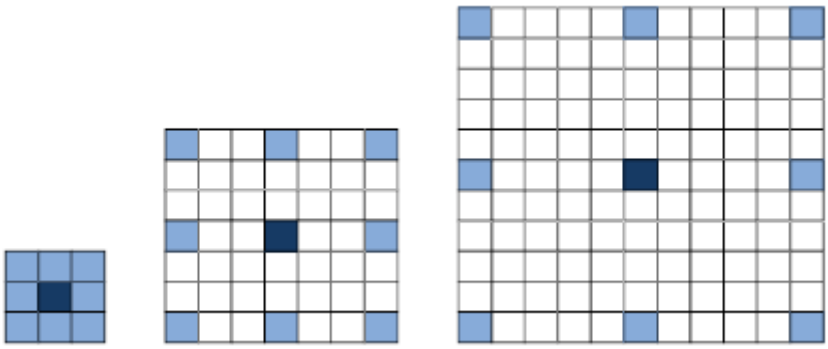


Figura 5.6: Ventanas cuadradas para $W = 3$, $W = 7$ y $W = 11$, respectivamente.

Para entrenar una red neuronal mediante aprendizaje supervisado (en este caso cuando se trabaje con ELM o en la capa de salida del ML-ELM) es necesario un conjunto de datos etiquetados o *Targets*. En este proyecto, para asegurar una buena capacidad de generalización de la red, se eligieron cuidadosamente cinco imágenes heterogéneas (con diferente orientación de la CCA, resolución espacial, etc.) para obtener un conjunto de datos consistente y representativo. Es importante mencionar que usar todos los píxeles/patrones de una imagen para el entrenamiento es algo inapropiado ya que el conjunto de entrenamiento sería muy grande. Por tanto el conjunto de datos se creó a partir de diferentes muestras de las cinco imágenes segmentadas seleccionadas, obteniendo un total de 8000 muestras/patrones. Este

conjunto se dividirá en tres diferentes tal y como se ha explicado a lo largo de todo el proyecto. El conjunto de entrenamiento estará formado por un 60% del total, es decir por 4800 muestras; y tanto el de validación como el de test tendrán una extensión de 1200 muestras cada uno (20% de las 8000 totales).

5.3. MEDIDA DEL IMT MEDIANTE UNA ELM

Entrenando la red de la forma explicada en el Capítulo 3 y usando los conjuntos ya comentados, se puede representar el error de clasificación cometido según los valores (L, λ):

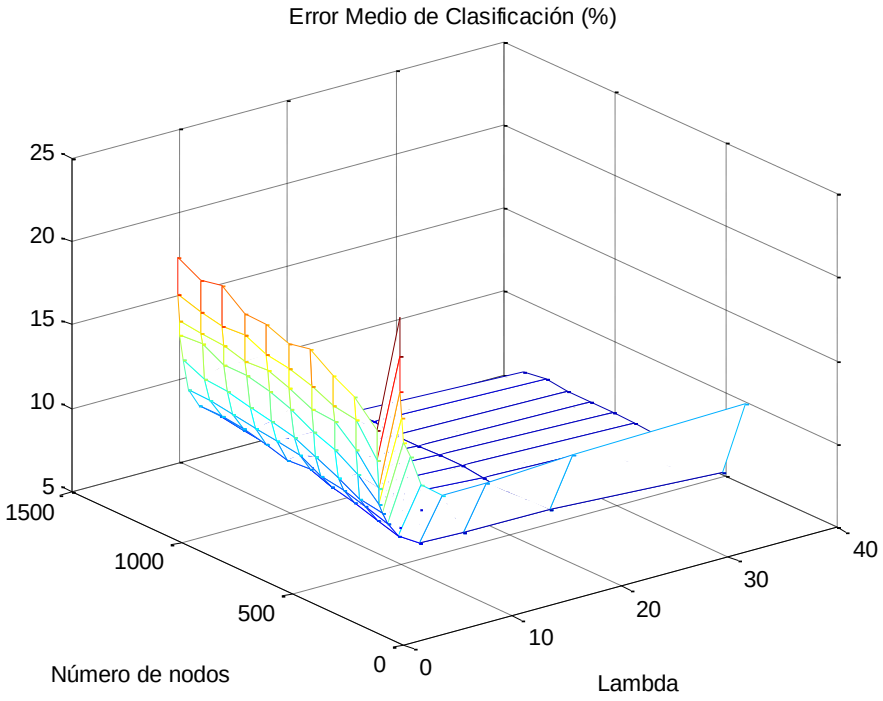


Figura 5.7: Error de clasificación con ELM.

En este caso se consigue un menor error para los valores de $L = 120$ nodos ocultos y $\lambda = 32$. Así, establecidos todos los parámetros de la red, ésta se testea obteniendo una probabilidad de acierto real del *89.70 %.*

Es el momento de pasar a la etapa de ejecución, en la que se pasarán por la red varias de las imágenes segmentadas de las que ya se ha hablado, y se comprobará cómo ésta es capaz de obtener una imagen binaria (en blanco y negro) a la que se le podría realizar una última etapa de post-procesado para obtener el contorno final. No obstante, en estas imágenes binarias ya se puede apreciar el contorno buscado de una forma clara. A pesar de que la red se validó mediante el procesado de todas las imágenes, a continuación se presentan sólo algunos ejemplos:

