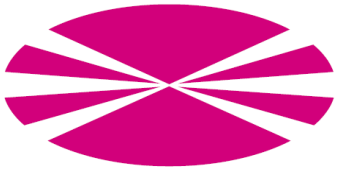


UNIVERSIDADE DA CORUÑA

FACULTAD DE INFORMÁTICA

Departamento de Computación

TESIS DOCTORAL

ALGORITMOS EFICIENTES, INCREMENTALES Y ESCALABLES PARA EL APRENDIZAJE EN REDES DE NEURONAS ARTIFICIALES

Autora: Beatriz Pérez Sánchez

Directores: Óscar Fontenla Romero
Bertha Guijarro Berdiñas

A Coruña, Octubre 2010

20 de octubre de 2010
UNIVERSIDAD DE A CORUÑA

FACULTAD DE INFORMÁTICA
Campus de Elviña s/n
15071 - A Coruña (España)

A mis padres

Dña. Bertha Guijarro Berdiñas, Profesora Titular del Departamento de Computación de la Facultad de Informática de la Universidade da Coruña

y

D. Óscar Fontenla Romero, Profesor Titular del Departamento de Computación de la Facultad de Informática de la Universidade da Coruña

CERTIFICAN que:

La memoria que se presenta, titulada *Algoritmos eficientes, incrementales y escalables para el aprendizaje en redes de neuronas artificiales* ha sido realizada por Dña. Beatriz Pérez Sánchez bajo nuestra dirección en el Departamento de Computación de la Universidade da Coruña y constituye la Tesis que presenta para optar al grado de Doctor.

Y para que así conste firmamos la presente en A Coruña a 6 de Octubre de 2010.

Fdo.: Bertha Guijarro Berdiñas. Fdo.: Óscar Fontenla Romero.

Agradecimientos

- A Bertha Guijarro Berdiñas y Óscar Fontenla Romero por darme la oportunidad de realizar esta Tesis Doctoral. Agradecer también su dedicación y apoyo incondicional, día tras día, durante todos estos años.
- A Enrique Castillo Ron y Cristina Solares por su acogida y su extrema generosidad.
- A todos mis compañeros de Valladolid por hacerme sentir como en casa desde el primer día.
- A Amparo Alonso Betanzos y Vicente Moret Bonillo por ofrecerme la posibilidad de formar parte de un extraordinario grupo de investigación.
- A Noelia, Elena y Mariano por su tiempo, sus consejos, y el inmenso apoyo que me han regalado.
- A todos y cada uno de los miembros del laboratorio LIDIA por compartir cada día durante todos estos años.
- A mis padres por su cariño, sus consejos, su preocupación constante y por supuesto por su enorme entrega y sacrificio.

- A mi abuela y a toda mi familia. A todos y cada uno, gracias por vuestro apoyo incondicional durante la realización de este trabajo.
- A mis tres duendes de la suerte Andrea, Emma y Alba por regalarme su cariño y sus inmensas sonrisas.
- A Javi por su dedicación, confianza, paciencia y optimismo. En definitiva, por ser cada día el motor de mi esperanza.
- A mis amigos por su apoyo infinito. Gracias por vuestra preocupación constante.
- Para todos aquellos que directa o indirectamente han conseguido que este trabajo sea posible. Gracias a todos.

Beatriz Pérez Sánchez

Octubre de 2010

Resumen

Este trabajo se centra en el desarrollo de nuevos modelos de aprendizaje supervisado para redes de neuronas artificiales alimentadas hacia delante. En primer lugar, se plantean mejoras sobre métodos de aprendizaje ya desarrollados, con el objeto de generalizar su comportamiento a nuevas situaciones manteniendo sus características originales. En concreto, se plantea el empleo de la regularización para manejar situaciones donde es posible el fenómeno de sobreajuste a los datos. En segundo lugar, se desarrollan algoritmos de aprendizaje *online* tanto para redes de una capa como de dos, que permiten su aplicación en entornos no estacionarios en los que el proceso a modelar no permanece inalterable. Además, para el caso de las redes de dos capas, este algoritmo *online* permitirá que también la topología de la red se adapte de manera automática según las necesidades del aprendizaje, añadiendo unidades ocultas únicamente en caso necesario. En todo momento se persigue un aprovechamiento de los recursos disponibles. Para los algoritmos propuestos se incluye una descripción teórica de sus capacidades, y su comportamiento se ilustra mediante su aplicación a casos concretos y significativos. Finalmente se analizan los resultados obtenidos, extrayendo las principales conclusiones del comportamiento de cada método, capacidades y limitaciones para su aplicación futura.

Palabras clave: *redes de neuronas artificiales, aprendizaje supervisado, regularización, aprendizaje online, aprendizaje incremental, cambio en el concepto a aprender, topología adaptativa.*

Resumo

Este traballo céntrase no desenvolvemento de novos modelos de aprendizaxe supervisada para redes de neuronas artificiais alimentadas cara a adiante. En primeiro lugar, expóñense melloras sobre métodos de aprendizaxe xa desenvolvidos, co obxecto de xeneralizar o seu comportamento a novas situacións mantendo as súas características orixinais. En concreto, móstrase o emprego da regularización para manexar situacións onde é posible o fenómeno de sobreaxuste aos datos. En segundo lugar, desenvólvense algoritmos de aprendizaxe online tanto para redes dunha capa como de dúas, que permiten a súa aplicación en contornas non estacionarias en que o proceso que se vai modelar non permanece inalterable. Alén diso, para o caso das redes de dúas capas, este algoritmo online permitirá que tamén a topoloxía da rede se adapte de xeito automático segundo as necesidades da aprendizaxe, engadindo unidades ocultas unicamente en caso de que sexa necesario. En todo momento perséguese un aproveitamento dos recursos dispoñibles. Para os algoritmos propostos, inclúese unha descrición teórica das súas capacidades e o seu comportamento ilústrase mediante a súa aplicación a casos concretos e significativos. Finalmente, analízanse os resultados obtidos, extraendo as principais conclusións do comportamento de cada método e as súas capacidades e limitacións para a súa aplicación futura.

Palabras clave: *redes de neuronas artificiais, aprendizaxe supervisada, regularización, aprendizaxe online, aprendizaxe incremental, cambio no concepto a aprender, topoloxía adaptativa*

Summary

The core of this work is the development of new supervised learning methods for feedforward neural networks. In the first place, improvements on already developed learning methods are presented in order to generalize their behaviour to new situations while keeping their original characteristics. In particular, the employment of the regularization technique is proposed to handle situations where the overfitting to data is possible. Secondly, we develop online learning algorithms for one and two layer neural networks so as to allow their application in stationary and non stationary contexts in which the process to be modelled does not remain unalterable. In addition, for the case of two layers neural networks, this algorithm will also automatically adapt the topology of the network according to the needs of learning, by adding new hidden units only when necessary. In all cases, an optimum employment of the computational resources is pursued. For all the proposed algorithms, theoretical descriptions of their capacities are included, and their behaviours are illustrated by means of their application to significant cases. Finally, we analyze the results obtained, extracting the main conclusions about each method, their capacities and limitations for their future application.

Palabras clave: *artificial neural networks, supervised learning, regularization, online learning, incremental learning, concept drift, adaptive topology*

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción

Gracias a los avances en tecnología de computadores, hoy en día es posible almacenar y procesar grandes cantidades de datos, así como acceder a ellos desde diferentes localizaciones. El análisis automático de estos datos almacenados resulta de especial utilidad si permite extraer información sobre los mismos por medio de la identificación de patrones de comportamiento. De este modo se facilita la construcción de una aproximación del modelo de comportamiento de los datos. Los patrones ayudan a entender el procedimiento que subyace a los datos analizados e incluso se pueden aplicar para realizar predicciones, siempre y cuando se asuma que las condiciones en un escenario futuro serán similares a las actuales [6]. Existen diversas disciplinas que intentan abordar este problema. Entre ellas cabe destacar la estadística, el aprendizaje automático o la minería de datos.

El *aprendizaje máquina o automático* es una disciplina de la Inteligencia Artificial cuya finalidad es el desarrollo de técnicas que permitan dotar de capacidad de aprendizaje a los sistemas informáticos empleados en un ordenador. De forma más concreta, se trata de crear algoritmos y programas capaces de generalizar patrones de comportamiento a partir de una información no estructurada suministrada en forma de ejemplos. Es, por tanto, un proceso de inducción del conocimiento que trata de extraer, a partir de

determinadas observaciones particulares, el principio general que en ellas está implícito. Estas técnicas permiten solucionar problemas que no son abordables de forma eficaz desde la óptica de las herramientas clásicas de la computación.

Idealmente, un sistema inteligente debe disponer de capacidad para aprender. El modelo de aprendizaje puede ser predictivo si predice datos futuros o descriptivo si obtiene información a partir de sus entradas. El aprendizaje persigue optimizar un criterio de rendimiento empleando los datos disponibles o incluso la experiencia previa, es decir, utiliza la estadística para construir modelos matemáticos que expliquen los datos disponibles, ya que realmente el proceso se reduce a realizar inferencias a partir de un ejemplo dado. El aprendizaje se divide en dos fases, entrenamiento y prueba. En el proceso de entrenamiento se determinan los pesos (parámetros) que definen el modelo empleando un conjunto de patrones mientras, en la fase de prueba, se utilizan nuevos ejemplos con el objeto de comprobar la eficacia del sistema generado. En cuanto a las condiciones de trabajo de los algoritmos de aprendizaje cabe mencionar que, en primer lugar, para el entrenamiento se necesitan algoritmos capaces de almacenar y procesar enormes cantidades de información, así como de resolver el problema de optimización del criterio de rendimiento de manera eficiente. En segundo lugar, una vez aprendido el modelo, su representación y solución han de ser eficientes.

Las técnicas de aprendizaje más representativas [6] se pueden clasificar en:

- Técnicas computacionales puras como árboles de decisión, clasificación del vecino más cercano, agrupamiento basado en teoría de grafos o reglas de asociación.
- Técnicas estadísticas como regresión multivariable, discriminación lineal, teoría de decisión bayesiana, redes bayesianas o K-medias.
- Técnicas mixtas, computacionales-estadísticas, tales como máquinas de vectores soporte o *adaboost*.
- Técnicas bio-inspiradas como redes neuronales, algoritmos genéticos o sistemas inmunológicos artificiales.

El trabajo desarrollado en la presente Tesis se encuadra dentro de las dos últimas aproximaciones mencionadas, debido a que utiliza técnicas computacionales y estadísticas y, además, emplea modelos basados en redes de neuronas artificiales.

1.1. Técnicas de aprendizaje inspiradas en la neurociencia

Antes de la aparición de las técnicas de aprendizaje inspiradas en neurociencia, los métodos y herramientas de computación estándar, empleados para el procesado de información, tenían las siguientes características comunes:

1. El conocimiento se representaba explícitamente empleando reglas, redes semánticas, modelos probabilísticos, etc.
2. Se imitaba el proceso humano de razonamiento lógico para resolver los problemas, centrando la atención en las causas que intervienen en el problema y en sus relaciones.
3. La información se procesaba de modo secuencial.

Sin embargo, aparecieron un gran número de problemas complejos para los que una representación explícita del conocimiento no era conveniente y no se disponía de un procedimiento de razonamiento lógico para resolverlos. Así, las aproximaciones algorítmicas y las estructuras computacionales estándar no eran apropiadas para resolver estos problemas. Las redes de neuronas artificiales [59] se introducen como estructuras de computación alternativas, creadas con el propósito de reproducir las funciones del cerebro humano. Las neuronas naturales (véase figura 1.1) reciben señales electroquímicas de otras neuronas a través de las uniones sinápticas que conectan el axón de las neuronas emisoras y las dendritas de las receptoras. Basándose en las informaciones o impulsos recibidos, la neurona computa y envía su propia señal. El potencial interno asociado a una neurona es el que controla el proceso de emisión. Si este potencial supera

un cierto umbral, se envía un impulso eléctrico al axón, en caso contrario no se envía la señal.

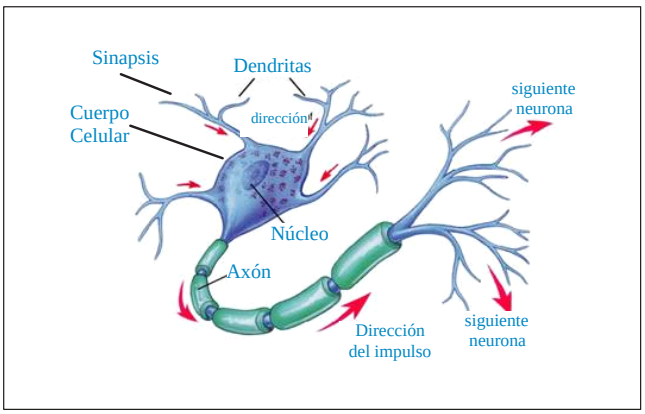


Figura 1.1: Descripción de una célula nerviosa típica.

Los modelos computacionales conocidos como redes neuronales están inspirados en las características neurofisiológicas anteriores y, por tanto, están formadas por un gran número de procesadores, o neuronas, dispuestos en varias capas e interconectados entre sí mediante conexiones con pesos. Estos procesadores realizan cálculos simples basados en la información que reciben de los procesadores vecinos. Las redes neuronales no siguen reglas programadas rígidamente, como hacen los computadores digitales más convencionales, más bien usan un proceso de aprendizaje por analogía donde los pesos de las conexiones se ajustan automáticamente para reproducir un conjunto de patrones representativo del problema a aprender. Este aprendizaje también está inspirado en la forma de aprender que tiene lugar en las neuronas, cambiando la efectividad de las sinapsis, de tal manera que la influencia de una neurona en otra varía. Es importante mencionar que las arquitecturas computacionales habituales de redes neuronales son extremadamente simplificadas cuando se analizan desde un punto de vista neurofisiológico; sin embargo, estos modelos tan simples permiten resolver muchos problemas interesantes.

1.2. Breve reseña histórica de las Redes de Neuronas Artificiales

Las redes de neuronas artificiales han sido reconocidas como una interesante herramienta para aprender y reproducir sistemas en varios campos de aplicación. Están inspiradas en el comportamiento del cerebro y consisten en una o varias capas de neuronas (o unidades de cálculo) unidas mediante conexiones. A través de estas conexiones cada neurona artificial recibe entradas de las neuronas de otras capas y produce una salida escalar mediante la combinación de los valores recibidos usando una función de activación.

Las redes neuronales no siguen reglas programadas rígidamente. Emplean un proceso de aprendizaje por analogía donde los pesos de las conexiones se ajustan automáticamente para reproducir un conjunto de patrones representativo del problema a aprender. Este aprendizaje también está inspirado en la forma de aprender que tiene lugar en las neuronas, cambiando la efectividad de las sinapsis, de tal manera que la influencia de una neurona en otra varía. Una de las principales propiedades de las redes neuronales es su capacidad para aprender a partir de los datos. Hay dos tipos de aprendizaje: estructural y paramétrico. El aprendizaje estructural consiste en aprender la topología de la red: número de capas, número de neuronas en cada una de ellas y las conexiones necesarias. En general, este proceso se realiza mediante prueba y error hasta obtener un buen ajuste a los datos. A su vez, el aprendizaje paramétrico obtiene los valores óptimos de los pesos para una topología dada de la red. Como las funciones neuronales son conocidas, este proceso de aprendizaje se consigue estimando los pesos asociados a cada conexión a partir de la información dada a la entrada de las neuronas. Con este propósito se minimiza una función de error empleando métodos de aprendizaje conocidos, tales como el algoritmo de retropropagación del error.

Los orígenes de las redes de neuronas artificiales (RNA) parten del descubrimiento en 1906 de Ramón y Cajal de las neuronas como células independientes en cuanto

a su función, estructura y origen. Estos estudios llevaron posteriormente a diversos neurólogos, entre los que destaca Hebb a realizar diversas especulaciones fisiológicas [61]. Sin embargo, los pioneros en el área de las RNA fueron McCulloch y Pitts [97] al formalizar en 1943 el funcionamiento de una neurona, basándose en la idea de que las neuronas operan mediante impulsos binarios.

La neurona artificial, célula o autómata, es un elemento que posee un estado interno, llamado nivel de activación, y recibe señales que le permiten, en su caso, cambiar de estado. Para ello, las neuronas poseen una función que les permite modificar de nivel de activación a partir de las señales que reciben; a dicha función se le denomina función de transición de estado o función de activación. Las señales que recibe cada neurona pueden provenir del exterior o de otras neuronas a las cuales está conectada. Para obtener el estado de activación se ha de calcular, en primer lugar, la entrada total a la célula. Este valor viene dado por la suma de todas las entradas ponderadas por ciertos valores llamados pesos sinápticos. De manera que, el nivel de activación de una célula depende de las entradas recibidas y de los valores sinápticos, pero no de anteriores valores de estados de activación.

La figura 1.2 muestra un modelo que representa esta idea. Se introduce un grupo de entradas $x_1, x_2, \ldots, x_n$ en una neurona artificial. Estas entradas, definidas por un vector $\mathbf{x}$, corresponden a las señales de la sinapsis de una neurona biológica. Cada señal se multiplica por un peso asociado $w_{j1}, w_{j2}, \ldots, w_{jn}$ antes de ser aplicado el sumatorio.

Cada peso corresponde a la fuerza de una conexión sináptica, es decir, el nivel de concentración iónica de la sinapsis, y se representa por un vector $\mathbf{w}_j$. El sumatorio, que corresponde al cuerpo de la neurona, suma todas las entradas ponderadas algebraicamente, produciendo una salida que se denomina u_j, así

$$u_j = x_1 w_{j1} + x_2 w_{j2} + \ldots + x_n w_{jn}$$

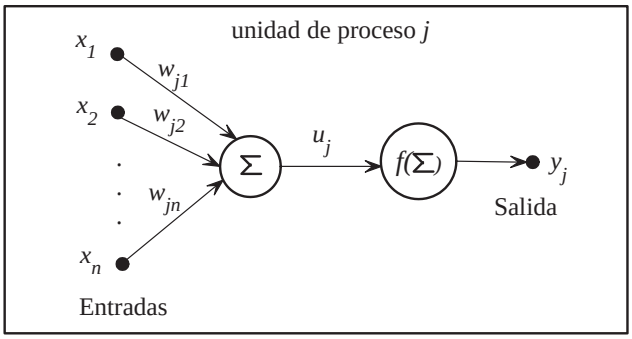


Figura 1.2: Esquema de la neurona artificial típica.

Las señales u son procesadas además por una función de paso por umbral, llamada función de activación o de salida f, que produce la señal de salida y_j de la neurona.