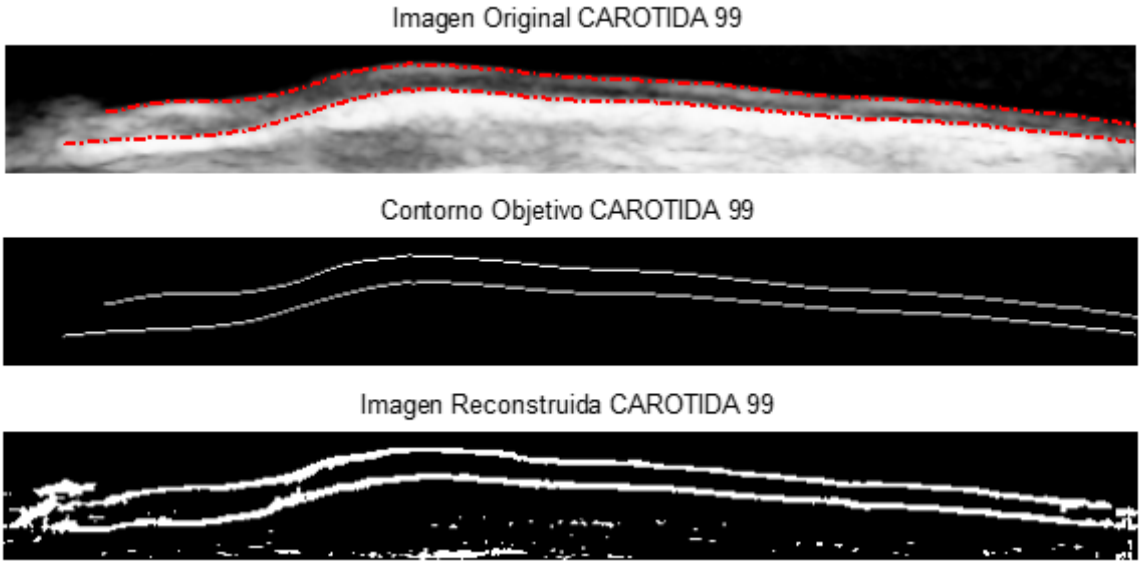


Figura 5.8: Imagen "CAROTIDA 99" con ELM.

La probabilidad de acierto obtenida al procesar esta imagen es del *90.75 %.*

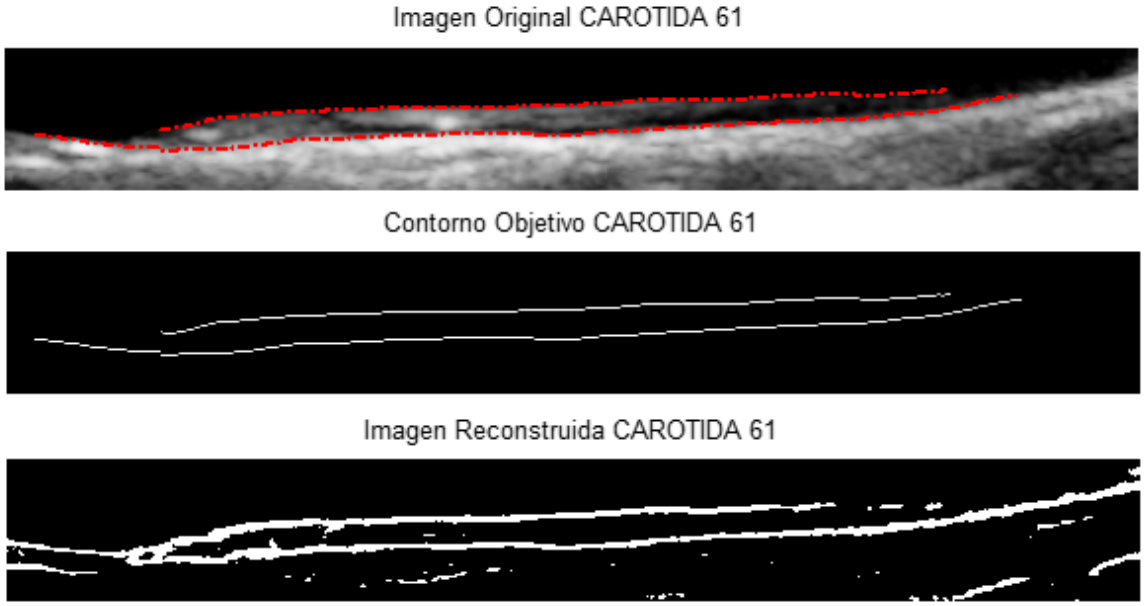


Figura 5.9: Imagen "CAROTIDA 61" con ELM.

En este caso se obtiene una probabilidad de acierto del *93.06 %.*

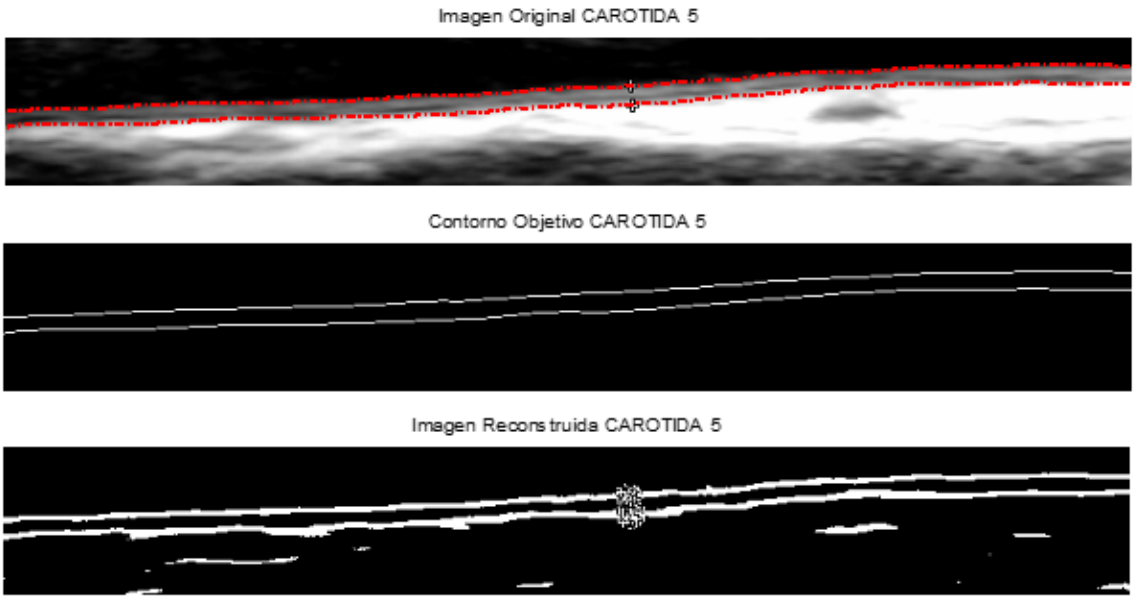


Figura 5.10: Imagen "CAROTIDA 5" con ELM.

Y con esta otra, un *92.83 %* de precisión.

5.4. PROCESAMIENTO DE IMÁGENES ULTRASONIDO CON AUTO-CODIFICADORES.

Ya se conoce el motivo por el que se presentan estos elementos antes de la red ML-ELM que se verá en el siguiente punto. Por tanto, sólo falta mostrar los resultados obtenidos al entrenar este tipo de red con los conjuntos de datos del problema que se está presentando.

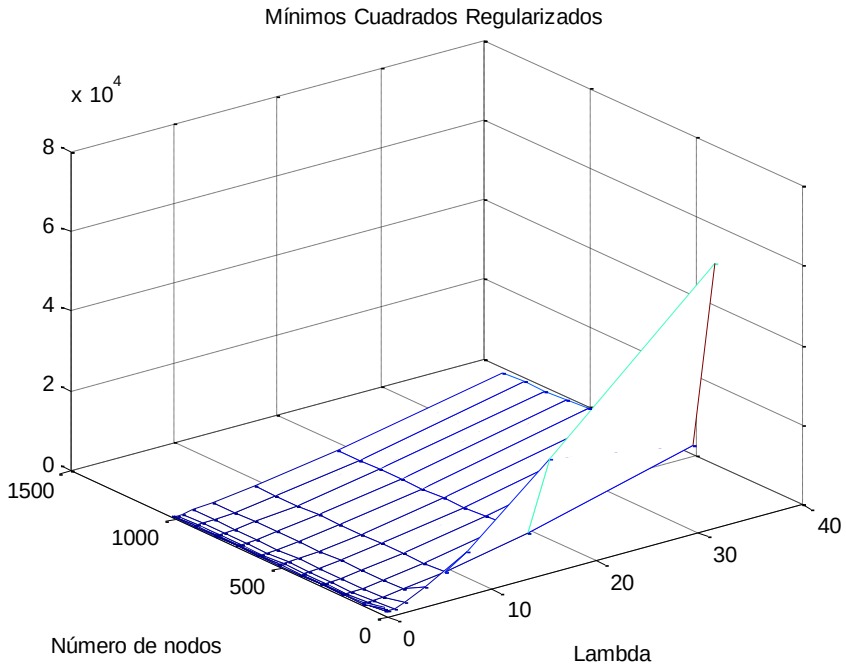


Figura 5.12: Error por Mínimos Cuadrados Regularizados.

De esta forma, se obtienen unos valores óptimos de:

- Mínimos Cuadrados: $L = 1020$ y $\lambda = 32$.
- Mínimos Cuadrados Regularizados: $L = 1020$ y $\lambda = 0.5$

Con estos valores se puede comprobar que la capacidad de representación de las redes es excelente. Para ello, se representan algunas de las imágenes usadas en el trabajo:

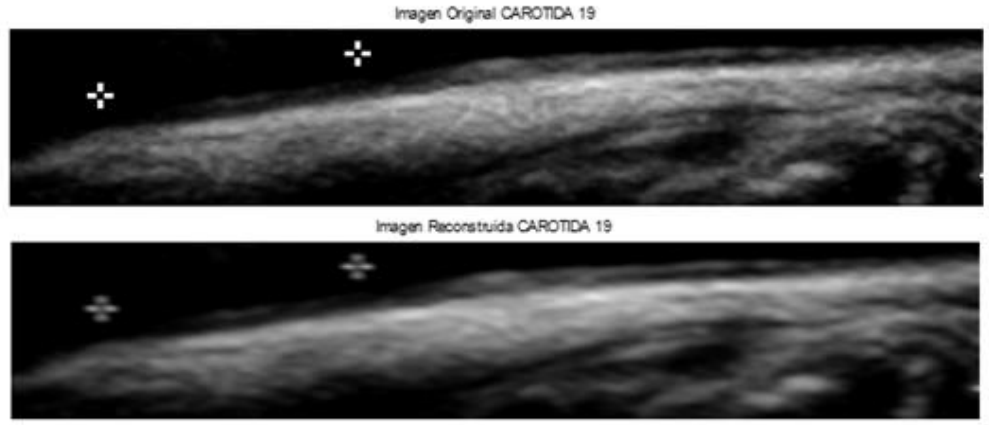


Figura 5.13: Representación de la imagen "CAROTIDA 19" con un Auto-Codificador.

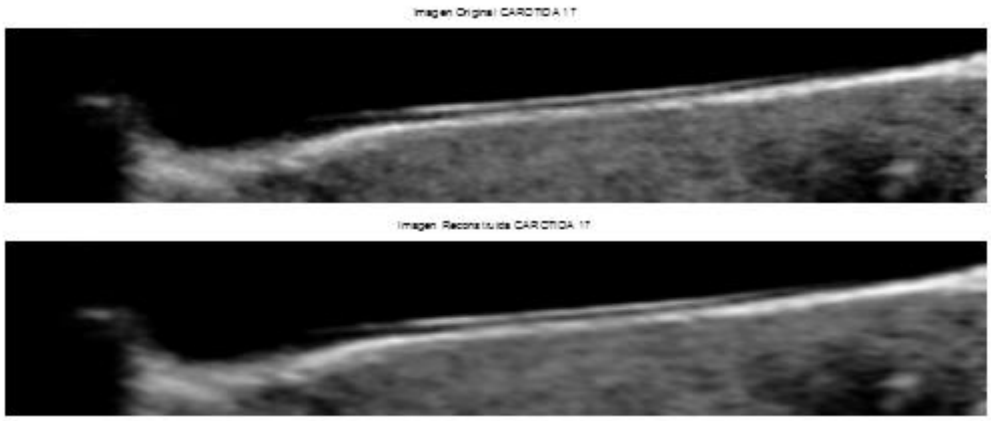


Figura 5.13: Representación de la imagen "CAROTIDA 17" con un Auto-Codificador.

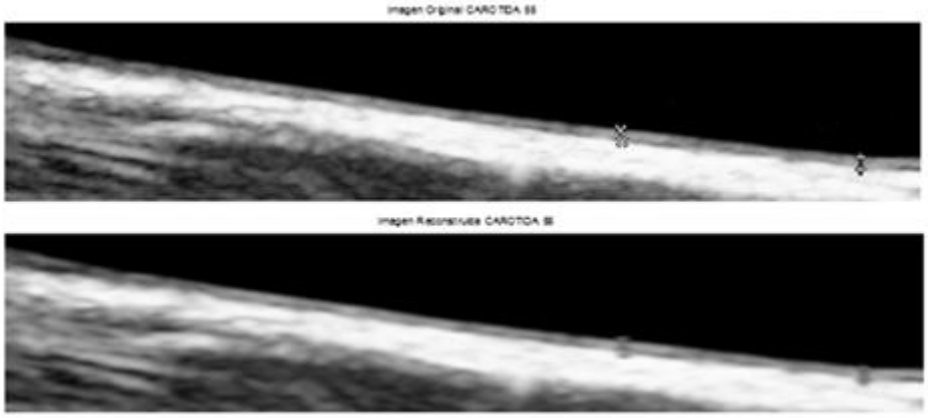


Figura 5.14: Representación de la imagen "CAROTIDA 55" con un Auto-Codificador.