En 1957, Frank Rosenblatt [122, 123] generalizó el modelo de células de McCulloch y Pitts añadiéndole aprendizaje y llamó a este modelo Perceptrón Simple, una subclase de red neuronal. Este modelo se concibió como un sistema capaz de realizar tareas de clasificación de forma automática. La idea era disponer de un sistema que, a partir de un conjunto de ejemplos de clases diferentes, fuera capaz de determinar las ecuaciones de las superficies que hacían de frontera de dichas clases (véase figura 1.3). La información sobre la que se basaba el sistema estaba constituida por los ejemplos existentes de las diferentes clases. Los patrones de entrenamiento aportaban información necesaria para que el sistema construyera las superficies discriminantes y además actuara como un discriminador para ejemplos nuevos y desconocidos. La arquitectura de la red es muy simple. Se trata de una estructura monocapa, en la que hay un conjunto de entradas (véase figura 1.4), tantas como sea necesario según los términos del problema; y una o varias células de salida. Cada una de las entradas tiene conexiones con todas las células de salida, y son estas conexiones las que determinan las superficies de discriminación del sistema. En el ejemplo de la figura 1.4 las entradas vienen dadas por x_1, x_2, la salida es y, y w_1 y w_2 hacen referencia a los pesos. Además existe un parámetro adicional llamado umbral y denotado por w_0. Este umbral se emplea como factor de comparación para producir la salida, y habrá tantos como células de salida existan en la red.

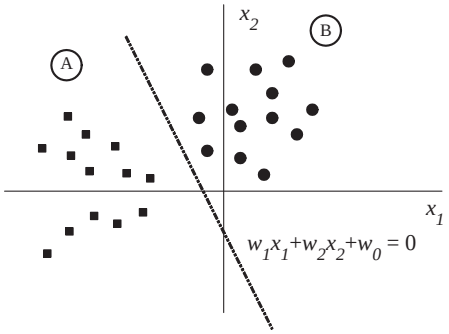


Figura 1.3: Ejemplo de la separación de dos clases mediante un Perceptrón.

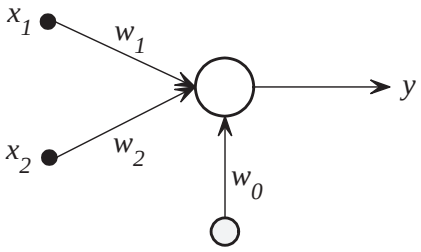


Figura 1.4: Ejemplo de la arquitectura de un Perceptrón con dos entradas y una salida.

Dos años después, Bernard Widrow [148, 149] diseñó una red artificial muy similar al perceptrón, denominada *Adaptive Linear Elements* o Adaline. El Adaline de dos niveles es muy parecido al Perceptrón. La diferencia entre los dos modelos es muy pequeña, pero las aplicaciones a las que van dirigidas difieren de manera considerable. En 1960, Widrow y Hoff [150] probaron matemáticamente que en determinadas circunstancias el error entre la salida deseada para la red y la obtenida por ella ante una entrada determinada podía ser minimizado hasta el límite que se desee. A pesar de esta importante investigación, tanto el Perceptrón como el Adaline mantienen el problema de la separabilidad lineal, y la exposición matemática de la naturaleza de los perceptrones elaborada por Minsky y Papert en 1969 [99] que hacía referencia a este problema puso en claro el alcance y las limitaciones de estos novedosos modelos de proceso en el campo de la Inteligencia Artificial.

En concreto, en lo referente al problema de la separabilidad no lineal, en ese mismo trabajo, Minsky y Papert mostraron que la combinación de varios perceptrones simples (inclusión de neuronas ocultas en lo que llamarán Perceptrón Multicapa), podía resultar la solución adecuada. Sin embargo, no presentaron una solución al problema de cómo adaptar los pesos de la capa de entrada a la capa oculta, pues la regla de aprendizaje del perceptrón simple no puede aplicarse en este escenario. Esta crítica tuvo una amplia repercusión y determinó en gran medida el decaimiento de la rama conexionista en la década de los setenta. No obstante, la idea de combinar varios perceptrones sirvió de base para estudios posteriores realizados por Rumelhart, Hinton y Willians en 1986 [124]. Estos autores presentaron el conocido algoritmo de retropropagación del error (*bakcpropagation*), basado en retropropagar los errores medidos en la salida de la red hacia las neuronas ocultas para funciones de activación no lineales y redes multicapa. Actualmente dentro del marco de las redes de neuronas, el perceptrón multicapa es una de las arquitecturas más utilizadas en la resolución de problemas debido a su capacidad de aproximador universal, su fácil uso y su aplicabilidad.

La estructura básica de interconexión entre células en un perceptrón es la mostrada en la figura 1.5. El primer nivel lo constituyen las entradas que reciben los valores de unos patrones representados como vectores que sirven de entrada a la red. A continuación hay una serie de capas intermedias, llamadas ocultas, cuyas células responden a rasgos particulares que pueden aparecer en los patrones de entrada y que, a su vez, están conectadas con todas las células de la capa siguiente. Una característica salientable es que la función de activación de la neurona (véase figura 1.2) es no lineal. Cada célula de la red, una vez recibida la totalidad de sus entradas, las procesa y, de acuerdo al proceso explicado para la neurona de McCulloch y Pitts, genera una salida que es propagada a través de las conexiones de las células, llegando como entrada a la célula destino. Los valores que se propagan se evalúan por los pesos de las conexiones, los cuales se ajustan durante la fase de aprendizaje para producir una red de neuronas final. Una vez que la entrada ha sido completamente propagada por toda la red, se producirá un vector de salida y se obtendrá el error asociado. Así, una red de neuronas

podría definirse como un grafo cuyos nodos están constituidos por unidades de proceso idénticas, y que propagan información a través de los arcos. Además, al aplicar el método de retropropragación del error, cada neurona de salida distribuye hacia atrás su error a todas la neuronas ocultas que se conectan a ella, ponderado por el valor de la conexión. De este modo, cada neurona oculta recibe un cierto error de cada neurona de salida, cuya suma es el error asociado a la neurona oculta. Dichos valores permiten, a su vez, obtener los valores de las neuronas ocultas de la capa anterior, y así sucesivamente hasta llegar a la primera capa oculta. De ahí viene el nombre de retropropagación pues los errores para las neuronas de la red se propagan hacia todas las neuronas de la capa anterior. Empleando estos errores, se modifican los pesos y sesgos de la red para la capa de salida y para el resto de los parámetros de la red.

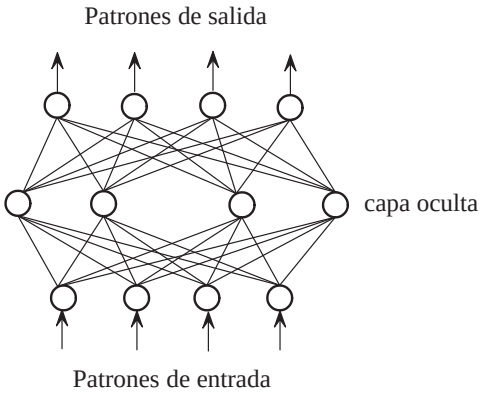


Figura 1.5: Esquema de un Perceptrón Multicapa de tres capas.

Posteriormente, James Anderson trabajó con un modelo de memoria basado en la asociación de las activaciones de la sinapsis de una neurona [7], y realizó un modelo de memoria asociativa lineal [8]. Empleó un nuevo método de corrección de error, y sustituyó la función umbral lineal por otra en rampa, creando un nuevo modelo llamado *Brain-state-in-a-box* [9]. Kohonen comenzó sus investigaciones sobre RNA con paradigmas de conexiones aleatorias en 1971. Los trabajos de Kohonen [75, 76] se centraron en memorias asociativas, de manera semejante a los trabajos de Anderson. El premio nobel Leon Cooper y su colega Charles Elbaum comenzaron a trabajar con

RNA a principios de los 70 [37, 38]. Formaron un grupo para el desarrollo, la patente y la explotación comercial de RNA, que tuvo bastante éxito en el desarrollo comercial de algunos sistemas de red neuronal.

Sejnowski es uno de los pocos investigadores que trabajaron con modelos matemáticos y biológicos. Una de sus contribuciones más importantes en este campo es el descubrimiento, junto con Geoff Hinton, del algoritmo de la máquina de Boltzmann [62]. Este método fue aplicado principalmente a distintas áreas de visión artificial [127]. Más recientemente son conocidas sus contribuciones a la aplicación del algoritmo de retropropagación, sobre todo para el reconocimiento de la voz.

En 1982, John Hopfield describió un método de análisis del estado estable en una red autoasociativa [66]. Introdujo una función de energía en sus estudios sobre sistemas de ecuaciones no lineales. Hopfield demuestra que se puede construir una ecuación de energía que describa la actividad de una red neuronal monocapa en tiempo discreto, y que esta ecuación de energía pueda ir disipándose y el sistema converger a un valor mínimo local. Este análisis hizo resurgir el interés por aplicar los paradigmas de RNA para resolver problemas difíciles que los métodos convencionales no pueden solucionar. Mas adelante, Hopfield extendió su modelo para considerar tiempos continuos [67].

Es importante mencionar que las arquitecturas computacionales habituales de redes neuronales son extremadamente simplificadas cuando se analizan desde un punto de vista neurofisiológico, sin embargo, estos modelos tan simples permiten resolver muchos problemas interesantes. Actualmente las redes de neuronas han probado su valía para resolver problemas complejos, que a primera vista parecen intratables y son difíciles de formular usando técnicas de computación convencionales. Ejemplos de tales problemas se pueden encontrar en gran variedad de campos tales como el reconocimiento de patrones [21, 120], el reconocimiento de imágenes y del habla [4, 128], predicción y pronóstico de series temporales [13, 102], control de procesos [98], procesamiento de señales [34], etc.

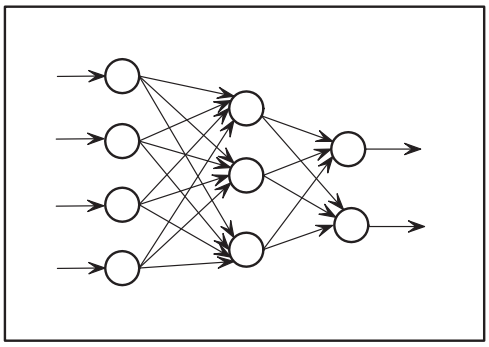
1.3. Arquitecturas básicas de las Redes de Neuronas Artificiales

Las neuronas que forman parte de una RNA se pueden organizar en capas conectadas por varios tipos de uniones incluyendo conexiones hacia delante, laterales, y hacia atrás:

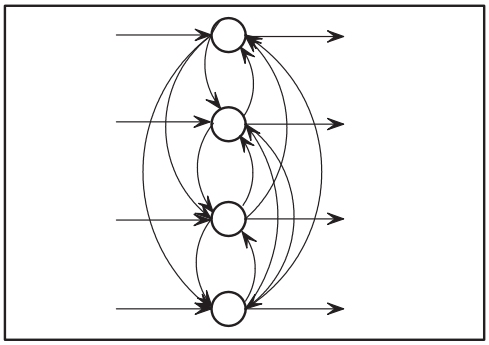
- Conexiones hacia delante: Conectan neuronas de una capa con neuronas de la capa siguiente (véase figura 1.6(a)).
- Conexiones laterales: Conectan neuronas de la misma capa (véase figura 1.6(b)).
- Conexiones hacia atrás (o recurrentes): Crean bucles en las neuronas de la red de manera que pueden considerarse conexiones de una neurona con ella misma, conexiones entre neuronas de una misma capa o conexiones de las neuronas de una capa a la capa anterior (véase figura 1.6(c)). Este tipo de conexiones permiten a las redes tratar modelos dinámicos y temporales, es decir, modelos con memoria.

Las redes de neuronas pueden clasificarse empleando como criterio la topología de la red es decir, considerando su número de capas, número de neuronas por capa, grado de conectividad y tipo de conexiones entre neuronas. Así se establecen los siguientes tipos:

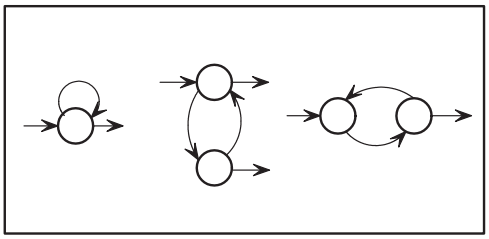
- Redes con alimentación hacia delante. Los datos fluyen en un único sentido desde las entradas a las salidas. Las salidas de las neuronas de una capa se conectan sólo con las entradas de las neuronas de la siguiente capa. Por tanto, no es posible que la salida de una neurona esté conectada con la entrada de alguna neurona de la misma capa o de capas previas. Este es el modelo típico explicado previamente en la figura 1.5.
- Redes recurrentes. Se diferencian de las anteriores en la existencia de lazos de realimentación en la red. Estos lazos pueden ser entre neuronas de diferentes



(a)



(b)



(c)

Figura 1.6: Ejemplos de tipos de conexiones entre neuronas: (a) Red multicapa con conexiones hacia delante, (b) Red con cuatro nodos y conexiones laterales dentro de las neuronas de la misma capa, (c) Ejemplos de neuronas con conexiones recurrentes.

capas, neuronas de la misma capa o de una neurona consigo misma (véanse figuras 1.6(b) y 1.6(c)).

1.4. Características del aprendizaje en RNA

En una red de neuronas artificiales es necesario definir un procedimiento por el cual las conexiones o pesos del sistema varían para reproducir la salida deseada es decir, se busca obtener los valores precisos de los pesos de todas las conexiones con el objeto de resolver un problema de manera eficiente. El proceso general de aprendizaje consiste en ir facilitando a la red los ejemplos de un conjunto de aprendizaje representativo del problema a resolver y modificando los pesos de sus conexiones en base a un determinado esquema de aprendizaje. Este esquema es el que determina el tipo de problemas que la red será capaz de resolver. Por otro lado, el aprendizaje en una RNA está basado en el uso de ejemplos, por lo que la capacidad de una red para resolver un problema estará ligada de forma fundamental al tipo y representatividad de los ejemplos de que dispone en el proceso de aprendizaje. Desde el punto de vista de los ejemplos, el conjunto de aprendizaje debe poseer las siguientes características [138]:

- Ser significativo. Debe haber un número suficiente de ejemplos ya que si el conjunto de aprendizaje es reducido, la red no será capaz de adaptar sus pesos de forma eficaz.
- Ser representativo. Los componentes del conjunto de aprendizaje deberán ser diversos. Es importante que todas las regiones significativas del espacio de estados estén suficientemente representadas en el conjunto de aprendizaje.

Por otro lado, se distinguen tres tipos principales de esquemas de aprendizaje [41]:

- *Aprendizaje Supervisado.* En este caso, los patrones del conjunto de entrenamiento para el aprendizaje están formados por parejas que constan de un vector de variables de entrada junto a sus correspondientes salidas, es decir, la respuesta deseada a las señales de entrada. Los pesos se obtienen minimizando alguna función de error que mide la diferencia entre los valores de salida deseados y los

calculados por la red neuronal. La manera más habitual de modificar los valores de los pesos de las conexiones es la representada en la figura 1.7(a). Cada vez que se introduce un ejemplo y se procesa para obtener una salida, ésta se compara con la salida que debería haber producido la red según el conjunto de aprendizaje. La diferencia entre ambas influirá en cómo se modifican los pesos.

- *Aprendizaje no supervisado.* En este caso, los datos se presentan sin información externa sobre la salida a obtener y la red tiene que descubrir patrones o categorías dentro de ellas. Este tipo de aprendizaje se encuentra dentro de las técnicas autoorganizativas, o técnicas automáticas para descubrir la estructura de los datos, debido a que la red se ajusta dependiendo únicamente de los valores recibidos como entrada. Los datos del aprendizaje no supervisado pueden contener valores de entrada y valores de salida, pero no hay información sobre qué salidas corresponden a cada una de las entradas de datos. En la figura 1.7(b) se representa este tipo de aprendizaje. La red modificará los valores de los pesos a partir de información interna para tratar de determinar características, rasgos significativos, regularidades o redundancias de los datos del conjunto de entrenamiento.

- *Aprendizaje por refuerzo.* Es una variante del aprendizaje supervisado en el que no se dispone de información concreta del error cometido por la red para cada ejemplo de aprendizaje, sino que simplemente se determina si la salida producida para dicho patrón es o no adecuada.

Normalmente, el proceso de aprendizaje es un proceso cíclico. La finalización de este proceso se realiza en función de un criterio de convergencia, que depende del tipo de red utilizada o del tipo de problema a resolver. Así, el período de aprendizaje finaliza cuando se cumple uno o varios de los siguientes criterios,

- Se alcanza un número fijo de ciclos de aprendizaje. Se decide a priori cuántas veces se facilita el conjunto completo de datos a la red, y superado dicho número se detiene el proceso y se acepta la red resultante.

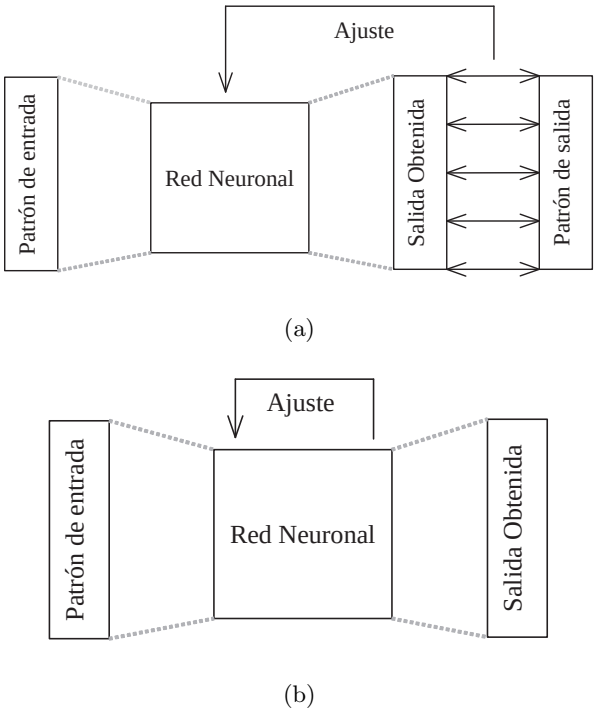


Figura 1.7: Esquemas de aprendizaje: (a) Aprendizaje supervisado, (b) Aprendizaje no supervisado.

- El error desciende de una cantidad preestablecida. En este caso habrá que definir en primer lugar una función de error, bien a nivel de patrón, bien a nivel de la totalidad del conjunto de entrenamiento. Se decide a priori un valor aceptable para dicho error, y sólo se para el proceso de aprendizaje cuando la red produzca un error por debajo del prefijado.

- Cuando la modificación de los pesos sea irrelevante. En algunos modelos se define un esquema de aprendizaje que hace que las conexiones vayan modificándose cada vez con menor intensidad. Llegará un momento, en el proceso de aprendizaje, en el que ya no se producirán variaciones en los valores de los pesos de ninguna conexión; la red alcanza la convergencia y el aprendizaje finaliza.

1.4.1. Problemas del algoritmo de aprendizaje por retropropagación del error. Variantes.

El trabajo desarrollado en esta Tesis Doctoral está encuadrado en el área de las redes de neuronas con conexiones o alimentación hacia delante y aprendizaje supervisado. Para este tipo de redes neuronales, existe un gran número de métodos de aprendizaje. Como se ha comentado, el algoritmo de retropropagación del error o *backpropagation* propuesto por Rumelhart [125] fue el primer algoritmo importante para redes multicapa con alimentación hacia delante. La idea principal de este algoritmo es modificar gradualmente los pesos de la red en la dirección que le indica el gradiente descendente de los pesos respecto a la función de error. A pesar de que este algoritmo es una aproximación muy útil para el aprendizaje en este tipo de redes plantea dos inconvenientes de importante consideración que se describen a continuación:

- *Lenta velocidad de convergencia.* De manera general, los algoritmos de aprendizaje basados en descenso de gradiente presentan una convergencia lenta hacia la solución. Este hecho es debido a que el algoritmo de aprendizaje avanza muy lentamente en aquellas regiones de la superficie de error en las que sus derivadas son prácticamente nulas. Esta situación es crítica en las mesetas de la función de error.