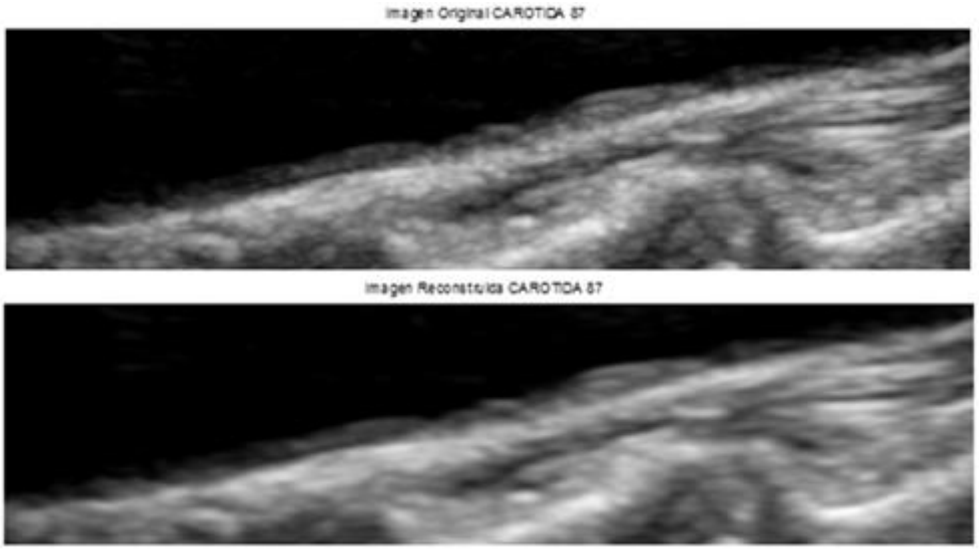


Figura 5.13: Representación de la imagen "CAROTIDA 87" con un Auto-Codificador.

5.5. MEDIDA DEL IMT MEDIANTE UNA ML-ELM

Como en los casos anteriores, lo primero que se hace es entrenar la red, y con ello se obtiene una estructura con el número óptimo de 2 capas intermedias, cuya forma final es de: 121-1020-1020-1. En la siguiente tabla se presenta la evolución del error de clasificación cometido en el entrenamiento.

Número de capas intermedias	Probabilidad de acierto (%)
1	91.1875
2	92.19
3	91.10

Tabla 5.1: Error de entrenamiento del ML-ELM.

Una vez creada la red, ésta se testea con el conjunto ya mencionado de 1200 patrones, obteniendo una probabilidad de acierto del *90.70 %,* y demostrando que en una aplicación real se vuelve a cumplir el objetivo de este estudio: mejorar los resultados obtenidos con una simple Máquina de Aprendizaje Extremo (que era del *89.70 %*).

Pero, para poder ver esto de una forma más clara, lo mejor es pasar a la fase de ejecución, en la que se valida la red procesando las imágenes segmentadas. Se presentan a continuación los mismos ejemplos vistos para el ELM, con objeto de ver la mejoría en la probabilidad de acierto.

Imagen	Probabilidad de acierto (%)
CARÓTIDA 99	91.70
CARÓTIDA 61	93.93
CARÓTIDA 5	93.35

Tabla 5.2: Precisión obtenida con el procesado de las imágenes mediante una ML-ELM.

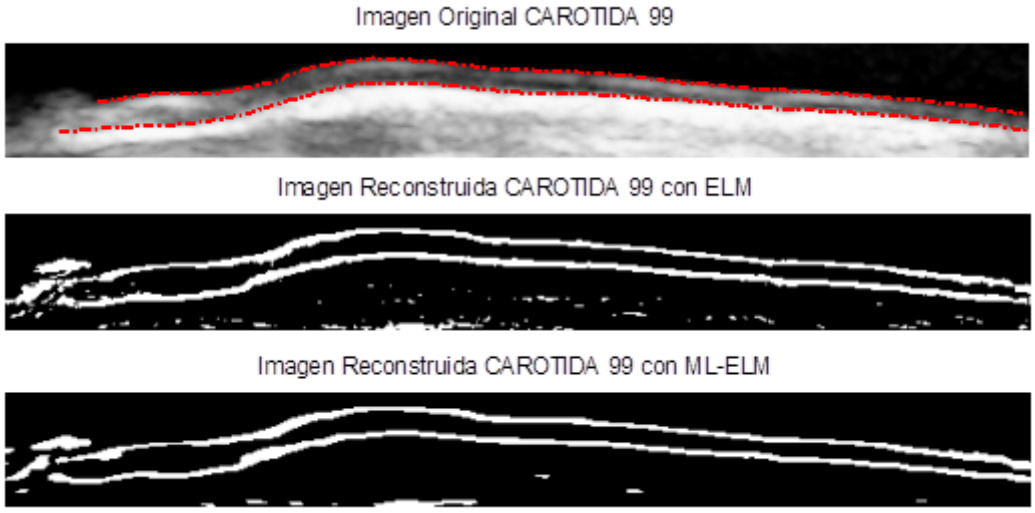


Figura 5.14: Imagen "CAROTIDA 99" con una ML-ELM.

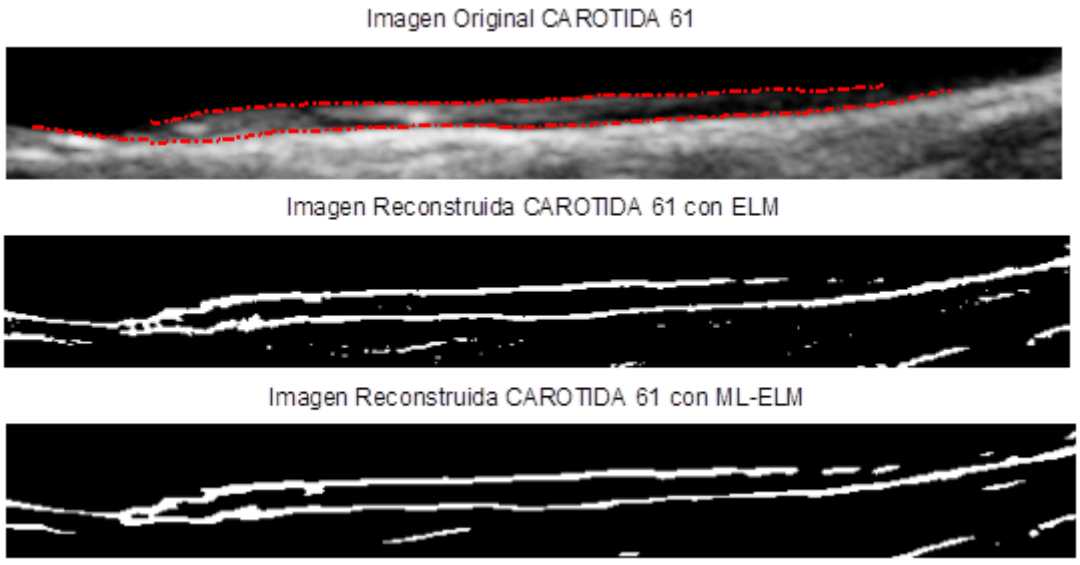


Figura 5.15: Imagen "CAROTIDA 61" con una ML-ELM.

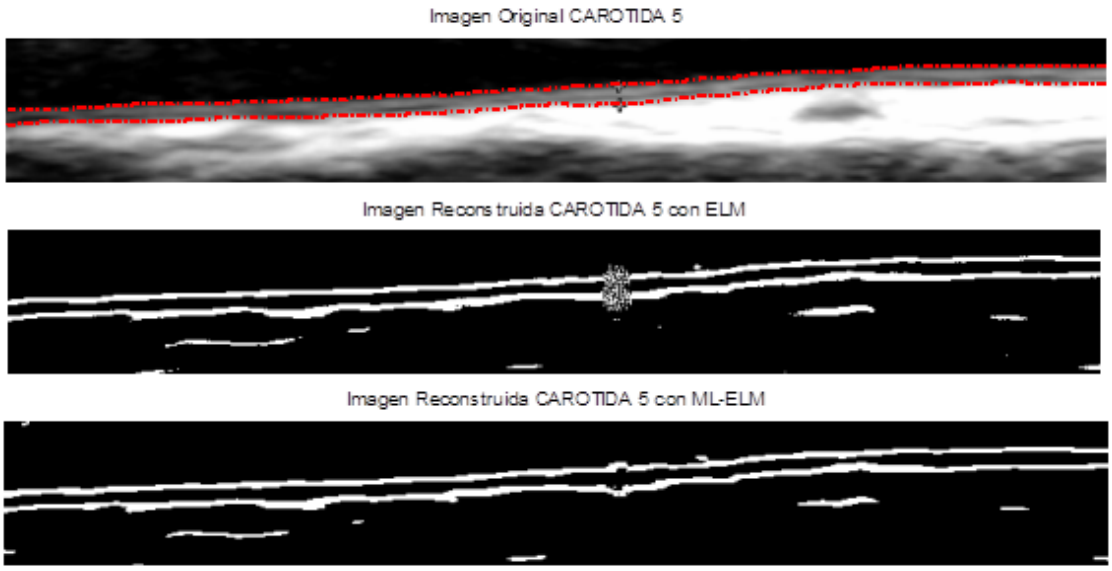


Figura 5.16: Imagen "CAROTIDA 5" con una ML-ELM.

Como se puede apreciar en la Tabla 5.2, con una ML-ELM se consiguen mejores resultados que con una ELM al procesar las imágenes segmentadas. Esto, aunque no se puede apreciar con mucha claridad en los contornos de las imágenes superiores, se ve que se "limpian" las partes más oscuras de las mismas, demostrando que se cumple el objetivo del estudio.

CAPÍTULO 6.

CONCLUSIONES, MEJORAS Y LÍNEAS FUTURAS

En primer lugar, en este estudio se ha presentado un algoritmo de aprendizaje eficiente para Redes Neuronales Progresivas (SLFN*)* llamado *Extreme Learning Machine (*ELM*)*. Como una técnica de aprendizaje, el ELM ha demostrado un buen potencial para resolver problemas tanto de clasificación como de regresión. Así, las principales conclusiones que se han podido sacar son las siguientes:

1) La velocidad de aprendizaje del algoritmo es extremadamente alta.

2) Destaca por su simplicidad. Se ha demostrado cómo una red puede ser entrenada con tres simples pasos.

3) Este algoritmo posee una gran capacidad de generalización a la hora de afrontar problemas de clasificación. Se ha podido observar cómo aumentando λ y L se van obteniendo mejores resultados de forma progresiva. Esto es así debido a que aumentando L se va creando una red más potente y, a su vez, el aumento de λ conlleva una disminución de la norma de los pesos de la capa de salida, lo que deriva en un aumento de la capacidad de generalización, suavizando la frontera de decisión y, por tanto, evitando el sobreajuste.

Además, se han analizado las prestaciones de una variante del ELM (ELM *Auto Encoder*, ELM-AE). Se trata de un caso especial del ELM en el que la entrada es igual a la salida deseada, y los pesos, generados aleatoriamente, son elegidos para que sean ortogonales. Respecto a esta estructura, se puede concluir:

1) Gracias a su extraordinaria capacidad de representación, el ELM-AE puede proporcionar una buena solución a redes SLFN frente a problemas de regresión.

2) Comparando este método con el procedimiento de Descomposición en Valores Singulares (SVD) para el tratamiento de imágenes, se ha podido ver que, aunque con un gran número de nodos ocultos la solución es excelente, con un simple Auto-

Codificador de tan sólo 120 nodos ocultos se mejoran los resultados que se obtienen con el SVD (teniendo en cuenta los 25 valores singulares más significativos). En general, con un ELM-AE se puede reducir la dimensión de los datos de entrada y obtener una matriz más comprimida que con el otro método. A su vez, con esta estructura se tendrán las propias ventajas del algoritmo ELM, tales como la capacidad de generalización.

3) Usar esta estructura para la creación de ML-ELMs es una buena alternativa de cara a mejorar las ventajas que proporciona el entrenamiento de un MLP con el algoritmo ELM.

Después de realizar el diseño y entrenamiento de una Máquina de Aprendizaje Extremo Multicapa, se ha demostrado que proporciona mejores resultados ante problemas de clasificación que un MLP entrenado con este mismo algoritmo. Es necesario, sin embargo, calcular mediante validación el número óptimo de capas.

Finalmente, en el Capítulo 5 se ha visto que todos estos conceptos pueden ser perfectamente aplicados para resolver problemas de aplicaciones reales, como en este caso, la medida del grosor íntima-media carotídeo. Se ha demostrado que con una ML-ELM se consigue una imagen binaria donde las interfaces LI y MA se distinguen claramente (aunque es necesaria una etapa sencilla de post-procesado para perfilarlas mejor). De esta forma, se puede medir el grosor objetivo a lo largo de toda la arteria, mejorando los métodos actuales de medida.