- *La convergencia hacia el mínimo global no está garantizada.* La minimización de la función de error es la que dirige el aprendizaje de la red. De manera general, los algoritmos de aprendizaje utilizan solamente información del estado actual para llevar a cabo tal minimización. Esto ocurre en el método de retropropagación del error. El problema que se plantea es consecuencia de los mínimos locales que puede presentar la función a optimizar. Si el estado inicial de la red, determinado por los valores iniciales asignados a los pesos, está en el área de atracción de un mínimo local de la función entonces el algoritmo se verá atraído por éste.

Con el objeto de resolver los problemas que se acaban de describir, a lo largo de los años se han planteado y desarrollado distintas alternativas. A continuación se presenta una selección de las soluciones presentes en la bibliografía que pretenden solventar los inconvenientes mencionados.

En primer lugar y con respecto al problema de la lenta velocidad de convergencia, se han propuesto diferentes soluciones inspiradas en distintas técnicas:

- *Modificaciones del algoritmo estándar.* Se han planteado varias modificaciones importantes del método clásico de Rumelhart. Ihm y Park [72] presentan un algoritmo de aprendizaje rápido y novedoso que pretende evitar la lenta convergencia debida a las oscilaciones de los pesos en las zonas de valles de la superficie de error. Para conseguirlo, derivan un nuevo término de gradiente al modificar el original mediante una estimación de la dirección del descenso en las zonas de valles. Además, el método estocástico (que actualiza los pesos en cada ciclo o iteración), puede disminuir con frecuencia el tiempo de convergencia, y es especialmente apropiado para trabajar con conjuntos de datos grandes en problemas de clasificación [84].

- *Mejoras en la función de error.* El algoritmo de retropropagación del error estándar emplea en la función de error la derivada de las funciones neuronales. Tal derivada tiene forma de campana de Gauss, hecho que puede originar en ocasiones una lenta velocidad de convergencia si la salida de una neurona es próxima a cero o uno, ya que en tales casos la derivada tiene un valor cercano a cero. Con el objeto de suprimir tal efecto sobre la señal de error, van Ooyen y Nienhuis [105] y Krzyzak et al. [78] emplearon un método basado en una función de error semejante a la entropía.

- *Métodos basados en mínimos cuadrados* (*Least-squares*). Se han planteado diferentes algoritmos basados en métodos de mínimos cuadrados para inicializar o entrenar redes de neuronas con alimentación hacia delante [19, 30, 33, 47, 113, 153]. La mayor parte de ellos están basados en la minimización de la media de los

cuadrados de los errores entre la salida de una neurona, antes de la función no lineal, y una modificación de la salida deseada, que corresponde con la inversa de dicha función no lineal aplicada sobre la salida deseada actual.

- *Métodos de segundo orden.* Se ha propuesto el uso de las segundas derivadas de la función de error para incrementar la velocidad de convergencia [16, 26, 108]. En concreto, se ha demostrado que en términos de velocidad de convergencia, estos métodos son más eficientes que aquellos basados exclusivamente en técnicas de descenso de gradiente con derivadas de primer orden [85]. De hecho, los métodos de segundo orden se encuentran entre los algoritmos de aprendizaje más rápidos. Como ejemplos relevantes de este tipo de métodos podemos mencionar Levenberg-Marquardt [57, 89, 93], quasi-Newton [21] y los algoritmos de gradiente conjugado [18, 100]. El método de Levenberg-Marquardt combina el gradiente y la aproximación de Gauss-Newton de la hessiana de la función del error en la propia regla de actualización de los pesos. La influencia de cada término viene determinada por un parámetro que se actualiza automáticamente. Los métodos quasi-Newton, al igual que el método de Newton, utilizan una aproximación local cuadrática de la función de error pero aplican una aproximación de la inversa de la matriz hessiana para actualizar los pesos gracias a la cual consiguen un menor coste computacional. Sin embargo, debido a las altas necesidades de memoria requeridas por estos métodos su aplicación no es factible en todos los casos, especialmente para redes de grandes dimensiones y grandes volúmenes de datos. A pesar de ello, ha habido diversos intentos para reducir el coste computacional gracias a aproximaciones estocásticas [84, 126].

- *Tamaño del paso de aprendizaje.* El método de retropropagación fija, al comienzo del proceso, el valor del paso de aprendizaje que determina la magnitud del cambio en los pesos de cada iteración del algoritmo. Algunos investigadores indican que en superficies de error complejas, un paso de aprendizaje constante durante todo el proceso no sería lo más apropiado. Así, se han propuesto diferentes métodos de tipo heurístico para llevar a cabo una adaptación del paso de aprendizaje de manera dinámica [70, 73, 139]. Entre ellos se encuentra el superSAB, un intere-

sante algoritmo propuesto por Tollenaere [134]. Este método es una estrategia de aceleración para el aprendizaje *backpropagation* del error que converge más rápido que el gradiente descendente con un valor óptimo del paso de aprendizaje. A mayores, en [146] se presenta un método que permite la autodeterminación del valor de este parámetro. Más recientemente, se presenta un algoritmo para el gradiente descendente estocástico que utiliza un esquema de momento adaptativo no lineal para optimizar el bajo ratio de convergencia [106]. También en [5], se plantea un método para adaptar el paso de aprendizaje en una optimización de gradiente estocástico que utiliza, para todos los parámetros, tamaños del paso de aprendizaje independientes y los adapta empleando valoraciones del procedimiento de optimización del gradiente.

- *Inicialización adecuada de los pesos.* El punto de partida del algoritmo, determinado por el conjunto inicial de pesos de la red, afecta a la velocidad de convergencia del mismo. De este modo se plantean diferentes aproximaciones para la inicialización de estos pesos. Nguyen y Widrow desarrollaron un método en el que se asigna a cada elemento de procesado de la capa oculta una parte aproximada del rango de la salida deseada [104]. Asimismo, Drago y Ridella [39] plantean una aproximación basada en una activación estadísticamente controlada. Esta técnica previene la saturación de las neuronas durante el proceso de adaptación, mediante la estimación de los valores máximos que deben tomar inicialmente los pesos [39].

- *Reescalado de variables.* El cálculo de la señal de error implica la derivada de la función neuronal que se multiplica en cada capa. Puesto que en el caso de funciones neuronales sigmoidales esta derivada tiene un valor entre 0 y $\frac{1}{4}$, los elementos de la matriz jacobiana (derivadas de la funcion de error con respecto a los pesos de la red) pueden diferenciarse en varios órdenes de magnitud para cada una de las capas. Esta es una de las causas que provocan muchos de los problemas del método de retropropagación del error. Para resolver este problema Rigler et al. [119] proponen un método basado en el reescalado de los elementos de esta matriz.

Aunque las técnicas descritas se han aplicado con éxito para acelerar la convergencia de algunos problemas, no hay suficientes experimentos y discusión sobre sus capacidades para evitar mínimos locales. Además, la mayoría de los métodos introducen parámetros adicionales en el entrenamiento que son dependientes del problema [69]. En cuanto a las soluciones encontradas en la bibliografía para el problema de la convergencia hacia mínimos locales, caben destacar:

- *El enfriamiento simulado (simulated annealing).* Es uno de los métodos más conocidos para solucionar el problema de los mínimos locales [74]. Esta técnica acepta la degradación de la función de error con una probabilidad decreciente. En el método de retropropagación del error, el deterioro se logra añadiendo ruido blanco gaussiano en la actualización de los pesos en cada una de las iteraciones del algoritmo. En este caso, la temperatura en la distribución de Boltzman se corresponde con la varianza del ruido. El proceso parte de un valor elevado que se va reduciendo de manera progresiva durante el proceso de entrenamiento, hasta llegar a un valor igual a cero. Styblinski y Tang presentaron un ejemplo para este tipo de aprendizaje [131] mientras que Geman y Geman [50] probaron la existencia de planes de enfriamiento que garantizan la convergencia hacia el mínimo global. Sin embargo, todos ellos requieren una inmensa cantidad de tiempo y el problema que presentan la mayoría de planes diseñados para reducir el tiempo llevan asociada una degradación del rendimiento del aprendizaje. También se han propuesto otras variantes como el *mean field annealing* [20, 112], que consiste en una aproximación determinista que proporciona una mayor velocidad de convergencia. En este método la temperatura controla la pendiente de las funciones sigmoidales, i.e., empezando con una pendiente pequeña que se irá incrementando paulatinamente. Aunque esta aproximación es cincuenta veces más rápida que el enfriamiento simulado para ciertos problemas [20], la calidad de la solución depende en gran medida de tres factores, la temperatura inicial, el plan y la temperatura final [86].

- *Modificaciones del algoritmo estándar de retropropagación del error,* como la que presenta Fukuoka et al. [49]. Proponen mantener la derivada de la función neuro-

nal relativamente grande, aunque de este modo también se mantengan elevados algunos errores. Teniendo en cuenta esta idea, cada peso se multiplica por un factor (dentro del rango (0,1]) en intervalos constantes durante el proceso de aprendizaje. Otra modificación es la presentada por Plaut et al. [114], que modifica la regla de retropropagación mediante la incorporación de un término adicional, denominado *momentum* que permite a la red evitar algunos mínimos locales.

- *Inicialización apropiada de los pesos.* Kolen y Pollack demostraron que el algoritmo de retropropagación del error es muy sensible a los pesos iniciales [77]. De manera habitual los pesos se inicializan con pequeños valores aleatorios. Sin embargo, una inicialización inadecuada de los pesos es una de las razones que provoca la convergencia hacia mínimos locales y un avance lento durante el aprendizaje. En concreto, Lee et al. [88] demuestran que valores iniciales grandes de los pesos pueden causar una saturación prematura de la red.

- *Actualización online de los pesos.* En el campo de las redes de neuronas artificiales se han propuesto diferentes esquemas para la actualización de los pesos. En el primer caso antes de actualizar los parámetros de la red se acumulan las derivadas parciales de la función de error con respecto a los pesos sobre *todos* los ejemplos de entrenamiento. Este modo de entrenamiento recibe el nombre de aprendizaje *batch* y obtiene una aproximación más precisa del gradiente real. Esta aproximación es la que emplea el método clásico de retropropagación del error. El aprendizaje *batch* es uno de los protocolos de entrenamiento más empleado pero existen otros dos, *online* y estocástico. En el entrenamiento *online* los pesos de la red se actualizan cada vez que se presenta una muestra de aprendizaje. Los ejemplos se presentan una única vez, por tanto no es necesario realizar un almacenamiento en memoria de las muestras [41]. Si además los ejemplos se seleccionan aleatoriamente del conjunto de entrenamiento, el método se denomina estocástico, y su nombre se debe a que los datos de entrenamiento pueden ser considerados como una variable aleatoria. El trato aleatorio de los datos provoca pequeñas fluctuaciones que en ocasiones empeoran el valor obtenido de la función de error, por tanto empleando esta aproximación es posible evitar algunos pequeños mínimos locales [152].

Cada uno de los métodos que se acaban de mencionar han aportado soluciones importantes con respecto al problema de los mínimos locales sin embargo, ninguno de ellos proporciona, por sí mismo, una solución realmente eficiente que resuelva a la vez el primero de los problemas planteados por el método de retropropagación del error, la lenta velocidad de convergencia. Las aproximaciones que pueden garantizar que la solución que proporcionan coincide con el mínimo global requieren una cantidad excesiva tanto de recursos temporales como computacionales, por este motivo su aplicación en entornos reales está limitada.

1.4.2. Aprendizaje a lo largo del tiempo

Con respecto a las distintas aproximaciones (*batch*, *online* y estocástica) que se acaban de describir en la sección anterior para la actualización de los pesos, cabe mencionar que la mayoría de los métodos propuestos son de tipo *batch*, es decir, asumen que se dispone, desde el inicio del entrenamiento, de un amplio conjunto de datos que se emplea para ajustar el modelo. Sin embargo, en muchos entornos de aprendizaje la información no está disponible desde el principio, sino que se recibe en tiempo real de manera continua. Por este motivo es fundamental disponer de métodos que pueden aprender a medida que van llegando los datos, en modo secuencial, pero obteniendo la misma eficacia que el aprendizaje *batch*. Las principales razones por las que se prefiere, de manera general, el protocolo de entrenamiento *online* se resumen brevemente a continuación [84]:

- Es crucial para un sistema de aprendizaje que recibe entradas de manera continua y debe procesarlas en tiempo real.
- Es adecuado en entornos dinámicos en los que la función a modelar varía en el tiempo y que presentan cambios de tendencia.
- Incluso si los datos están disponibles desde el inicio, a menudo decrece el tiempo de convergencia, particularmente cuando los conjuntos de datos son grandes y contienen información redundante.

- Un sistema de aprendizaje incremental actualiza sus hipótesis cuando recibe una nueva muestra sin necesidad de reexaminar todos los patrones previos ya conocidos.

- Como no necesita almacenar ni reprocesar viejas instancias requiere menos recursos computacionales y temporales.

Debido a que muchas de las aplicaciones del mundo real presentan un comportamiento dinámico y deben trabajar en tiempo real, lo ideal sería disponer de métodos de aprendizaje secuencial e incremental que presenten las siguientes características. En primer lugar, deben ser capaces de adaptar de manera automática la estructura de la red (*constructive algorithms*) en función de las necesidades generadas en el proceso de aprendizaje [65]. En segundo lugar, deben disponer de la capacidad de olvidar parte de lo que han aprendido previamente, priorizando el conocimiento relativo a la información más reciente y ajustando el modelo aprendido en función de los cambios actuales. Es decir, el algoritmo de aprendizaje debe disponer de capacidad de olvido para adaptarse a aquellas situaciones en las que los cambios en el concepto a aprender (*concept drift*) son altamente posibles [135, 147].

1.5. Objetivos y estructura de la Tesis

El trabajo de Tesis que se presenta en esta memoria se centra en el desarrollo de nuevos modelos de aprendizaje supervisado para redes de neuronas artificiales alimentadas hacia delante. La investigación parte de dos algoritmos desarrollados en el Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (*LIDIA*, [2]) de la Universidade da Coruña. Los principales objetivos del presente trabajo se pueden resumir en:

- Desarrollo de un método de aprendizaje lineal para redes de una capa [30] que permite su uso en modo *online*. Este método se basa en un método de aprendizaje

lineal previo [30] que se caracteriza principalmente por (1) el uso de una función de error que garantiza la no existencia de mínimos locales, y (2) realizar el aprendizaje mediante la resolución de sistemas de ecuaciones lineales. La adaptación llevada a cabo en esta Tesis permite la aplicación de este método en entornos en los que la información disponible es incompleta y susceptible de cambios y, en consecuencia, han de trabajar en tiempo real.

- Desarrollar la adaptación de un método de aprendizaje lineal basado en sensibilidad estadística para redes de dos capas [31] de manera que incorpore la técnica de regularización del *weight decay*. Este método es una extensión del algoritmo de aprendizaje de una capa mencionado en el punto anterior que hace uso de la sensibilidad estadística de los parámetros de cada capa con respecto a sus entradas y salidas. La modificación realizada permite la aplicación del método en aquellas situaciones en las que es posible el sobreajuste a los datos, un problema especialmente crítico cuando se dispone de un número reducido de muestras para realizar el aprendizaje o cuando se manejan datos distorsionados por ruido.

- Diseño y desarrollo de nuevos métodos de aprendizaje adaptativos capaces de aprender en tiempo real y manejar entornos de naturaleza dinámica adaptando los parámetros y estructura de la red. Entre sus características se persigue principalmente que (1) presenten requisitos temporales y espaciales reducidos, y (2) sean escalables con el objeto de manejar conjuntos de datos de dimensión considerable.

Estas ideas se presentaron como proyecto de Tesis en la sección *Doctoral Consortium* de la XIII Conferencia de la Asociación Española para la Inteligencia Artificial celebrada en Salamanca en Noviembre de 2007. El proyecto de Tesis fue galardonado con el premio *José Cuena al Mejor Artículo Doctoral Consortium* por la Asociación Española para la Inteligencia Artificial (AEPIA).

Esta memoria de Tesis Doctoral está organizada como se detalla a continuación. Después de realizar en este capítulo un breve recorrido por las distintas técnicas de

aprendizaje máquina, describiendo conceptos generales del entorno de trabajo y estableciendo el estado del arte para las redes de neuronas artificiales, en el siguiente Capítulo 2 se muestran los algoritmos de aprendizaje que se emplean como punto de referencia del trabajo desarrollado. En primer lugar se describe un método para redes de una capa caracterizado por el empleo de una función de error que garantiza la no existencia de mínimos locales. En segundo lugar, se presenta también su extensión para redes de neuronas de dos capas, un algoritmo llamado SBLLM (*Sensitivity Based Linear Learning Method*, Método de aprendizaje lineal basado en sensibilidad estadística).

Los Capítulos 3 y 4 corresponden con los dos primeros objetivos mencionados previamente. Se describen las mejoras realizadas sobre los algoritmos de aprendizaje de partida para redes de neuronas de una y dos capas con alimentación hacia delante. Concretamente, en el Capítulo 3 se aborda el problema del aprendizaje en redes de neuronas de una sola capa y se muestra un nuevo método de aprendizaje incremental con capacidad para manejar entornos dinámicos y aprender en tiempo real sin la necesidad de mantener datos pasados o modelos obsoletos. El método incorpora la capacidad de olvido gracias a la introducción de un término en la función de coste, lo que le permite adaptarse tanto a entornos estáticos como dinámicos. Este trabajo ha sido aceptado para su publicación en la revista *Neurocomputing* [94]. A su vez, en el Capítulo 4 se presenta una modificación del algoritmo SBLLM para redes de neuronas de dos capas. Con el objeto de evitar el problema del sobreeentrenamiento, en aquellas situaciones en las que el conjunto de entrenamiento no es suficientemente representativo del problema a resolver, se incorpora la técnica de regularización *weight decay*. La nueva versión del método junto con la experimentación que avala su rendimiento en posibles situaciones de *overfitting* fue presentado en el congreso internacional *European Symposium on Artificial Neural Networks (ESANN 2008)* [55]. En este trabajo, la estimación del parámetro de regularización se realiza mediante la técnica de validación cruzada lo que implica un coste temporal y computacional a tener en cuenta. Como mejora el algoritmo desarrollado incorpora una aproximación que proporciona una estimación automática del valor de dicho parámetro. Esta parte del trabajo fue presentada en el

congreso internacional *International Work-Conference on Artificial and Natural Neural Networks (IWANN 2009)* [116].