Sin embargo, como ya se ha mencionado, para el desarrollo del proyecto se ha tenido una gran limitación a nivel de Hardware. Por tanto, la principal mejora que podría ser realizada sería un mayor refinamiento en el entrenamiento de todas las arquitecturas de red estudiadas. Es decir, los barridos realizados en busca de los parámetros óptimos se deberían realizar sobre espectros más amplios y con intervalos más pequeños. Además, con el objetivo de demostrar que estas nuevas ML-ELMs son una buena alternativa para el procesamiento de Grandes Datos, se podría ampliar el estudio actual comparando estas redes con otras arquitecturas profundas existentes a la hora de afrontar problemas de aplicaciones reales como el que aquí se ha visto. Una vez validado el algoritmo, el siguiente paso sería crear un programa en *C* y conseguir integrar esta aplicación en los hospitales.

Como líneas futuras de investigación, se podría continuar explorando las capacidades de estas arquitecturas novedosas que son los auto-codificadores, con el objetivo de explotar su capacidad de representación, compresión y regeneración de datos. Una buena aplicación podría ser la de entrenar esta estructura para que sea capaz de distinguir un objeto concreto dentro de una imagen más amplia. También podría investigarse la opción de regenerar imágenes o datos que estén dañados por algún motivo.

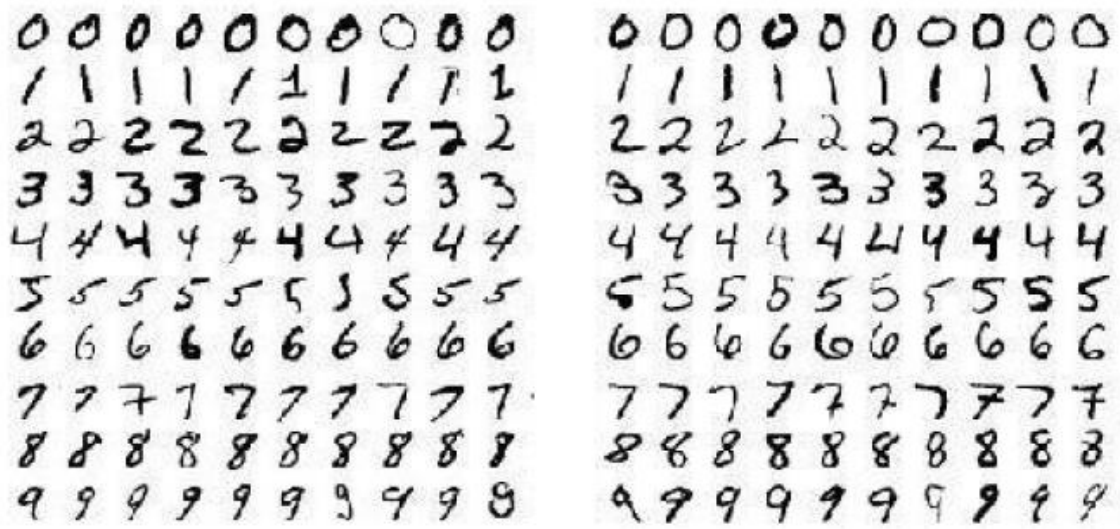
Sin duda, los auto-codificadores poseen un gran potencial que podrá ser explotado en un futuro de cara a mejorar lo que hasta hoy se conoce en el campo del aprendizaje máquina.

APÉNDICE A.

BASE DE DATOS MNIST

La base de datos usada para entrenar y testear los sistemas descritos en este trabajo fue construida a partir de la Base de Datos Especial 3 (SD-3) y la Base de Datos Especial 1 (SD-1) del NIST, *National Institute of Standards and Technology,* y contiene 70000 imágenes de números escritos a mano incluyendo diferentes estilos de escritura. Al principio el NIST estableció el SD-3 como conjunto de entrenamiento y el SD-1 como conjunto de test, sin embargo, el SD-3 es mucho más limpio y fácil de reconocer. Esto es así porque el conjunto de SD-3 fue obtenido de entre los empleados de la Oficina del Censo, mientras que el SD-1 se obtuvo de entre estudiantes de un instituto. Se demostró que para tomar conclusiones sensatas de los experimentos de aprendizaje era necesario que los resultados fueran independientes de la elección del conjunto de entrenamiento y test. Por lo tanto, fue necesario construir una nueva base de datos mezclando los conjuntos de datos del NIST [38].

SD-1 contiene 58527 imágenes digitales escritas por 500 escritores diferentes. En contraste con el SD-3 donde los bloques de datos de cada escritor aparecen de forma secuencial, en el SD-1 están mezclados. Las identidades de los escritores del conjunto SD-1 están disponibles y son usadas para ordenarlos. Con el objetivo de construir esta nueva base de datos se divide el SD-1 en dos: el nuevo conjunto de entrenamiento estará formado por los caracteres escritos por los primeros 250 escritores, mientras que con los 250 restantes se forma el conjunto de test. De esta forma se tienen dos conjuntos de aproximadamente 30000 muestras cada uno. El resto de muestras del conjunto de entrenamiento se obtienen del SD-3, comenzando por el patrón número 0 hasta completar 60000 muestras. De igual manera, el nuevo conjunto de test se completa con muestras del SD-3, comenzando por el patrón número 35000, hasta obtener un total de 60000 patrones. No obstante, en los experimentos que se realizan en este proyecto se utiliza un subconjunto de 10000 imágenes de test (5000 del SD-1 y 5000 del SD-3), y el conjunto completo de entrenamiento formado por las 60000 muestras. A la nueva base de datos resultante se le llamó MNIST, *Modified* NIST.



(a) Conjunto de Entrenamiento (b) Conjunto de Testeo

Figura 7.1: Base de datos MNIST.

Las imágenes en blanco y negro originales se normalizaron de tal forma que cupiesen en un cuadrado de 20x20 píxeles manteniendo su forma. Las imágenes resultantes contienen un nivel de gris motivo de la técnica de interpolación de imágenes llevada a cabo en el algoritmo de normalización. Se usaron tres versiones diferentes de la base de datos. En la primera, las imágenes se centraron en una imagen de 28x28 píxeles calculando el centro de masa de los píxeles y trasladándolo al centro de la imagen de 28x28. Precisamente esta versión es la utilizada en este proyecto, en la que cada imagen presenta un tamaño de 28x28 píxeles y está representada en escala de grises de 256 valores. Esta base de datos es de acceso libre. Algunos ejemplos pueden verse en la Figura 7.1.