Posteriormente, los Capítulos 5 y 6 corresponden al tercer objetivo descrito anteriormente. En el Capítulo 5 se presenta un nuevo algoritmo de aprendizaje *online* e incremental para redes de neuronas de dos capas con alimentación hacia delante. Esta aproximación permite, por una parte, el aprendizaje en entornos que tienen que trabajar en tiempo real y, por otra, el tratamiento de grandes conjuntos de datos ya que realiza un empleo óptimo de los recursos computacionales disponibles. Este trabajo fue presentado en la *XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2009)* [115]. A mayores el método es capaz de manejar problemas no estacionarios. Esta habilidad para olvidar parcialmente el conocimiento adquirido previamente se consigue gracias a la incorporación de un factor que pondera el error cometido en cada una de las muestras, obteniendo un balance entre el conocimiento que proporcionan las muestras antiguas y las más recientes. Esta extensión del algoritmo ha sido publicada en el congreso internacional *International Joint Conference on Neural Networks (IJCNN 2010)* [117]. En el Capítulo 6 se plantea la posibilidad de que el algoritmo de aprendizaje presentado en el capítulo previo adapte también la topología de la red en función de las necesidades del proceso de aprendizaje *online*. En la primera parte de este capítulo se demuestra la idoneidad del algoritmo para adaptar la estructura de red manteniendo el conocimiento previo procesado. Una vez comprobada la capacidad de adaptar la topología de la red, en la segunda parte del capítulo se aplica una técnica para la automatización del proceso de optimización de la estructura del modelo.

Finalmente, el Capítulo 7 contiene las principales conclusiones y aportaciones de esta Tesis, así como las posibles líneas de trabajo futuro.

Capítulo 2

Antecedentes

En el presente capítulo se presentan los algoritmos de aprendizaje que establecen el punto de partida del trabajo desarrollado, y que han servido como herramienta para la elaboración de esta Tesis Doctoral. En primer lugar nos centramos en el problema de aprendizaje para redes de neuronas de una sola capa con alimentación hacia delante. El algoritmo tomado como referencia para tratar este problema fue presentado inicialmente en [30]. Este método se caracteriza principalmente por el empleo de una función de error que permite garantizar la no existencia de mínimos locales. Además tiene la particularidad de llevar a cabo el aprendizaje mediante la resolución de sistemas de ecuaciones lineales. El método de aprendizaje es rápido y proporciona la solución en un tiempo menor que cualquier otro algoritmo basado en métodos iterativos. La filosofía de este algoritmo puede extenderse para su aplicación en el aprendizaje de redes de dos capas con alimentación hacia delante. De esta manera aparece un método de aprendizaje basado en la sensibilidad estadística de los parámetros de cada capa con respecto a sus entradas y salidas (*Sensitivity Based Linear Learning Method*, SBLLM) [31], y que también emplea sistemas de ecuaciones lineales independientes para la obtención de los pesos de cada una de las capas de la red neuronal.

Ambos métodos de aprendizaje presentan características importantes e innovadoras. En las siguientes secciones se presentan cada uno de los algoritmos de manera detallada, con el objeto de establecer las bases del trabajo desarrollado.

2.1. Aprendizaje en redes de neuronas de una capa

Las redes neuronales de una capa fueron ampliamente estudiadas en la década de los 60, y revisadas posteriormente en diferentes trabajos. Para una red de neuronas de una capa con alimentación hacia delante, con funciones de activación lineales, los valores de los pesos que minimizan la función de error (error cuadrático medio, ECM) se pueden encontrar gracias a la pseudo-inversa de una matriz [21, 107]. Además, es posible demostrar que la superficie ECM de esta red lineal es una función cuadrática de los pesos [151]. De este modo, esta superficie hiperparaboloide convexa puede ser atravesada mediante un método de gradiente descendente. Sin embargo, si se emplea como función de transferencia de las neuronas una función no lineal pueden existir mínimos locales en la función objetivo basada en el criterio del ECM [24, 25, 129]. En el trabajo de Auer et al. [11] se demuestra que el número de estos mínimos locales puede aumentar exponencialmente con la dimensión de las entradas de la red. Sólo en ciertas situaciones se garantiza la ausencia de mínimos locales. En el caso concreto de manejar patrones linealmente separables y emplear el criterio del ECM, es posible probar la existencia de un único mínimo en la función objetivo [54, 129]. Sin embargo, ésta no es la situación general.

El problema de los mínimos locales para redes de una capa fue demostrado matemáticamente por Sontag y Sussmann [129], quienes proporcionaron un ejemplo de red neuronal sin capas ocultas con funciones de transferencia sigmoidales para las cuales la suma de los errores al cuadrado tiene un mínimo local que no es el mínimo global. Además observaron que la existencia de mínimos locales es debida al hecho de que la función de error es la superposición de funciones cuyos mínimos se encuentran en dife-

rentes puntos. En el caso de funciones de activación lineales, todas estas funciones son convexas, de modo que también lo es la suma de las mismas. Sin embargo, las unidades sigmoidales conllevan funciones no lineales, de manera que no se garantiza que su suma posea un mínimo único.

Durante las últimas décadas han ido apareciendo diferentes aproximaciones que intentan solucionar los problemas ocasionados por la presencia de estos puntos estacionarios en redes de neuronas de una capa. Así, en [35] se define una homotopía globalmente convergente para perceptrones de una capa mediante la deformación de la no linealidad. Esta homotopía rastrea un número posiblemente infinito de pesos transformando las coordenadas y caracterizando todas las soluciones mediante un número finito de soluciones únicas y diferentes. Sin embargo, a pesar de que esta aproximación garantiza la obtención de una solución, no proporciona una optimización global [21]. Al mismo tiempo, estos autores proponen en [36] un método con dos características interesantes. En primer lugar, permite obtener una evaluación a posteriori para conocer si la solución alcanzada es un óptimo único o global y por otro lado, con el objeto de garantizar la unicidad, escala los valores de las salidas deseadas a partir del análisis de los datos de entrada. A pesar de que estas aproximaciones son una ayuda potencial para evaluar tanto la optimalidad como la unicidad, su inconveniente principal es que los mínimos se caracterizan tras haber finalizado el proceso de entrenamiento.

Pao [107] propone una aproximación denominada *functional link* que obtiene una solución analítica de los pesos al establecer un sistema de ecuaciones lineales $Xw = z$, donde X es la matriz de patrones de entrada, w el vector de pesos y z otro vector formado por la inversa de la función de activación aplicada a la salida deseada. La dimensión de la matriz X viene dada por $S \times N$, donde S es el número de patrones de entrenamiento y N el número de pesos. Como menciona Pao en su trabajo, en caso de que $S = N$ y el determinante de X sea distinto de cero la solución se puede obtener como $w = X^{-1}z$. Sin embargo, esta situación no es habitual ya que en conjuntos reales generalmente $S > N$ ó $S < N$. Pao aborda estas situaciones analizando los casos de manera separada. Si se cumple que $S < N$, se puede obtener un número elevado de

soluciones (posiblemente un número infinito). Obviamente esta situación no es deseable así que para evitar el problema, al menos en la medida de lo posible, Pao propone emplear una partición de X. Cuando $S > N$, se pueden generar un número infinito de funciones ortonormales, en este caso Pao plantea un método basado en la pseudoinversa ($w = (X^T X)^{-1} X^T z$). Sin embargo, esta solución puede alcanzar un error inaceptable al final del proceso.

2.1.1. Método de aprendizaje lineal

Consideremos la red de una capa que se presenta en la figura 2.1. Asumimos que las funciones de activación no lineales $f_1, f_2, \ldots, f_j$ son invertibles como es el caso de las funciones empleadas habitualmente en este tipo de sistemas.

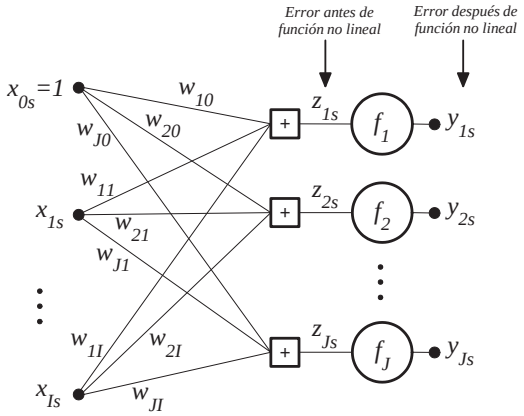


Figura 2.1: Red de neuronas de una capa con alimentación hacia delante.

El conjunto de ecuaciones que permite relacionar las entradas y las salidas de la red viene dado por,

$$y_{js} = f_j \left(\sum_{i=0}^{I} w_{ji} x_{is} \right); j = 1, 2, \ldots, J; s = 1, 2, \ldots, S, \tag{2.1}$$

donde I es el número de entradas, J el número de salidas, y S el número de muestras de entrenamiento para la red. Además, w_{ji} son los pesos asociados a la neurona j y a la

entrada i y los sesgos x_{0s} toman el valor 1. Estos sesgos o entradas de valor constante a las neuronas permiten disponer de pesos extra que proporcionan un nivel de activación independiente de las entradas y los vectores de entrenamiento con el objetivo de obtener una salida distinta de cero en el caso de que todas las variables de entrada tomen este valor. El valor del peso ligado al sesgo se modifica de la misma forma que el resto de pesos de la red. De este modo, los pesos w_{ji} forman el conjunto de parámetros adaptativos de la red.

El sistema correspondiente a la ecuación 2.1 tiene $J \times S$ ecuaciones y $J \times (I+1)$ incógnitas. En la práctica, como el número disponible de datos S es habitualmente mucho mayor que el número de entradas ($S >> I+1$), este conjunto de ecuaciones es no compatible y, en consecuencia, no posee solución. En esta situación, la aproximación habitual es considerar ciertos errores, ε_{js}, entre la salida esperada y la obtenida por cada neurona de salida de la red de manera que la ecuación 2.1 se transforma en,

$$\varepsilon_{js} = d_{js} - y_{js} = d_{js} - f_j\left(\sum_{i=0}^{I} w_{ji}x_{is}\right); j = 1, 2, \ldots, J; s = 1, 2, \ldots, S; \tag{2.2}$$

donde d_{js} es el valor de la salida deseada para la neurona j y el dato de entrada s.

En este caso, para estimar los pesos se minimiza, usualmente, el ECM definido como:

$$Q = \sum_{s=1}^{S}\sum_{j=1}^{J}\varepsilon_{js}^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(d_{js} - f_j\left(\sum_{i=0}^{I} w_{ji}x_{is}\right)\right)^2. \tag{2.3}$$

Nótese que debido a la presencia de funciones de activación neuronales no lineales f_j, la función que aparece en la ecuación 2.3 no es lineal en los pesos w_{ji} de manera que no se puede garantizar que la función objetivo Q tenga un único óptimo global. Para obtener la solución, de manera general, se emplean métodos iterativos basados en el descenso del gradiente.

Como ya se ha introducido previamente, este tipo de métodos clásicos para el aprendizaje de pesos en redes de neuronas presentan dos inconvenientes importantes. En primer lugar, la función a optimizar (Q) tiene varios mínimos locales. Esto supone que

no es posible conocer si con el resultado obtenido para los pesos se está en presencia de un óptimo global o local y, en este último caso, es imposible saber cómo de lejos se encuentra de una solución óptima. De hecho, es posible conseguir, para el mismo conjunto de datos y el mismo modelo de red, diferentes soluciones si se utilizan en el entrenamiento conjuntos iniciales de pesos distintos. En segundo lugar, el proceso de aprendizaje requiere un elevado consumo computacional en comparación con otros modelos, ya que los pesos han de obtenerse mediante un mecanismo iterativo que utilice el gradiente de la función de error con respecto a los valores de estos pesos.

Motivado por los problemas que se acaban de describir, en [30] se plantea una aproximación alternativa para estimar los valores óptimos de los pesos de la red, que mide los errores *antes de* las funciones no lineales en lugar de después de ellas. Por tanto, el sistema de ecuaciones de 2.2 se puede reescribir de la siguiente forma:

$$\bar{\varepsilon}_{js} = \bar{d}_{js} - z_{js} = f_j^{-1}(d_{js}) - \sum_{i=0}^{I} w_{ji} x_{is}, s = 1, 2, \ldots, S; j = 1, 2, \ldots, J. \tag{2.4}$$

Es importante destacar que en la ecuación 2.4, a diferencia de lo que sucede en 2.2, los pesos no están dentro de la función de activación (f_j) y, por tanto, el error es lineal con respecto a los pesos de la red. De este modo, para obtener los pesos óptimos se minimiza la siguiente suma de los errores al cuadrado,

$$Q = \sum_{s=1}^{S} \sum_{j=1}^{J} \bar{\varepsilon}_{js}^2 \quad = \quad \sum_{s=1}^{S} \sum_{j=1}^{J} \left(f_j^{-1}(d_{js}) - \sum_{i=0}^{I} w_{ji} x_{is} \right)^2 .$$

A mayores, es necesario tener en cuenta la influencia de las funciones de activación no lineales en los errores considerados en la función objetivo. Así en [48] se demuestra que la minimización de la media de los errores al cuadrado en la salida de la red es equivalente (hasta primer orden) a la minimización del error antes de las neuronas de salida ponderado por un factor $f_j^{'}(\bar{d}_{js})$. Este factor asegura que cada error, correspondiente a un par entrada-salida del conjunto de entrenamiento, se pondera de manera adecuada de acuerdo a la derivada del valor de la función de activación no lineal en el punto correspondiente de la respuesta deseada. El único requisito que exige el problema de

minimización alternativo es que las funciones de activación deben ser invertibles, pero ésta no es una condición muy restrictiva ya que la mayoría de las funciones de activación empleadas en redes de neuronas presentan esa característica. De este modo, ahora el objetivo es minimizar la función de coste alternativa [48], que para cada salida j de la red se puede realizar de forma independiente, y viene dada por la siguiente ecuación:

$$Q_j = \sum_{s=1}^{S} \left(f_j^{'}(\bar{d}_{js})\bar{\varepsilon}_{js} \right)^2 = \sum_{s=1}^{S} \left(f_j^{'}(\bar{d}_{js}) \left(f_j^{-1}(d_{js}) - \sum_{i=0}^{I} w_{ji}x_{is} \right) \right)^2 . \tag{2.5}$$

El óptimo global de esta función objetivo convexa se obtiene de manera sencilla al calcular su derivada con respecto a los pesos de la red e igualando la misma a cero:

$$\frac{\partial Q_j}{\partial w_{jp}} = -2 \sum_{s=1}^{S} \left(f_j^{'}(\bar{d}_{js}) \left(f_j^{-1}(d_{js}) - \sum_{i=0}^{I} w_{ji}x_{is} \right) \right) x_{ps} f_j^{'}(\bar{d}_{js}) = 0; p = 0, 1, \ldots, I,$$

que se puede reescribir como:

$$\sum_{i=0}^{I} A_{pi} w_{ji} \quad = \quad b_{pj}; \quad p = 0, 1, \ldots, I, \tag{2.6}$$

donde

$$A_{pi} \quad = \quad \sum_{s=1}^{S} x_{is} x_{ps} f_j^{'2}(\bar{d}_{js}); \quad p = 0, 1, \ldots, I; \quad i = 0, 1, \ldots, I; \tag{2.7}$$

$$b_{pj} \quad = \quad \sum_{s=1}^{S} f_j^{-1}(d_{js}) x_{ps} f_j^{'2}(\bar{d}_{js}); p = 0, 1, \ldots, I. \tag{2.8}$$

Cabe indicar que en la ecuación 2.6 se obtiene un sistema de ecuaciones lineales con $(I+1)$ ecuaciones e incógnitas que es compatible determinado (excepto para problemas mal condicionados).

El Algoritmo 2.1 resume los pasos del método de aprendizaje que permite obtener los pesos óptimos para una red de una capa empleando los conceptos descritos hasta ahora.

En resumen, las principales características del método de aprendizaje que se acaba de presentar son:

Algoritmo 2.1 Aprendizaje lineal para redes de neuronas de una capa con alimentación hacia delante.

Entradas: entradas, $\mathbf{x}_s = (x_{1s}, x_{2s}, \ldots x_{Is})$, $s = 1, \ldots, S$
salidas deseadas, $\mathbf{d}_s = (d_{1s}, d_{2s}, \ldots d_{Js})$
sesgo, $x_{0s} = 1$.

1. Para cada salida j de la red ($j = 1, \ldots, J$),
2. Calcular A_{pi} usando la ecuación 2.7, $\forall i, p = 0, 1, \ldots, I$.
3. Calcular b_{pj} empleando la ecuación 2.8, $\forall p$.
4. Calcular w_{ji} resolviendo el sistema de ecuaciones lineales de la ecuación 2.6.
5. fin del Para.

- El óptimo global obtenido es aproximadamente equivalente al que se obtiene con el ECM cuando se mide después de las funciones de activación [48].
- La función objetivo empleada, basada en el ECM y evaluada antes de las funciones de activación no lineales, es estrictamente convexa y asegura la ausencia de mínimos locales.
- Para cada salida j, el sistema de la ecuación 2.6 tiene $I + 1$ ecuaciones lineales e incógnitas, de manera que existe una única solución real (excepto para problemas mal condicionados) que se corresponde con el óptimo global de la función objetivo. Existen varios métodos computacionalmente eficientes que pueden emplearse para resolver este tipo de sistemas con complejidad $O(N^2)$ donde $N = I + 1$ es el número de pesos para la salida j [23, 27].
- Tiene naturaleza incremental y distribuida. Este hecho se comprueba de manera sencilla al observar las ecuaciones 2.7 y 2.8. Los coeficientes A_{pi} y b_{pj} se calculan como una suma de términos en función de los pares entrada-salida deseada del conjunto de datos. Por tanto, debido a las propiedades asociativa y conmutativa de la suma, se obtiene la misma solución independientemente del orden y de la

presentación de los datos. Además, es posible entrenar M redes en diferentes ordenadores con subconjuntos de datos $(D_1, D_2, \ldots, D_M)$ y obtener posteriormente una red sencilla que represente la unión de las M redes correspondientes. Para ello simplemente habría que sumar las correspondientes M matrices de coeficientes (A_{pi} y b_{pj}) de todas las redes y obtener los pesos para la nueva matriz.

La idea desarrollada en este método fue empleada posteriormente para obtener un nuevo método de aprendizaje para redes de neuronas de *dos* capas con alimentación hacia delante. Esta investigación fue publicada en [31] y se presenta de manera detallada en la siguiente sección.

2.2. Aprendizaje lineal basado en sensibilidad estadística

El análisis de la sensibilidad es una técnica muy útil para derivar cómo y en qué medida la solución a un problema dado depende de los datos [28, 29, 32]. Por este motivo, las fórmulas de la sensibilidad también pueden emplearse en el entrenamiento. De esta manera es posible aplicar el método de aprendizaje mostrado en la sección previa y el concepto de sensibilidad estadística para el desarrollo de un nuevo método de aprendizaje para redes de neuronas de dos capas con alimentación hacia delante. En [31] se propone este novedoso algoritmo de aprendizaje que presenta una rápida velocidad de convergencia hacia la solución. Este algoritmo, conocido como Método de Aprendizaje Lineal Basado en Sensibilidad Estadística (*Sensitivity Based Linear Learning Method, SBLLM*), se basa en primer lugar, en el uso de las sensibilidades de los parámetros de cada capa con respecto a sus entradas y salidas, y en segundo lugar, en el empleo de sistemas de ecuaciones lineales independientes para cada capa que permiten calcular los valores óptimos de sus parámetros.

2.2.1. Descripción del método

Consideremos la red de neuronas de dos capas con alimentación hacia delante de la figura 2.2. Siendo I el número de entradas, J el número de salidas, K el número de unidades ocultas y S el número de muestras de entrenamiento. Además, los sesgos asociados x_{0s}, z_{0s} toman el valor 1, y los superíndices (1) y (2) se emplean para indicar la primera y segunda capa, respectivamente.