APÉNDICE B.

DESCOMPOSICIÓN EN VALORES SINGULARES (SVD)

La descomposición en valores singulares de una matriz no es más que una factorización de la misma en tres matrices diferentes. Sin embargo, lo importante aquí es el tratamiento de imágenes a través de la descomposición en valores singulares, ya que el objetivo es comparar este método con el funcionamiento de los auto-codificadores. Para explicar el funcionamiento del SVD, se tomará como ejemplo de imagen el sugerente grabado *Melancolía* de Albrecht Dürer [39].

Figura 8.1: Melancolía (Albrecht Dürer).

Lo que interesa aquí es que una imagen digital no es más que una matriz de píxeles. En concreto, esta imagen se almacena como una matriz de 736×566 píxeles. El objetivo es comprimir esta imagen sin perder mucha resolución. Es decir, tratar de conseguir la misma imagen, quizá con unos tonos de grises menores, con muchos menos datos.

Llamando A a la matriz de tamaño 736×566 que contiene los datos necesarios para obtener la imagen de la Figura 8.1. Se necesitan, por tanto 416576 números en punto flotante (datos)

para conseguir la imagen. ¿Cómo se pueden usar los valores singulares para obtener esta misma imagen con menos datos? La idea es la siguiente: de acuerdo con el Teorema SVD, existen matrices ortogonales U y V tales que

$$A = U\Sigma V^T$$

Siendo U una matriz de tamaño 736×736, V de tamaño 566×566, ambas ortogonales, y

$$\Sigma = \begin{bmatrix} S \\ 0 \end{bmatrix} \qquad S = Diag(s_1, s_2, \dots, s_{566})$$

Con $s_1 \geq s_2 \geq \cdots \geq s_{566}$ los valores singulares de A.

El número de filas especificadas como cero en Σ es, cuando menos, $736 - 566 = 170$, así que las últimas 170 columnas de U no juegan ningún papel. Se denota como U_1 la submatriz de U formada por sus primeras 566 columnas. Se puede escribir

$$A = U_1\Sigma V^T$$

Si u_i representa la i-ésima columna de U_1 y v_i^T la i-ésima fila de V^T entonces

$$A = \sum_{i=1}^{566} s_i u_i v_i^T$$

Esto es fácil de ver. Un ejemplo con números más pequeños como con A una matriz 4×3:

$$A = [u_1 \quad u_2 \quad u_3 \quad u_4] \begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix} = [u_1 \quad u_2 \quad u_3] \begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix}$$

$$A = s_1 u_1 v_1^T + s_2 u_2 v_2^T + s_3 u_3 v_3^T$$

Volviendo a la matriz original, cada una de las matrices $s_i u_i v_i^T$ es de tamaño 736×566 pero sólo se necesitan $736 + 566 + 1 = 1303$ datos para formarla: los 736 elementos de u_i, los 566 de v_i^T y 1 de s_i. A la matriz $s_i u_i v_i^T$ se le llama el modo i-ésimo de A y algunos modos son más dominantes que otros para formarla.

Las matrices $u_i v_i^T$ son matrices de rango 1 de forma que determinan subespacios de dimensión 1, que se pueden identificar con direcciones en $\mathbb{R}^{566}$. El valor de s_i da la coordenada de A en la dirección $u_i v_i^T$. Si hay valores singulares mucho mayores que otros, las direcciones de estos valores singulares son más determinantes en la formación de A que las de los valores singulares pequeños.

Por tanto, aproximando A por la suma de sus primeros modos se consiguen matrices que toman en consideración las direcciones fundamentales de A. Las imágenes de estas matrices deberán parecerse a las de A a medida que el número de modos que se tomen en consideración aumente (Figuras 8.2 y 8.3).

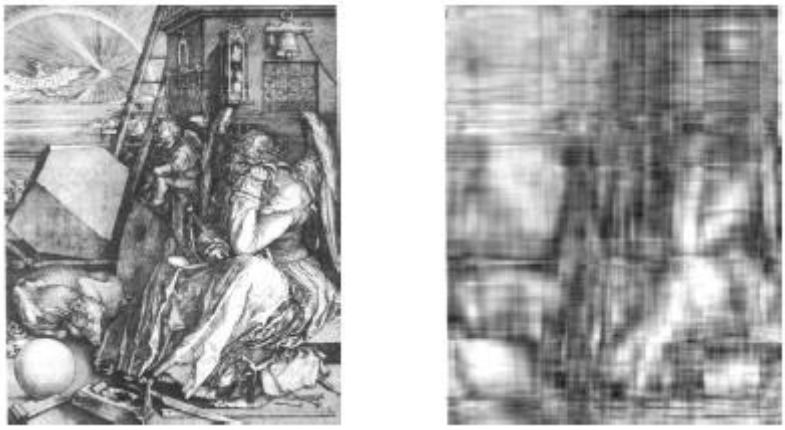


Figura 8.2: Melancolía usando 10 modos.

Como se muestra, tomando 10 modos (13030 datos) la resolución de la imagen es bastante baja, pero si se va incrementando el número de modos ésta mejora, hasta tal punto que con 150 modos (195450 datos, un 47% de la matriz original) no se distingue fácilmente la original de la copia.

Figura 8.3: Melancolía usando 150 modos.

BIBLIOGRAFÍA

1. Y. Bengio, "Learning Deep Architectures for AI", *Foundations and Trends in Machine Learning Vol. 2, No. 1 (2009) 1-127*.
2. E. Alpaydin, "Introduction to Machine Learning", *(2010) 2nd ed. Massachusetts Institute of Technology.*
3. R. Rojas, "Neural Networks. A Systematic Introduction", *Springer-Verlag, Berlin, 1996.*
4. G. Peter Zhang, "Neural Networks for Classification: A Survey", *IEEE Trans. Syst., Man, Cybern.—Part C: Applications and Reviews, vol. 30, No. 4, November 2000.*
5. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, "Extreme Learning Machine: Theory and applications", *School of Electrical and Electronic Engineering, NanyangTechnological University, Nanyang Avenue, Singapore 639798, Singapore. Neurocomputing 70 (2006) 489-501.*
6. K. Hornik, "Aproximation capabilities of multilayer feedforward networks", *Neural Networks 4 (1991) 251-257.*
7. M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function", *Neural Networks 6 (1993) 861-867.*
8. G.-B. Huang, H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions", *IEEE Trans. Neural Networks 9 (1) (1998) 224-229.*
9. S. Tamura, M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three", *IEEE Trans. Neural Networks 8 (2) (1997) 251-255.*
10. G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks", *IEEE Trans. Neural Networks 14 (2) (2003) 274-281.*
11. P. L. Barlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more imprortant tan the size os the network", *IEEE Trans. Inf. Theory 44 (2) (1998) 525-536.*
12. I. Guyon, G. Dror, V. Lemaire, G. Taylor and D. Silver, "Autoencoders, Unsupervised Learning, and Deep Architectures", *Pierre Baldi. Department of Computer Science. University of California. Irvine, CA 92697-3435.*
13. M Yasmin, M. Sharif and S. Mohsin, "Neural Networks in Medical Imaging Applications: A Survey", *Department of Computer Science, COMSATS Institute of Information Technology, Pakistan. World Applied Sciences Journal 22 (1): 85-96, 2013.*
14. Bastida-Jumilla MC, Menchón-Lara RM, Morales-Sánchez J, Verdú-Monedero R, Larrey-Ruiz J, Sancho-Gómez JL, "Segmentation of the Common Carotid Artery Walls Based on a Frequency Implementation of Active Contours", *J Digit Imaging (2013) 26: 129-139.*