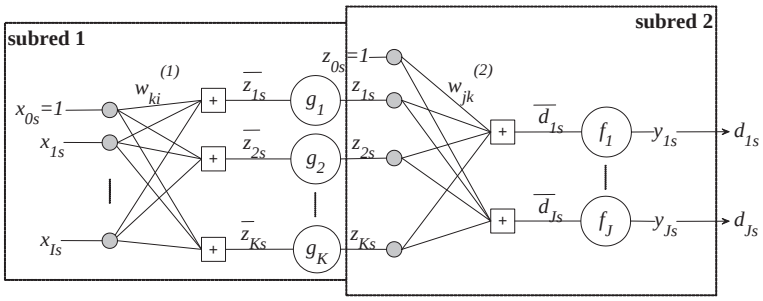


Figura 2.2: Red de neuronas de dos capas con alimentación hacia delante.

La red puede ser considerada como la composición de dos redes de neuronas de una capa, de este modo, asumiendo que las salidas de la capa intermedia $\mathbf{z}$ son conocidas, y utilizando los resultados del método expuesto en la sección 2.1, se define la función de coste para la primera de las subredes como,

$$Q^{(1)} = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\bar{\varepsilon}_{ks}\right)^2 = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)}x_{is} - \bar{z}_{ks}\right)\right)^2, \qquad (2.9)$$

siendo $\bar{z}_{ks} = g_k^{-1}(z_{ks})$. De manera análoga, es posible definir la función de coste para la segunda subred como,

$$Q^{(2)} = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(f_j'(\bar{d}_{js})\bar{\varepsilon}_{js}\right)^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(f_j'(\bar{d}_{js})\left(\sum_{k=0}^{K} w_{jk}^{(2)}z_{ks} - \bar{d}_{js}\right)\right)^2, \qquad (2.10)$$

donde $\bar{d}_{js} = f_j^{-1}(d_{js})$. De este modo, la función de coste para la red completa se define

como la suma de las funciones objetivo parciales,

$$Q(\mathbf{z}) = Q^{(1)}(\mathbf{z}) + Q^{(2)}(\mathbf{z}) = \tag{2.11}$$
$$\sum_{s=1}^{S}\left[\sum_{k=1}^{K}\left(g_k^{'}(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)^2 + \sum_{j=1}^{J}\left(f_j^{'}(\bar{d}_{js})\left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - \bar{d}_{js}\right)\right)^2\right].$$

Como se puede observar, empleando las salidas z_{ks} es posible aprender, de manera independiente para cada una de las capas de la red, los pesos $w_{ki}^{(1)}$ y $w_{jk}^{(2)}$ al resolver sistemas de ecuaciones lineales independientes. Asimismo, se pueden calcular las sensibilidades de la función de coste con respecto a z_{ks} como la suma de la sensibilidad de la primera capa con respecto a sus salidas y la sensibilidad de la segunda de las capas con respecto a sus entradas. De este modo, obtenemos la ecuación

$$\frac{\partial Q}{\partial z_{ks}} = \frac{\partial Q^{(1)}}{\partial z_{ks}} + \frac{\partial Q^{(2)}}{\partial z_{ks}}, \tag{2.12}$$

donde el primer término de la suma, sensibilidad de la primera capa con respecto a sus salidas viene dado por,

$$\begin{aligned}\frac{\partial Q^{(1)}}{\partial z_{ks}} = -2&\left(g_k^{'}(\bar{z}_{ks})(g_k^{-1})^{'}(z_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)\\ &\left(g_k^{'}(\bar{z}_{ks}) - g_k^{''}(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right), k = 1, \ldots, K, \forall s.\end{aligned} \tag{2.13}$$

Del mismo modo, el segundo término, la sensibilidad de la segunda subred con respecto a sus entradas, se define como

$$\frac{\partial Q^{(2)}}{\partial z_{ks}} = 2\sum_{j=1}^{J}\left(f_j^{'}(\bar{d}_{js})\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - \bar{d}_{js}\right)\right) f_j^{'}(\bar{d}_{js}) w_{jk}^{(2)}, \forall j, \forall s. \tag{2.14}$$

Utilizando estas sensibilidades, es posible modificar los valores de las salidas $\mathbf{z}$ de la capa intermedia empleando la siguiente aproximación de la serie de Taylor,

$$Q(\mathbf{z} + \Delta\mathbf{z}) = Q(\mathbf{z}) + \sum_{k=1}^{K}\sum_{s=1}^{S}\frac{\partial Q(\mathbf{z})}{\partial z_{ks}}\Delta z_{ks} \approx 0, \tag{2.15}$$

que permite obtener los siguientes incrementos,

$$\Delta\mathbf{z} = -\rho\frac{Q(\mathbf{z})}{||\nabla Q||^2}\nabla Q, \tag{2.16}$$

con los que actualizar los valores z_{ks}, siendo ρ el paso de aprendizaje. Posteriormente, haciendo uso de los nuevos valores calculados para las salidas de la capa intermedia z_{ks}, se pueden obtener, para cada capa de forma independiente, los pesos óptimos utilizando el método de aprendizaje para redes de neuronas de una capa presentado en la sección 2.1 [30]. Teniendo en cuenta las consideraciones que se acaban de mencionar el algoritmo de aprendizaje del SBLLM se presenta a continuación,

Algoritmo 2.2 Algoritmo de aprendizaje lineal basado en sensibilidad estadística, SBLLM.

Entradas: $\mathbf{x}_s = (x_{1s}, x_{2s}, \dots x_{Is}),\ s = 1, \dots, S,$
$\mathbf{d}_s = (d_{1s}, d_{2s}, \dots d_{Js}),$
sesgos, $x_{0s} = 1, z_{0s} = 1$
umbrales de error ϵ y ϵ' para control de convergencia,
paso de aprendizaje ρ.

Salidas: pesos de ambas capas, $\mathbf{W}^{(l)}, l = 1, 2,$
sensibilidades de la función de coste con respecto a entradas y salidas.

Fase de inicialización.

- Asignar a las salidas $\mathbf{z}$ de la capa intermedia la salida generada con ciertos pesos aleatorios $\mathbf{w}^{(1)}(0)$ más un cierto error, es decir:

$$z_{ks} = f_k^{(1)} \left(\sum_{i=0}^{I} w_{ki}^{(1)}(0) x_{is} \right) + \epsilon_{ks}; \quad k = 1, \dots, K,$$

  donde $\epsilon_{ks} \sim U(-\eta, \eta)$ y η es un número pequeño.

- Inicializar también $Q_{anterior}$, $ECM_{anterior}$ y $\mathbf{z}_{anterior}$ a números grandes, donde ECM mide el error cuadrático medio entre la salida obtenida y la deseada. Estas variables se emplearán para restaurar el sistema a un estado previo cuando tras un cambio en los pesos se produzca un deterioro en el comportamiento.

Paso 1: Solucionar los subproblemas de cada capa.

Obtener los pesos óptimos de las capas 1 y 2 y las sensibilidades asociadas con respecto a las salidas de la capa intermedia, mediante la resolución de los sistemas de ecuaciones lineales correspondientes, esto es,

1. Para cada k neurona oculta de la red ($k = 1, \ldots, K$), y siendo $\bar{z}_{ks} = g_k^{-1}(z_{ks})$
2. Calcular $A_{pi}^{(1)}$ como $A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is} x_{ps} g_k'^2(\bar{z}_{ks})$, $p = 0, 1, \ldots, I; \forall k$.
3. Calcular $b_{pk}^{(1)}$ como $b_{pk}^{(1)} = \sum_{s=1}^{S} \bar{z}_{ks} x_{ps} g_k'^2(\bar{z}_{ks})$, $p = 0, 1, \ldots, I; \forall k$.
4. Obtener $w_{ki}^{(1)}$, mediante el sistema de ec. lineales $\sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} = b_{pk}^{(1)}$.
5. Obtener las sensibilidades con respecto a sus salidas, $\dfrac{\partial Q^{(1)}}{\partial z_{ks}}$, según la ec. 2.13.
6. fin del Para.
7. Para cada salida j de la red ($j = 1, \ldots, J$), y siendo $\bar{d}_{js} = f_j^{-1}(d_{js})$,
8. Calcular $A_{qk}^{(2)}$ como $A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks} z_{qs} f_j'^2(\bar{d}_{js})$, $q = 0, 1, \ldots, K; \forall j$
9. Calcular $b_{qj}^{(2)}$ como $b_{qj}^{(2)} = \sum_{s=1}^{S} \bar{d}_{js} z_{qs} f_j'^2(\bar{d}_{js})$, $q = 0, 1, \ldots, K; \forall j$
10. Obtener $w_{jk}^{(2)}$ mediante el sistema de ec. lineales $\sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} = b_{qj}^{(2)}$.
11. Obtener las sensibilidades con respecto a sus entradas, $\dfrac{\partial Q^{(2)}}{\partial z_{ks}}$, según la ec. 2.14.
12. fin del Para.

Paso 2: Calcular la suma de errores al cuadrado.

- Calcular Q empleando la ecuación 2.11,
- Calcular el ECM a la salida de la red.

Paso 3: Comprobar la convergencia.

- Si $|Q - Q_{anterior}| < \epsilon$ ó $|ECM_{anterior} - ECM| < \epsilon'$ parar el proceso y devolver los pesos y las sensibilidades.
- En otro caso, el método continúa en el paso 4.

Paso 4: Comprobar la mejora de Q.

- Si $Q > Q_{anterior}$ se reduce el valor de ρ y se regresa al estado anterior. Esto quiere decir que se restauran los pesos $\mathbf{z} = \mathbf{z}_{anterior}$ y $Q = Q_{anterior}$ y también $ECM = ECM_{anterior}$.
- En otro caso, almacenar los valores $Q_{anterior} = Q$, $ECM_{anterior} = ECM$ y $\mathbf{z}_{anterior} = \mathbf{z}$. Calcular además las sensibilidades $\dfrac{\partial Q}{\partial z_{ks}}$ empleando la ecuación 2.12.

Paso 5: Actualizar las salidas intermedias. Empleando la aproximación de la serie de Taylor de la ecuación 2.15, actualizar las salidas intermedias como

$$\mathbf{z} = \mathbf{z} - \rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q,$$

y volver al Paso 1.

La complejidad computacional de este método viene dada por la complejidad del Paso 1 en el que se resuelven los sistemas de ecuaciones lineales para cada capa (uno para cada neurona de salida de cada subred). Como se ha comentado en la sección previa, existen diferentes métodos eficientes que se pueden aplicar para resolver este tipo de sistemas con una complejidad $O(N^2)$, siendo N el número de parámetros desconocidos. Por tanto, la complejidad del método de aprendizaje es $O(MN^2)$, donde $M = K + J$.

En la figura 2.3 se presenta un ejemplo del comportamiento del SBLLM para la serie temporal de Dow-Jones [31]. El SBLLM se compara con cinco de los métodos de aprendizaje más populares tanto de primer como de segundo orden. Estos son, el gradiente descendente (GD), el gradiente descendente con momento y paso de aprendizaje adaptativo (GDX), el gradiente descendente estocástico (SGD), el gradiente conjugado escalado (SCG) y el Levenberg Marquardt (LM). Para cada uno de ellos se muestra la curva correspondiente al ECM de test obtenido sobre 100 simulaciones. Como se puede observar el SBLLM presenta una mayor velocidad de convergencia mientras que alcanza un buen valor del error en un número reducido de iteraciones de entrenamiento.

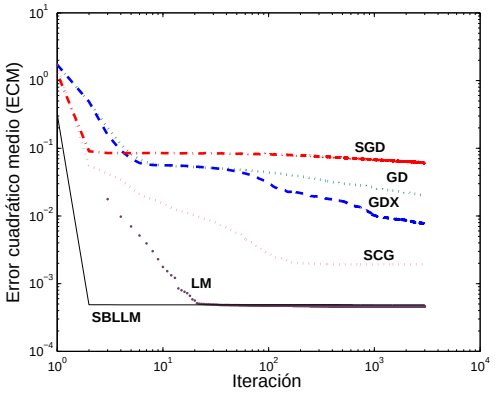


Figura 2.3: Comparativa del comportamiento del algoritmo SBLLM con otros métodos de aprendizaje de primer y segundo orden para la serie temporal Dow-Jones.

El algoritmo SBLLM hereda muchas de las importantes propiedades que presenta el método de aprendizaje lineal para redes de una capa. Las principales innovaciones que presenta el SBLLM con respecto a métodos de aprendizaje presentados anteriormente por otros autores, se pueden resumir fundamentalmente en:

1. La función de coste que se minimiza es el error cuadrático medio calculado antes de las funciones de activación neuronales.

2. Los pesos de cada capa de la red se calculan resolviendo un sistema de ecuaciones lineales.

3. Muestra una alta velocidad de convergencia (véase figura 2.3).

4. Obtiene un buen rendimiento en términos de error cometido (véase figura 2.3).

5. Presenta un comportamiento homogéneo. Esto puede comprender varios aspectos:

   - Las curvas de aprendizaje del SBLLM se estabilizan pronto.
   - Presenta una baja dispersión (con respecto al valor mínimo del ECM que el algoritmo alcanza al final del proceso de aprendizaje).

Capítulo 3

Algoritmo de aprendizaje en entornos no estacionarios para redes de neuronas de una capa

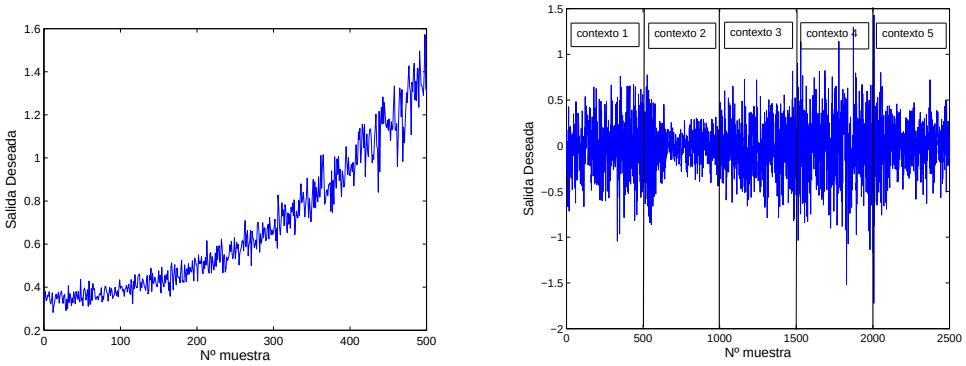
Existe un gran número de escenarios en los problemas a resolver con los métodos de aprendizaje máquina que son no estacionarios, esto quiere decir que la función que debe ser modelada varía en el tiempo. En consecuencia, el aprendizaje pasado debe ser sustituido por el conocimiento que proporciona la nueva información recibida. El comportamiento no estacionario puede aparecer tanto en problemas de clasificación como de regresión o identificación de sistemas. En aplicaciones del mundo real, se encuentra un número considerable de problemas que se comportan de este modo, tales como filtrado del spam en el correo electrónico, análisis de datos climáticos o financieros, detección de intrusos, protección de fraude de tarjetas bancarias, monitorización de tráfico, predicción del comportamiento del consumidor, etc [42, 143, 144]. El aprendizaje en este tipo de entornos puede ser un reto muy complicado, ya que los algoritmos deben tener en cuenta dos consideraciones importantes. En primer lugar, tienen que asumir que la información disponible para el entrenamiento en cualquier momento es

incompleta y susceptible de cambios. En segundo lugar, no pueden tratar todas las muestras de entrenamiento de la misma manera, sino que deben extraer el concepto subyacente de cada muestra particular [40].

En este tipo de entornos, el aprendizaje de tipo *batch* no obtiene resultados satisfactorios. Los retos actuales en el campo del aprendizaje máquina hacen necesario poder aprender en entornos no estacionarios. Este hecho implica el desarrollo de nuevos algoritmos capaces de manejar los posibles cambios en el problema a aprender. Durante los últimos años el aprendizaje *online* ha sido un campo de interés creciente debido a las necesidades que presentan los modernos sistemas de información. Los modelos de aprendizaje incremental o secuencial, son capaces de procesar los datos uno a uno de tal manera que, en cada paso, el modelo obtenido no es peor que otro que hubiese sido entrenado en modo *batch* empleando todos los datos disponibles. Las principales cualidades que debe presentar un buen algoritmo *online* de aprendizaje máquina se reducen fundamentalmente a [130]:

- El modelo debe ser capaz de aprender sin necesidad de revisar la información pasada. Por tanto, no es necesario almacenar las muestras o datos analizados previamente. Es decir, debe tener capacidad de aprendizaje incremental.
- Cada una de las muestras debe ser procesada en tiempo real.
- El modelo debe proporcionar la mejor respuesta posible en cada instante de tiempo.

En entornos no estacionarios, la naturaleza dinámica de la tarea de aprendizaje puede ser clasificada en uno de estos tipos [147]: cambios *graduales* o de *tendencia* y cambios *bruscos* también denominados *contextos ocultos*. Un ejemplo de cada uno de ellos se puede observar en la figura 3.1. Debido a que la dinámica de los cambios puede ser muy diferente, los algoritmos de aprendizaje máquina existentes muestran dificultades para abordar esta tarea.



(a) (b)

Figura 3.1: Ejemplos de los tipos de cambios que puede sufrir el proceso que genera la serie a modelar: (a) cambio gradual continuo a lo largo del tiempo, (b) cambios bruscos cada 500 muestras dando lugar a diferentes contextos.

Como ya se describió en el Capítulo 2 este trabajo de Tesis toma como punto de partida un método de aprendizaje para redes de neuronas de una sola capa que está basado en la resolución de un sistema de ecuaciones lineales. Por su naturaleza, este algoritmo puede ser aplicable de modo incremental. En este capítulo, se propone una nueva versión de este método de aprendizaje con el fin de incorporar la capacidad de adaptación para trabajar en entornos dinámicos. Esta habilidad se consigue gracias a la introducción de una función de olvido que asigna una importancia creciente a los nuevos datos de entrenamiento frente a los más antiguos. Debido a la combinación del aprendizaje incremental y la asignación de importancia creciente, la red olvidará en presencia de un cambio mientras mantiene un comportamiento estable cuando el contexto es estacionario. El modelo planteado, como demuestran los experimentos realizados, consigue una elevada adaptación a los cambios que se producen en el sistema a modelar, manteniendo al mismo tiempo un consumo reducido de recursos computacionales.

3.1. Descripción del método propuesto

Consideremos la arquitectura que se muestra en la figura 3.2 para una red de neuronas de una sola capa. Las entradas se denotan como $x_i(s)$ y las salidas como $y_j(s)$ siendo $i = 0, 1, ..., I$; $j = 1, 2, ..., J$ y $s = 1, 2, ..., S$. La red contiene una única capa con J neuronas de salida, con funciones de activación no lineales $f_1, f_2, ..., f_J$. La variable $d_j(s)$ es la salida deseada para la neurona j y el patrón de entrenamiento s.

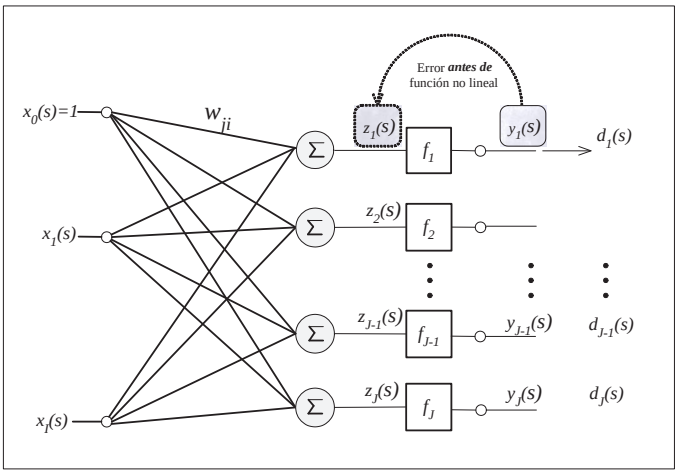


Figura 3.2: Arquitectura de una red de neuronas de una capa con alimentación hacia delante.

En este contexto y como se ha visto en el Capítulo 2, se pueden obtener los errores antes de las funciones de activación no lineales en lugar de después de ellas. Sin embargo, la función de coste presentada en la sección 2.1.1 (véase ecuación 2.5) no resulta adecuada para el aprendizaje *online* en entornos dinámicos. En primer lugar, es necesario realizar una pequeña modificación para que sea aplicable en modo *online*, de manera que el error $\varepsilon_j(s)$ se calcule para un único patrón en cada iteración del aprendizaje (muestra actual). Al medir los errores de esta manera, la función objetivo para la salida j viene dada por [48]:

$$Q_j(s) = \left(f_j^{'}(\bar{d}_j(s))\bar{\varepsilon}_j(s)\right)^2 = \left(f_j^{'}(\bar{d}_j(s))\left(\bar{d}_j(s) - \sum_{i=0}^{I} w_{ji}x_i(s)\right)\right)^2, \qquad (3.1)$$

donde $\bar{d}_j(s) = f_j^{-1}(d_j(s))$ es la salida deseada antes de la función de activación. El empleo de esta función objetivo permite el aprendizaje en entornos donde las muestras de entrenamiento no están disponibles desde el principio. Es decir, permite aprendizaje incremental u *online*, muestra a muestra, ponderando del mismo modo todos los errores cometidos. Este hecho hace que se conceda la misma importancia al conocimiento previo y a la información más reciente. Sin embargo, en caso de que el proceso modifique la función intrínseca que genera los datos o bien existan cambios de contexto, se hace necesario adaptar la red de manera que el nuevo conocimiento pese más que la información antigua. Por este motivo, la función anterior no es válida en un contexto no estacionario y necesita una adaptación. En consecuencia, para cada salida j, se define una nueva función [94] como,

$$\begin{aligned} Q_j(s) &= h_j(s)\left(f_j^{'}(\bar{d}_j(s))\bar{\varepsilon}_j(s)\right)^2 = \\ &= h_j(s)\left(f_j^{'}(\bar{d}_j(s))\left(\bar{d}_j(s) - \sum_{i=0}^{I} w_{ji}x_i(s)\right)\right)^2, \end{aligned} \tag{3.2}$$

donde $h_j(s)$ es una función de olvido que indica la importancia del error para la $s-$ésima muestra. Como función de olvido pueden emplearse diferentes funciones tales como, por ejemplo, la función lineal o la exponencial. La función seleccionada establece la forma y la velocidad de adaptación de la red a los nuevos datos al trabajar en un entorno que varía con el tiempo. Cuando el contexto es estacionario todas las muestras deben ser consideradas de igual modo, es decir los errores asociados tienen la misma importancia, por tanto esta función puede ser constante. Sin embargo, en un contexto variable o estacionario, la función debe ser monótonamente creciente para tener en cuenta el aumento de la importancia asociada a la información actual con respecto al conocimiento pasado. Gracias a la combinación de la propiedad de aprendizaje incremental y la asignación de una importancia creciente a los datos más recientes, la red es capaz de olvidar rápidamente en presencia de un cambio al tratar con entornos muy dinámicos. Al mismo tiempo la red podría seguir manteniendo un comportamiento estable en contextos de trabajo estacionarios.

La función objetivo definida en la ecuación 3.2 continúa siendo una función convexa cuyo óptimo global se puede calcular al derivar la misma con respecto a los parámetros de la red e igualando a cero:

$$\frac{\partial Q_j(s)}{\partial w_{jp}} = -2h_j(s)\left(f_j^{'}(\bar{d}_j(s))\left(\bar{d}_j(s) - \sum_{i=0}^{I} w_{ji}x_i(s)\right)\right)x_p(s)f_j^{'}(\bar{d}_j(s)) = 0,$$

donde $p = 0, \ldots, I$ y s indica la iteración de aprendizaje actual que se corresponde con la muestra de entrenamiento s-ésima. Al reescribir los términos de la ecuación anterior se obtiene un sistema de $(I+1) \times (I+1)$ ecuaciones lineales definido por la expresión:

$$\sum_{i=0}^{I} A_{pi}(s)w_{ji}(s) = b_{pj}(s); p = 0, 1, ..., I; \tag{3.3}$$

con

$$A_{pi}(s) = A_{pi}(s-1) + h_j(s)x_i(s)x_p(s)f_j^{'2}(\bar{d}_j(s)); \tag{3.4}$$

$$b_{pj}(s) = b_{pj}(s-1) + h_j(s)\bar{d}_j(s)x_p(s)f_j^{'2}(\bar{d}_j(s)); \tag{3.5}$$

donde $A(s-1)$ y $b(s-1)$ son las matrices y los vectores que almacenan los coeficientes de los sistemas de ecuaciones lineales obtenidos en las iteraciones previas para calcular los pesos y que permiten realizar el aprendizaje *online*. En otras palabras, los coeficientes empleados para calcular los pesos en la iteración actual se emplean para obtener los pesos en la iteración siguiente. De este modo, se puede manejar el conocimiento previo y emplearlo para aproximar de manera incremental el valor óptimo de los pesos. Como el proceso de aprendizaje comienza sin conocimiento previo, en la iteración inicial del proceso los elementos de las matrices $A(s-1)$ y los vectores $b(s-1)$ de coeficientes toman el valor cero.

La ecuación 3.3 puede reescribirse en notación matricial como,

$$\mathbf{A}_j(s)\mathbf{w}_j(s) = \mathbf{b}_j(s); j = 1, \ldots, J, \tag{3.6}$$

con

$$\mathbf{A}_j(s) = \mathbf{A}_j(s-1) + h_j(s)\mathbf{x}(s)\mathbf{x}^T(s)f_j'^2(\bar{d}_j(s)), \tag{3.7}$$

$$\mathbf{b}_j(s) = \mathbf{b}_j(s-1) + h_j(s)\bar{d}_j(s)\mathbf{x}(s)f_j'^2(\bar{d}_j(s)). \tag{3.8}$$

Finalmente,

$$\mathbf{w}_j(s) = \mathbf{A}_j^{-1}(s)\mathbf{b}_j(s), \forall j. \tag{3.9}$$

Este sistema tiene una única solución excepto para problemas mal condicionados. En el caso de matrices mal condicionadas, el problema puede resolverse aplicando la pseudoinversa de Moore-Penrose [111]. De maneral general se pueden emplear diferentes métodos, computacionalmente eficientes, para resolver este tipo de problemas con complejidad $O(N^2)$, donde N es el número de pesos asociados a cada salida j de la red. Como se puede comprobar, el método propuesto mantiene la complejidad computacional del algoritmo original.

Teniendo en cuenta estas consideraciones el Algoritmo 3.1 detalla el método de aprendizaje *online* y adaptativo, empleando los conceptos descritos hasta ahora.

3.2. Resultados experimentales

El principal objetivo de este estudio experimental es verificar si el método propuesto es capaz de adaptar su respuesta en entornos cambiantes gracias a la incorporación del término de olvido (h) en la función de coste. Con este fin, se ilustra el comportamiento del método mediante su aplicación a diferentes problemas de identificación de sistemas. En el estudio se emplean tres series temporales artificiales, las dos primeras reproducen señales que sufren, respectivamente, cambios bruscos y graduales a lo largo del tiempo. El último conjunto considerado pretende probar el sistema en un entorno casi real, simulando el comportamiento de la vibración de un rodamiento durante su ciclo de vida.

Algoritmo 3.1 Algoritmo de aprendizaje *online* con capacidad adaptativa para redes de neuronas de una capa.

Entradas: $\mathbf{x}(s) = (x_1(s), x_2(s), \ldots x_I(s))$,
$\mathbf{d}(s) = (d_1(s), d_2(s), \ldots d_J(s))$,
sesgo $x_0(s) = 1$

1. $\mathbf{A}_j(0) = \mathbf{0}_{(I+1)\times(I+1)}$, $\mathbf{b}_j(0) = \mathbf{0}_{(I+1)\times 1}$, $\forall j = 1, \ldots, J$
2. Para cada muestra s $(s = 1, \ldots, S)$
3. Para cada salida j de la red $(j = 1, \ldots, J)$
4. $\mathbf{A}_j(s) = \mathbf{A}_j(s-1) + h_j(s)\mathbf{x}(s)\mathbf{x}^T(s)f_j'^2(\bar{d}_j(s))$ (véase ec. 3.7)
5. $\mathbf{b}_j(s) = \mathbf{b}_j(s-1) + h_j(s)\bar{d}_j(s)\mathbf{x}(s)f_j'^2(\bar{d}_j(s))$ (véase ec. 3.8)
6. Calcular $\mathbf{w}_j(s)$ empleando la ecuación 3.9.
7. fin del Para
8. fin del Para

En todos los experimentos se emplea la misma función de olvido (h) para cada una de las J salidas de la red. Con el objeto de comprobar la adecuación de la función de olvido a cambios bruscos y suaves en las señales a modelar se realizaron pruebas con tres aproximaciones diferentes. Específicamente las funciones empleadas fueron:

- La función exponencial,

$$h_j(s) = e^{\mu s}, \quad \forall j \tag{3.10}$$

- La función de distribución acumulativa de *pareto* definida por,

$$h_j(s) = 1 - (1/s)^{\mu}, \quad \forall j \tag{3.11}$$

- Una función polinomial de tipo,

$$h_j(s) = s^{\mu}, \quad \forall j. \tag{3.12}$$

Para todas ellas, μ es un parámetro real positivo que controla el crecimiento de la función y, en consecuencia, la respuesta de la red a los posibles cambios en el contexto. Todas las funciones permiten que el sistema pondere de forma diferente los errores cometidos sobre las muestras analizadas, dando mayor importancia a las muestras recientes pero sin descartar el conocimiento previo, adquirido al procesar las muestras más antiguas.

Los resultados alcanzados por el método propuesto se compararon con los obtenidos por el método de mínimos cuadrados normalizado (*normalized least-mean-square*, NLMS [103]). Este método es una versión mejorada del algoritmo original de mínimos cuadrados (*least-mean-square*, LMS), que fue introducido por Widrow [60, 150] y que incorpora la capacidad de respuesta en entornos dinámicos. La elección del método NLMS para el estudio comparativo se debe a que se trata de uno de los métodos de aprendizaje de referencia dentro de los sistemas adaptativos y ha sido empleado en muchas aplicaciones reales tales como sistemas de filtrado [110] y radares [83]. Merece la pena mencionar

que el algoritmo NLMS presenta dos importantes ventajas con respecto al algoritmo LMS original:

1. Es potencialmente más rápido en cuanto a velocidad de convergencia tanto para muestras correlacionadas como para muestras sin ruido.

2. Presenta un comportamiento estable para un rango conocido de valores del parámetro μ de la función h ($0 < \mu < 2$), independientemente de la correlación estadística de los datos de entrada [53, 103].

Con el objeto de evitar comparaciones sesgadas con respecto a la capacidad lineal del NLMS en la experimentación se emplearon funciones de activación lineales.

3.2.1. Serie 1: entorno no estacionario con cambios bruscos

En este experimento se genera un conjunto de datos artificial formado por cuatro variables de entrada aleatorias que contienen valores procedentes de una distribución normal con media 0 y desviación típica 0,1. La salida deseada se obtiene como una mezcla lineal de las variables de entrada. La mezcla se va modificando en el tiempo cada 500 muestras empleando para tal propósito una fila diferente de la siguiente matriz:

$$M = \begin{pmatrix} 2{,}1 & 1{,}3 & -1{,}1 & 1{,}3 \\ -1{,}1 & -0{,}3 & 0{,}7 & -1{,}1 \\ -0{,}1 & 0{,}2 & 1{,}8 & -2{,}3 \\ -3{,}1 & -1{,}5 & 0{,}4 & 1{,}8 \\ 1{,}5 & -0{,}9 & 1{,}2 & 0{,}6 \end{pmatrix}.$$

La figura 3.3 muestra la señal empleada como salida deseada durante la fase de entrenamiento. Como esta señal evoluciona en el tiempo, se generan varios cambios

de contexto. Las líneas de puntos en la figura 3.3 indican los instantes en los que se produce un cambio de estado. De este modo, se dispone de un conjunto de test diferente para cada uno de los estados generados. El conjunto de test asociado a cada muestra de entrenamiento es aquel que representa el contexto al cual pertenece dicha muestra. Por tanto, el conjunto final contiene 2.500 muestras de entrenamiento y 5 conjuntos de test (cada uno con 500 ejemplos), uno para cada uno de los diferentes estados por los que pasa la señal como consecuencia de los cambios introducidos por la mezcla lineal del conjunto de entrenamiento.

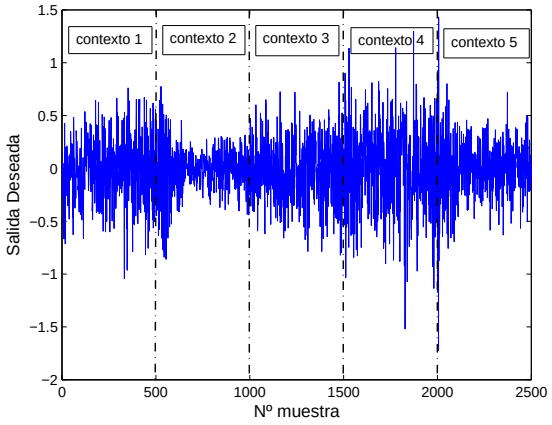
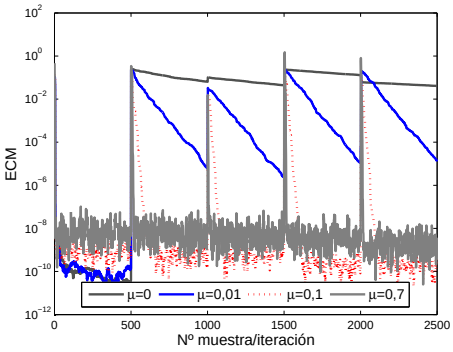


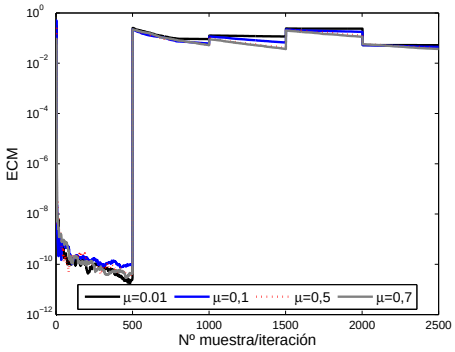
Figura 3.3: Salida deseada para el conjunto de entrenamiento en un entorno no estacionario con cambios bruscos de contexto cada 500 muestras.

El conjunto de entrenamiento generado se emplea en un proceso de aprendizaje *online* en el cual las muestras se proporcionan de una en una a la red con el objeto de adaptar sus pesos muestra a muestra. Por tanto, el número total de iteraciones (*epochs*) de entrenamiento se corresponde con el número de muestras.

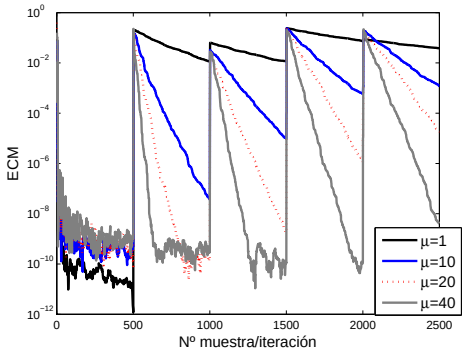
Influencia del factor μ. En primer lugar, resulta de interés comprobar en qué medida influye en el comportamiento del método propuesto el valor del factor μ de la función de olvido. A continuación se presenta una comparativa para algunos de los valores más representativos. Cabe mencionar que el rango de valores del factor μ seleccionado para cada función de olvido depende del comportamiento de cada una de ellas.



(a) exponencial



(b) pareto



(c) polinomial

Figura 3.4: Error de test para distintos valores del factor μ en las tres funciones de olvido (h) empleadas en un entorno no estacionario con cambios bruscos de contexto cada 500 muestras.

En la figura 3.4 se muestran las curvas de ECM obtenidas en la fase de test con diferentes valores del factor μ en cada una de las tres funciones de olvido consideradas. Para cada punto de la señal, el valor mostrado se corresponde con la media del error obtenido sobre el conjunto de test correspondiente a la muestra de entrenamiento actual. Como se puede observar en las gráficas, en este ejemplo concreto no se aprecian diferencias significativas en los resultados obtenidos con la función de *pareto* para distintos valores de μ. Para las funciones exponencial y polinomial, a medida que el valor del factor μ aumenta también lo hace la rapidez de adaptación de la red al nuevo estado del sistema. Sin embargo, y como se puede comprobar, este hecho conlleva

un aumento de la variabilidad del error asociado lo cual es lógico pues se aumenta el peso de los errores cometidos para las últimas muestras y por tanto el sistema es más sensible a cambios puntuales en los datos. Por otro lado, cuando el factor toma valor cero cualquiera de las funciones toma un valor constante y por tanto se está asignando exactamente la misma importancia a los errores asociados a cada una de las muestras. Por este motivo, el sistema no consigue adaptarse a los cambios que van apareciendo durante el proceso de entrenamiento al considerar que las muestras antiguas tienen la misma importancia que las más recientes evitando su flexibilidad frente a nuevos contextos. Este comportamiento se corresponde con el algoritmo de aprendizaje *online* sin factor de olvido.

Finalmente, para las siguientes experimentaciones el valor de μ se establece a 0,7 para la función exponencial, a 0,1 para *pareto*, mientras que en el caso de la función polinomial toma el valor 20.

Selección de la función de olvido. Una vez analizada la influencia del factor μ para cada una de las funciones de olvido, se muestra una breve comparativa de los resultados obtenidos por el método propuesto empleando cada una de ellas.