15. F. Izaurieta, C. Saavedra, “Redes Neuronales Artificiales”, *Departamento de Física, Universidad de Concepción, Concepción, Chile.*
16. X. Basogain Olabe, “Redes Neuronales Artificiales y sus Aplicaciones”, *Dpto. Ingeniería de Sistemas y Automática, Escuela Superior e Ingeniería de Bilbao. UPV-EHU.*
17. M. Gestal Pose, “Introducción a las Redes de Neuronas Artificiales”, *Dpto. Tecnologías de la Información y las Comunicaciones. Universidade da Coruña.*
18. M. Gallardo Campos, A. J. de Castro González, E. García Cuesta, “Aplicación de Técnicas de Clustering para la Mejora del Aprendizaje”, *Universidad Carlos III de Madrid. Escuela Politécnica Superior- Leganés (2009).*
19. M. J. L. Orr, “Introduction to Radial Basis Function Networks”, *Centre for Cognitive Science, University of Edinburgh, Scotland (1996).*
20. G.-B. Huang, D Wang, Y. Lan, “Extreme Learning Machines: A Survey”, *School of Electrical and Electronic Engineering (Singapore), Department of Computer Science and Computer Engineering (Australia).*
21. C. Cortes, V. Vapnik, “Support vector networks”, *Machine Learning vol. 20, no. 3, pp. 273-297, 1995.*
22. D. Lowe, “Adaptative radial basis function nonlinearities and the problema of generalisation”, *in Proceedings of First IEE International Conference on Artificial Neural Networks, pp. 171-175, 1989.*
23. G.-B. Huang, L. Chen, “Enhanced random search based incremental extreme learning machine”, *Neurocomputing, vol. 71, pp. 3460-3468, 2008.*
24. G.-B. Huang, L. Chen, C. K. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes”, *IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879-892, 2006.*
25. G.-B. Huang, H. Zhou, X. Ding, R. Zhang, “Extreme Learning Machine for regression and multi-class classification”, *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, October 2010.*
26. W. Deng, Q. Zheng, L. Chen, “Regularized extreme learning machine”, *in IEEE Symposium on computational Intelligence and Data Mining (CIDM2009), pp. 389-395, March 30 2009-April 2 2009.*
27. L. K. Hansen, P. Salamon, “Neural network ensemble”, *IEEE Transaction Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, 1990.*
28. Z.-L. Sun, T.-M. Choi, K.-F. Au, Y. Yu, “Sales forecasting unsing extreme learning machine with applications in fashion retailing”, *Decision Support Systems, vol. 46, no. 1, pp. 411-419, 2008.*
29. M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. A. Hilbers, T. Honkela, E. Oja, A. Lendasse, “Adaptative ensembles models of extreme learning machines for time series prediction”, *Lecture Notes in Computer Science, vol. 5769, pp. 305-314, 2009.*
30. Y. Lan, Y. C. Soh, G.-B. Huang, “Ensemble of online sequential extreme learning machine”, *Neurocomputing, vol. 72, pp. 3391-3395, 2009.*
31. G. E. Hinton, R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, *Science, vol. 313, no. 5786, pp. 504-507, 2006.*
32. E. Cambria, G.-B. Huang, L. L. Chamara Kasun, H. Zhou, V. Chi Man, L. Jiarun, Y. Jianping, C. Zhiping, L. Qiang, L. Kuan, Victor C.M, F. Liang, O. Yew-Soon, L. Meng-Hiot, A. Anton, L. Amaury, F. Corona, N. Rui, M. Yoan, P. Gastaldo, R. Zunino, S.

Decherchi, Y. Xuefeng, M. Kezhi, O. Beom-Seok, J. Jehyoung, T. Kar-Ann, T. Andrew Beng Jin, K. Jaihie, Y. Hanchao, C. Yiqiang, L. Junfa, "Extreme Learning Machines [Trends & Controversies]," *Intelligent Systems, IEEE , vol.28, no.6, pp.30,59, Nov.-Dec.2013. doi: 10.1109/MIS.2013.140.*

33. R. L. Burden, J. Douglas Faires, "Análisis Numérico", *Grupo Editorial Iberoamericana, México, 1985.*
34. J. Demmel, "Applied Numerical Linear Algebra", *SIAM Philadelphia, 1997.*
35. Menchón-Lara RM, Bastida-Jumilla MC, Morales-Sánchez J, Sancho-Gómez JL, "Segmentación de Imágenes Mediante Redes Neuronales para el Análisis de Señales Acústicas Emitidas por Cetáceos", *Dpto. Tecnologías de la Información y las Comunicaciones Universidad Politécnica de Cartagena.*
36. F. Molinari, G. Zeng, J. S. Suri, "A state of the art review on intima-media thickness (IMT) measurement and wall segmentation techniques for carotid ultrasound", *Computer Methods and Programs in Biomedicine 100 (2010) 201-221.*
37. Menchón-Lara RM, Bastida-Jumilla MC, Morales-Sánchez J, Sancho-Gómez JL, "Automatic detection of the intima-media thickness in ultrasound images of the common carotid artery using neural networks", *Med. Biol. Eng. Comput. (2014) 52: 169-181.*
38. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-Based Learning Applied to Document Recognition", *Proc. Of the IEEE, November 1998.*
39. I. Zaballa, "Valores Singulares. ¿Qué son? ¿Para qué sirven?", *Departamento de Matemática Aplicada y EIO Universidad del País Vasco.*