En la figura 3.5 se muestra el error cuadrático medio obtenido durante el proceso de test por el método propuesto haciendo uso de las distintas funciones de olvido con los valores del factor μ seleccionados previamente. La función exponencial aparece etiquetada como *NN-exp*, la *pareto* como *NN-pareto*, y finalmente la función polinómica se referencia mediante *NN-poly*. Como se puede apreciar en esta figura, con la función *pareto* el método no consigue adaptarse a los distintos cambios de contexto, debido a que la importancia que concede esta función a los errores cometidos sobre las muestras más recientes no es suficiente para permitir que el método adapte su comportamiento adecuadamente. En cuanto a las funciones exponencial y polinómica, ambas muestran un comportamiento adecuado, pero la elección de una u otra depende, en gran medida, de la aplicación en la que se emplee. Cabe destacar, que con la función exponencial

el método presenta una mayor adaptación a los cambios, debido a que corrige el error bruscamente como consecuencia de la elevada importancia que le concede esta función a las muestras recientes. Este hecho implica que pequeñas variaciones entre muestra y muestra pueden suponer cambios significativos en el error. Sin embargo, como se puede observar en el caso de la función polinómica, el método se adapta a los cambios de forma gradual de manera que, ahora, la diferencia de ponderación muestra a muestra no es tan acusada, mostrando una tendencia a adaptarse cada vez menos como se puede comprobar en los picos asociados a cada cambio de contexto. De este modo se consigue una adaptación más suave siendo, a mayores, mucho menor la variabilidad del error alcanzado. Por tanto, en función de la aplicación real en la que se emplee el sistema, será necesario un tipo distinto de adaptación. Por ejemplo, en una aplicación industrial como puede ser el control de una caldera la adaptación es lenta y el uso de una función polinómica sería acertado, ya que consigue una buena adaptación evitando una alta variabilidad. Sin embargo, en aplicaciones típicamente de clasificación, como puede ser la identificación de spam o datos biomédicos, la función exponencial sería más adecuada ya que se desea conseguir la adaptación lo antes posible.

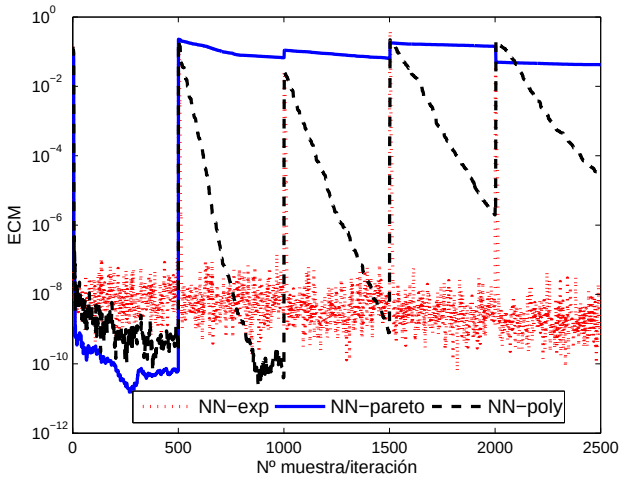
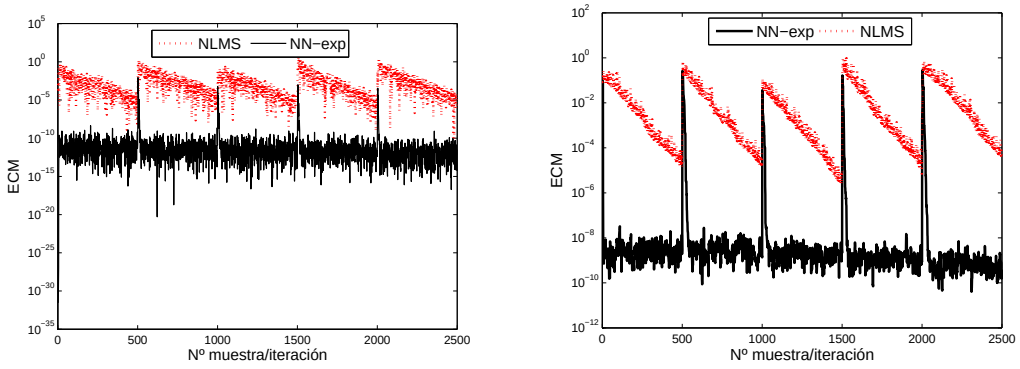


Figura 3.5: Errores de test obtenidos para las tres funciones de olvido empleadas en un entorno no estacionario con cambios bruscos de contexto cada 500 muestras.

En resumen, es importante tener en cuenta que el método propuesto puede optimi-

zarse para una tarea específica ajustando la función de olvido según las características y necesidades del problema a resolver.

Comparativa del método propuesto con el NLMS. Para este caso, se selecciona la exponencial como función de olvido para el método propuesto con un valor de μ=0,7. El método se compara con el algoritmo NLMS con un paso de aprendizaje de 0,9. En la figura 3.6 se muestra el ECM obtenido por ambos métodos durante las fases de entrenamiento y test. Ambos disminuyen su error después de cada uno de los cambios del sistema, pero el mejor comportamiento lo alcanza el método propuesto.



(a) Fase de entrenamiento.

(b) Fase de de test.

Figura 3.6: Curvas de ECM de entrenamiento y test para el método propuesto y el NLMS en un entorno no estacionario con cambios bruscos de contexto cada 500 muestras.

3.2.2. Serie 2: entorno no estacionario con cambio gradual

En esta nueva prueba, el objetivo es comprobar si el método propuesto es capaz de adaptar su respuesta en entornos que presentan cambios de estado graduales a lo largo del tiempo. Para ello se genera un nuevo conjunto de datos cuya distribución presenta continuos cambios como consecuencia de la evolución de los coeficientes de un vector de

mezcla para cada muestra de entrenamiento. El conjunto está formado por 4 variables de entrada, generadas mediante una distribución normal de media cero y desviación típica 0,1, y la salida se obtiene por medio de una mezcla no lineal sobre las mismas fijando los coeficientes iniciales del vector de mezcla como,

$$a(0) = \left[\begin{array}{cccc} 0{,}5 & 0{,}2 & 0{,}7 & 0{,}8 \end{array} \right].$$

y evolucionando estos coeficientes en el tiempo de acuerdo a la siguiente ecuación,

$$a_j(s) = a_j(s-1) + \frac{s}{10^{4,7}} logsig(a_j(s-1)), s = 1, \ldots, S,$$

el subíndice j indica la componente del vector de mezcla. Finalmente, obtenemos un nuevo conjunto de 500 datos cuya salida deseada durante el proceso de entrenamiento puede observarse en la figura 3.7. Debido a que la mezcla presenta cambios constantes se dispone de un contexto distinto para cada muestra, y por tanto, de un conjunto de test diferente para cada muestra de entrenamiento.

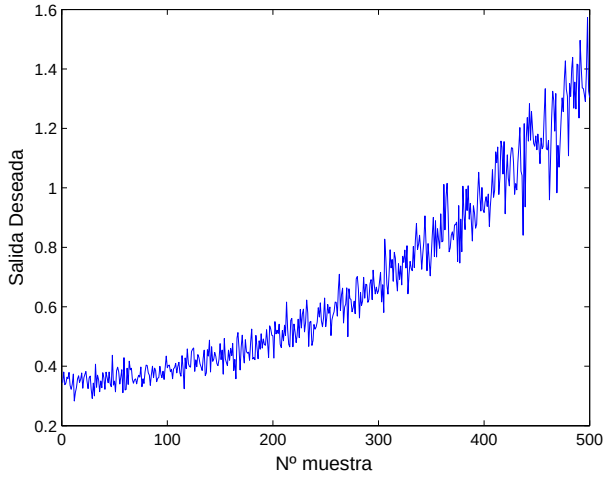


Figura 3.7: Salida deseada para el conjunto de entrenamiento en un entorno no estacionario con cambio gradual continuo.

Al igual que en el ejemplo anterior se analizó el comportamiento del método propuesto según la función de olvido empleada. Finalmente, los valores del factor μ seleccionados para cada función son los especificados para el conjunto de datos previo.

En la figura 3.8 puede comprobarse como en los tres casos el método comienza con un error bajo que crece regularmente. El pico inicial que aparece en el ECM para las tres funciones de olvido se debe a que al principio, la red se adapta perfectamente a los datos ya que existen más parámetros libres que muestras. Sin embargo, a medida que se dispone de más datos, y al mantener el conocimiento obtenido previamente, el error aumenta debido a que la función que genera los datos cambia de manera constante. Las muestras nuevas corresponden a un nuevo modelo de datos, de manera que la red no es capaz de encontrar una función que permita modelar de manera adecuada tanto las muestras antiguas como las recientes, y consecuentemente el error aumenta de manera continua.

En cuanto al comportamiento del método propuesto según la función de olvido empleada, puede verse cómo con la función de pareto alcanza el error más elevado, mientras que con las funciones exponencial y polinómica, el comportamiento es similar al del ejemplo presentado anteriormente. La función exponencial obtiene una mayor adaptación pero también presenta una elevada variabilidad, en cambio con la función polinómica la adaptación es más suave pero la variabilidad es menor. Al igual que para el ejemplo previo, la función de olvido que se empleará en la comparativa siguiente es la exponencial.

La figura 3.9 muestra los resultados obtenidos durante las fases de entrenamiento y test cuando el método propuesto con función de olvido exponencial se compara con el algoritmo NLMS. Como se puede apreciar en ambas fases, el NLMS comienza con un error relativamente elevado que no es capaz de reducir durante el proceso de aprendizaje. Con ambos métodos, el error de test crece continuamente como consecuencia de los continuos cambios que sufre la señal. Sin embargo, este crecimiento es menor con el método propuesto. Además, se puede comprobar cómo la variabilidad de los errores alcanzados por el método propuesto es menor que con el NLMS.

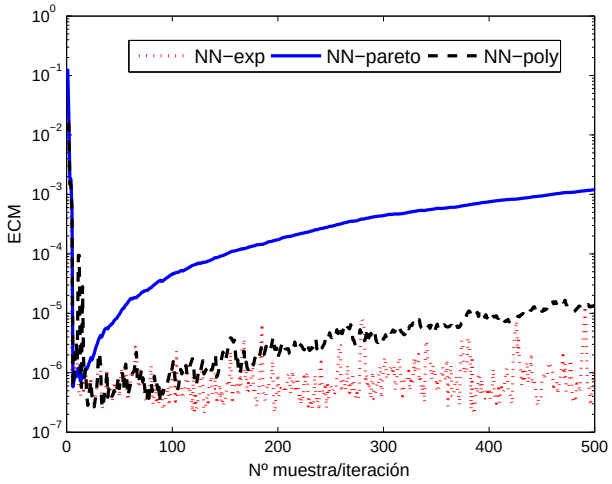
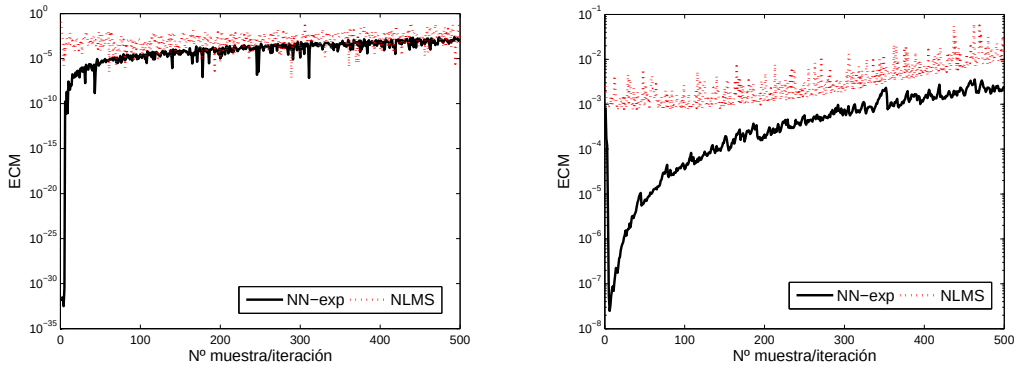


Figura 3.8: Entorno no estacionario con cambio gradual continuo. Errores de test obtenidos para las tres funciones de olvido empleadas.



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 3.9: Curvas de ECM de entrenamiento y test para el método propuesto y el NLMS en un entorno no estacionario con cambio gradual continuo.

3.2.3. Serie 3: datos simulados de la vibración de un rodamiento

El tercer y último problema que se plantea pretende probar el sistema en un entorno casi real y consiste en predecir una serie temporal que simula el comportamiento de la vibración de un rodamiento durante su ciclo de vida [94]. La figura 3.10 muestra la señal de entrada para el modelo predictivo. En la gráfica superior aparece representado

la raíz cuadrada del error cuadrático medio (*root mean squared*, RMS) del nivel de vibración que se registra en un sensor localizado sobre el rodamiento, mientras que en la figura inferior se pueden observar las revoluciones por minuto (RPM) a las que trabaja el componente en cada instante.

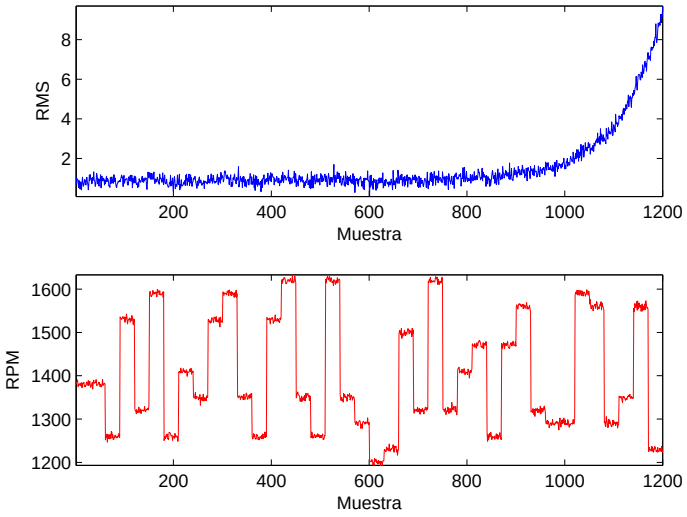


Figura 3.10: Serie simulada de la vibración de un rodamiento. Raíz cuadrada del error cuadrático medio (RMS) de la vibración y revoluciones por minuto (RPM) del rodamiento.

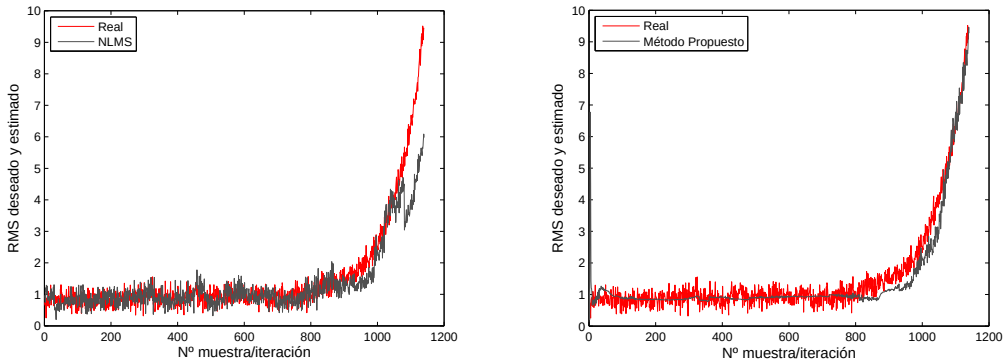
La primera señal se modela por medio de un proceso exponencial definido por la ecuación que ahora se presenta. Esta ecuación intenta modelar el comportamiento típico del desgaste de un componente mecánico.

$$RMS(s) = \frac{e^{s/8}}{(s+100)^{2,4}} + \frac{RPM(s)}{2000} + N(0,0{,}2).$$

Como se puede observar, la señal RMS depende de la carga de trabajo (RPM) en cada instante y además contiene ruido aleatorio con distribución normal. El objetivo que se plantea es predecir en cada instante de tiempo s el valor del RMS en un instante futuro $s+d$ empleando para ello únicamente dos entradas, una muestra previa de la vibración, $RMS(s-1)$, y las revoluciones para un instante futuro, $RPM(s+d)$. En el presente

ejemplo se establece el tamaño de predicción como $d = 30$.

Al igual que en los conjuntos anteriores se realizó un estudio experimental para distintos valores del parámetro μ de la función de olvido y, finalmente se seleccionó el valor de 0,01. Así, en la figura 3.11 se incluye el RMS de test estimado por ambos métodos de aprendizaje. En la figura 3.11(b) se puede observar cómo el método propuesto es capaz de obtener una mejor aproximación. Este hecho es más pronunciado en la última parte de la señal donde la vibración aumenta, indicando de este modo un deterioro considerable del rodamiento.



(a) Resultados para el NMLS.

(b) Resultados para el método propuesto.

Figura 3.11: Curvas del RMS real y estimado por el NLMS y el método propuesto para el conjunto de test de la serie simulada de la vibración de un rodamiento.

3.3. Discusión

En este capítulo se ha presentado un nuevo algoritmo de aprendizaje *online* para redes de neuronas de una sola capa. La principal característica es que el método incluye la capacidad de adaptarse a entornos no estacionarios gracias a la inclusión de un término de olvido en su función de coste. Además, se demuestra experimentalmente que el comportamiento del método es adecuado para su aplicación en problemas no

estacionarios tanto si se producen cambios bruscos o cambios graduales. Las principales ventajas que se pueden extraer sobre el funcionamiento del método se resumen brevemente a continuación:

1. En cuanto a los requisitos temporales, se puede establecer que es un método rápido ya que para cada una de las muestras que forman parte del proceso de entrenamiento, el algoritmo presenta una complejidad de $O(N^2)$, siendo N el número de parámetros adaptativos de la red para cada una de las j salidas de la red.

2. En lo que respecta a las necesidades espaciales, durante el proceso de aprendizaje únicamente es necesario almacenar el patrón actual y, para cada neurona de salida de la red, la matriz de coeficientes $\mathbf{A}$ (véase ecuación 3.7) con $(I+1) \times (I+1)$ elementos y el vector de coeficientes $\mathbf{b}$ (véase ecuación 3.8) con $(I+1)$ elementos, donde I es el número de entradas de la red de neuronas. Por ello podemos decir que el método permite un empleo óptimo de los recursos computacionales disponibles.

3. El método propuesto tiene la característica de ajustar su capacidad de adaptación, por lo que muestra un comportamiento flexible para afrontar los diferentes tipos de cambio que se presenten, tanto graduales como bruscos, en el concepto a aprender, y así proporcionar la mejor respuesta posible en cada instante de tiempo.

4. Como se ha comprobado gracias a los resultados experimentales, el método aprende de la información más reciente a la vez que retiene el conocimiento relevante que ha adquirido previamente de la información más antigua. Esta característica en su comportamiento nos permite hablar de un equilibrio estabilidad-plasticidad.

5. Gracias a estas características, el método resulta aplicable en entornos donde el aprendizaje debe realizarse en tiempo real.

6. No obstante, y dadas sus características de aprendizaje incremental, el método es escalable en lo que se refiere a sus necesidades espaciales. Por este motivo, también resulta adecuado para el aprendizaje en situaciones en las que se trata

con grandes bases de datos, ya que en este caso, podría aplicarse un aprendizaje por lotes.

7. Por último, vale la pena destacar que el método propuesto es una generalización del método previo ya que este último es un caso particular, cuando la función de olvido es constante, del que se presenta en este capítulo. Además, matemáticamente se demuestra que la solución encontrada por el método *online* es la misma a la que llegaría si fuese aplicado en un aprendizaje en modo *batch*.

Capítulo 4

Algoritmo de aprendizaje basado en sensibilidad estadística con regularización

Existen dos situaciones que afectan negativamente a la capacidad de generalización de los sistemas de aprendizaje en general, y a las redes de neuronas en particular:

- Aplicaciones en las que se dispone de un número reducido de ejemplos.
- Datos afectados por ruido.

En el primer caso, la red de neuronas puede tender a memorizar los datos de entrenamiento debido a su alta capacidad no lineal y el número de parámetros libres [21] (problema del sobreentrenamiento o sobreaprendizaje, *overfitting*) mientras que en el segundo, la red puede ajustarse de manera precisa a los datos, incluyendo también el ruido, en lugar de obtener un modelo intrínseco que genere los datos originales sin perturbaciones.

En el Capítulo 2 se comentaban las principales innovaciones aportadas por el algoritmo de aprendizaje denominado SBLLM, así como sus ventajas con respecto a otros métodos. Como se podía observar en la comparativa con otros métodos, el SBLLM presenta una rápida velocidad de convergencia y alcanza un buen error en pocas iteraciones del proceso de aprendizaje. Su comportamiento es adecuado cuando se manejan grandes redes y conjuntos de datos, sin embargo, cuando el número de muestras disponible es reducido o los datos están afectados por ruido, en este algoritmo resulta complicado evitar el problema del sobreaprendizaje empleando simplemente técnicas tales como la de parada temprana. Esto es consecuencia del reducido número de iteraciones que requiere el método SBLLM para alcanzar la convergencia.

La teoría de regularización [21, 52] ha sido ampliamente estudiada y empleada con el objetivo de evitar, al menos en la medida de lo posible, los problemas citados al establecer una solución de compromiso entre la complejidad de la transformación y un buen ajuste de los datos de entrenamiento. Para conseguirlo intenta suavizar las transformaciones que genera la red, en general mediante una función no lineal entre las entradas y las salidas, añadiendo un término de penalización a la función de error. De este modo, una nueva función de error podría definirse como

$$\tilde{E} = E + \alpha R, \tag{4.1}$$

donde E es una función de error típica (por ejemplo, el ECM), α el parámetro que controla el grado de influencia del término de penalización en la forma de la solución y, por tanto la suavidad del ajuste y, por último R es ese término de penalización. Cuando la red sobreajusta las muestras, se obtendrá un valor reducido del error E pero un valor elevado de R, por el contrario, en caso de que la red subajuste se alcanza un valor pequeño de R pero un gran valor del error E.

Entre los métodos de regularización propuestos, el decaimiento de pesos, *weight decay* [21], es uno de los más empleados por su sencillez y eficacia. Este método define el término de regularización R como la suma de los cuadrados de todos los parámetros

de la red (pesos y sesgos), que para una red de neuronas de una capa viene dada por,

$$R = \frac{1}{2} \sum_i \sum_j w_{ij}^2. \tag{4.2}$$

Veamos una justificación heurística del método de regularización *weight decay*. Para llegar a una transformación sobreajustada en regiones de gran curvatura se necesitan valores de los pesos relativamente grandes. Empleando un regularizador de la forma indicada en la ecuación 4.2 se potencia que durante el entrenamiento los pesos tiendan a valores pequeños. Muchos algoritmos de entrenamiento de redes de neuronas emplean las derivadas de la función de error total con respecto a los pesos de la red. De este modo teniendo en cuenta las ecuaciones 4.1 y 4.2 se obtiene,

$$\nabla \tilde{E} = \nabla E + \alpha \nabla R. \tag{4.3}$$

Si se supone ausente el término E y teniendo en cuenta que $\nabla R = w$, entonces

$$\nabla \tilde{E} = \alpha w. \tag{4.4}$$

Considerando que el entrenamiento se lleva a cabo mediante el gradiente descendente, el vector de pesos evoluciona en el tiempo según

$$\begin{aligned} \mathbf{w}(\tau) &= \mathbf{w}(\tau - 1) + \triangle \mathbf{w}^{(\tau)} \\ &= \mathbf{w}(\tau - 1) - \eta \frac{\partial \tilde{E}}{\partial \mathbf{w}^{(\tau)}} \\ &= \mathbf{w}(\tau - 1) - \eta \nabla \tilde{E} \end{aligned} \tag{4.5}$$

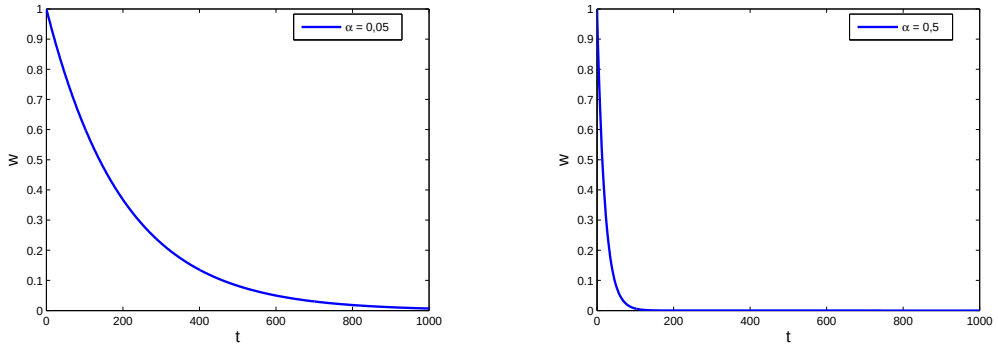
donde η es el paso de aprendizaje. Por tanto,

$$\frac{d\mathbf{w}}{d\tau} = -\eta \nabla \tilde{E} = -\eta \alpha w. \tag{4.6}$$

La ecuación 4.6 tiene como solución

$$\mathbf{w}(\tau) = \mathbf{w}(0) e^{(-\eta \alpha \tau)} \tag{4.7}$$

y, por lo tanto, todos los pesos tienden exponencialmente a cero, tal como se muestra en la figura 4.1, donde también se puede apreciar la influencia del término α.



(a) Evolución vector de pesos para α =0,05.

(b) Evolución vector de pesos para α =0,5.

Figura 4.1: Ejemplo del efecto de la técnica de regularización *weight decay*. Evolución del vector de pesos para el caso unidimensional en función del tiempo y del valor del parámetro α.

Por las razones que se acaban de mencionar, en este capítulo se presenta una versión regularizada del algoritmo SBLLM. Se describe el método presentando las nuevas ecuaciones que incorporan un término de regularización en la función de coste a minimizar. Por último, se desea comprobar si el método propuesto resulta ser una buena aproximación para manejar situaciones favorables para la aparición del problema de sobreentrenamiento. Con este fin se analizará su comportamiento en este tipo de problemas.

A mayores, se hace necesario estimar un valor adecuado del parámetro de regularización que permita que el término de regularización penalice de manera adecuada la función de error según el problema a resolver, con este objetivo se propone una técnica que permite realizar la estimación de manera automática. De este modo, antes de describir el método propuesto se introduce el mecanismo empleado para la estimación automática del parámetro de regularización.

4.1. Estimación automática del parámetro de regularización

En la bibliografía se encuentran diferentes aproximaciones que permiten estimar el valor del parámetro α. Una de las posibles opciones que se plantean, y la más utilizada, consiste en tantear diferentes valores de α y estimar el error cometido para cada uno de ellos bien por corrección del error de entrenamiento mediante algún factor [58, 136, 140], o bien mediante el uso de validación cruzada [141]. El empleo de un conjunto de datos de validación para estimar el error es robusto pero presenta el inconveniente de su lentitud debido a que requiere el ajuste de multitud de modelos. Como alternativa, Weigend [145] planteó un conjunto de reglas heurísticas para modificar el valor del parámetro a lo largo del proceso de entrenamiento. Sin embargo, resulta inviable adaptar tales reglas para la función de coste que se propone.

Durante el desarrollo del SBLLM se persiguió, en todo momento, optimizar el empleo de los recursos computacionales y temporales disponibles. En este trabajo de Tesis se plantea la mejora de la eficacia en la solución de problemas del algoritmo SBLLM, sin perder de vista su eficiencia computacional. Para conseguirlo es necesario incluir alguna técnica que permita llevar a cabo una selección automática adecuada del valor del parámetro de regularización. Por este motivo, en el trabajo presentado en esta Tesis se optó por la aproximación desarrollada por Guo et al. [56]. La justificación de esta elección es sencilla, con esta aproximación únicamente es necesario el conjunto de datos de entrenamiento para estimar el valor del parámetro de regularización. Este hecho permite que el valor del parámetro se optimice de manera conjunta con el error de generalización minimizado.

El trabajo realizado por Guo et al. comprende un complejo desarrollo matemático que se puede analizar detalladamente en [56]. Con el objeto de mostrar, de manera general, la aproximación derivada por los autores a continuación se muestra un breve resumen de su trabajo.

Dado un conjunto de datos $D = \{\mathbf{x}_i, \mathbf{d}_i\}_{i=1}^{S}$, se considera que éste se puede modelar mediante una función de probabilidad. Para un diseño particular, es factible considerar que: 1) $p_\alpha(\mathbf{x}, \mathbf{d})$ es la densidad kernel del conjunto de datos y, 2) la función de probabilidad conjunta $P(\mathbf{x}, \mathbf{d})$ denota la arquitectura del mapeo. La entropía relativa o distancia Kullback-Leiber (KL) [80] entre ambas funciones de probabilidad para este sistema particular se denota mediante la función de coste $J(\alpha, \Theta)$, donde Θ representa el vector de parámetros, asociados a la red neuronal en nuestro caso, y α el parámetro de suavizado. De este modo,

$$J(\alpha, \Theta) = \int\int p_\alpha(\mathbf{x}, \mathbf{d}) ln \frac{p_\alpha(\mathbf{x}, \mathbf{d})}{P(\mathbf{x}, \mathbf{d})} d\mathbf{x} d\mathbf{d}. \tag{4.8}$$

Empleando la notación de Bayes $P(\mathbf{x}, \mathbf{d}) = P(\mathbf{d}|\mathbf{x}, \Theta) p_0(x)$ y, sabiendo que $P(\mathbf{d}|\mathbf{x}, \Theta)$ es la probabilidad condicional y $p_0(x)$ es la función de probabilidad a priori, es posible descomponer la expresión anterior de manera que,

$$J(\alpha, \Theta) = J_1(\alpha, \Theta) + J_2(\alpha) \tag{4.9}$$

siendo

$$J_1(\alpha, \Theta) \equiv -\int\int p_\alpha(\mathbf{x}, \mathbf{d}) ln P(\mathbf{d}|\mathbf{x}, \Theta) d\mathbf{x} d\mathbf{d}, \tag{4.10}$$

$$J_2(\alpha) \equiv \int\int p_\alpha(\mathbf{x}, \mathbf{d}) ln \left(p_{\alpha 0}(\mathbf{x}, \mathbf{d})\right) d\mathbf{x} d\mathbf{d}, \text{con } p_{\alpha 0}(\mathbf{x}, \mathbf{d}) \equiv \frac{p_\alpha(\mathbf{x}, \mathbf{d})}{p_0(\mathbf{x})} \tag{4.11}$$

De acuerdo con el principio de longitud de descripción mínima (*minimum description length*, MDL) [14, 121], es necesario seleccionar el valor del parámetro α de tal manera que se minimice la función J.

En el procedimiento de aprendizaje de los parámetros de la red únicamente está implicado el término J_1, ya que el término J_2 no depende de Θ. Bishop [22] establece que en el caso de estimación por máxima probabilidad, $J_1(\alpha, \Theta)$ se reduce al regularizador Tikhonov de primer orden [133] para redes de neuronas con alimentación hacia delante. De este modo, siendo $\Theta = \mathbf{w}$, el vector de pesos de la red, g la función de activación y

σ la varianza de la función de densidad gaussiana, se obtiene que

$$\begin{aligned} J_1(\alpha,\Theta) \approx & \frac{1}{2S\sigma^2}\sum_{i=1}^{S}\{\parallel \mathbf{d}_i - g(\mathbf{x}_i,\mathbf{w}) \parallel^2 + \alpha[\parallel g^{'}(\mathbf{x},\mathbf{w}) \parallel^2 \\ & - \parallel (\mathbf{d}_i - g(\mathbf{x}_i,\mathbf{w}))\, g^{''}(\mathbf{x}_i,\mathbf{w}) \parallel]\}. \end{aligned} \tag{4.12}$$

Esta última ecuación se puede descomponer de manera que,

$$J_1 \approx J_s + \alpha J_r \tag{4.13}$$

siendo

$$J_s = \frac{1}{2S\sigma^2}\sum_{i=1}^{S} \parallel \mathbf{d}_i - g(\mathbf{x}_i,\mathbf{w})) \parallel^2, \tag{4.14}$$

$$J_r = \frac{1}{2S\sigma^2}\sum_{i=1}^{S}\{\parallel g^{'}(\mathbf{x}_i,\mathbf{w}) \parallel^2 - \parallel (\mathbf{d}_i - g(\mathbf{x}_i,\mathbf{w}))g^{''}(\mathbf{x}_i,\mathbf{w}) \parallel\}, \tag{4.15}$$

donde J_s representa la función de error cuadrático medio, mientras que J_r establece el término de regularización. Para valores suficientemente pequeños del parámetro α de suavizado, J_r se reduce a

$$J_r \approx \frac{1}{2S\sigma^2}\sum_{i=1}^{S} \parallel g^{'}(\mathbf{x}_i,\mathbf{w}) \parallel^2, \tag{4.16}$$

y consecuentemente,

$$\begin{aligned} & J_1 \approx J_s + \alpha J_r = \\ & = \frac{1}{2S\sigma^2}\sum_{i=1}^{S}\{\parallel \mathbf{d}_i - g(\mathbf{x}_i,\mathbf{w}) \parallel^2 + \alpha \parallel g^{'}(\mathbf{x}_i,\mathbf{w}) \parallel^2\}. \end{aligned} \tag{4.17}$$

Para lograr la minimización de $J(\alpha,\mathbf{w})$ con respecto a α, y teniendo en cuenta la ecuación 4.9, los autores llegan a la siguiente expresión de α

$$\alpha = \frac{d_x E_a(h)}{2J_r}, \tag{4.18}$$

donde d_x corresponde a la dimensión de la entrada. Asumiendo que el conjunto de datos disponible es pequeño, es posible obtener finalmente que el término E_a viene dado por,

$$E_a = d_x[1 + (d_x - 1)^2]. \tag{4.19}$$

En consecuencia empleando las ecuaciones 4.16 y 4.18 se obtiene que,

$$h \simeq d_x^2[1 + (d_x - 1)^2]\frac{S\sigma^2}{\sum_{i=1}^{S} \parallel g'(\mathbf{x}_i, \mathbf{w}) \parallel^2}. \tag{4.20}$$

En estimación por máxima verosimilitud (*maximum likelihood*, ML),

$$\sigma^2 = \frac{1}{S}\sum_{i=1}^{S} \parallel \mathbf{d}_i - g(\mathbf{x}_i, \mathbf{w}) \parallel^2, \tag{4.21}$$

por tanto, substituyendo la ecuación 4.21 en la ecuación 4.20 finalmente se obtiene,

$$\alpha \simeq d_x^2[1 + (d_x - 1)^2]\frac{\sum_{i=1}^{S} \parallel \mathbf{d}_i - g(\mathbf{x}_i, \mathbf{w}) \parallel^2}{\sum_{i=1}^{S} \parallel g'(\mathbf{x}_i, \mathbf{w}) \parallel^2}, \tag{4.22}$$

siendo ésta la aproximación que proponen los autores [56] para estimar el valor del parámetro de suavizado. Para su implementación, es necesario encontrar los pesos mediante algún algoritmo de aprendizaje adaptativo. La idea fundamental es la siguiente, al comienzo del entrenamiento se inicializa el parámetro α con un valor pequeño y se obtienen los pesos iniciales $\mathbf{w}$ mediante el algoritmo de aprendizaje que se considere. A partir de este momento el valor del parámetro α se recalcula periódicamente, empleando la ecuación 4.22, durante el aprendizaje. La característica principal de esta aproximación es que la estimación del parámetro de regularización depende únicamente de los datos de entrenamiento, y su valor se optimiza al mismo tiempo que el error minimizado.

Para emplear esta aproximación en el algoritmo SBLLM se hace necesario adaptar la fórmula presentada en la ecuación 4.22. El valor del parámetro α se establece inicial-

mente a un valor pequeño y a lo largo del aprendizaje se recalcula su valor empleando la siguiente formula,

$$\alpha = I^2 \left[1 + (I-1)^2\right] \frac{ECM}{W^2}, \tag{4.23}$$

donde I es el número de entradas, ECM el error cuadrático medio actual y W^2 representa la suma de los cuadrados de todos los elementos de las matrices de pesos de ambas capas de la red de neuronas.

Teniendo en cuenta estas consideraciones, en la siguiente sección se describe de manera detallada la nueva versión del algoritmo SBLLM que incluye la técnica de regularización del decaimiento de pesos y la estimación automática del parámetro de regularización.

4.2. Descripción del algoritmo de aprendizaje SBLLM con regularización

Supongamos una red de neuronas de dos capas con alimentación hacia delante como la que se presenta en la figura 4.2. Como se explicó en el Capítulo 2 la red se considera como la composición de dos redes de neuronas de una capa.

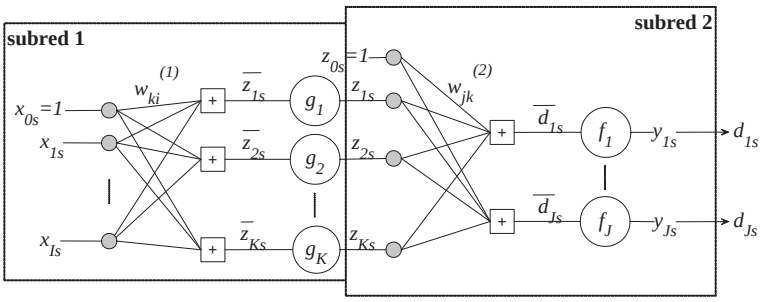


Figura 4.2: Red de neuronas de dos capas con alimentación hacia delante.

De este modo, asumiendo que las salidas de la capa intermedia **z** son conocidas se define una nueva función de coste que incluye un término de regularización para la primera de las subredes como,

$$Q^{(1)} = L^{(1)} + \alpha R^{(1)}, \tag{4.24}$$

donde el término $L^{(1)}$ mide el error de entrenamiento como la suma de los errores al cuadrado antes de las funciones de activación no lineales [30, 43] y se puede escribir como,

$$L^{(1)} = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\bar{\varepsilon}_{ks}\right)^2 = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)^2, \tag{4.25}$$

siendo $\bar{z}_{ks} = g_k^{-1}(z_{ks})$, con $k = 1, \ldots, K$ y $z_{0s} = 1, \forall s$. Con respecto al segundo término de la ecuación 4.24, α es el parámetro de regularización que controla la influencia que ejerce el término de regularización en la función de coste, y $R^{(1)}$ es ese término de regularización, definido como

$$R^{(1)} = \sum_{i=0}^{I}\sum_{k=1}^{K} {w_{ki}^{(1)}}^2. \tag{4.26}$$

Por tanto, la función de coste para la primera capa de la red viene dada por,

$$Q^{(1)} = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)^2 + \alpha\sum_{i=0}^{I}\sum_{k=1}^{K} {w_{ki}^{(1)}}^2. \tag{4.27}$$

De manera análoga, la función objetivo para la segunda de las subredes se define como,

$$Q^{(2)} = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(f_j'(\bar{d}_{js})\left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - \bar{d}_{js}\right)\right)^2 + \alpha\sum_{k=0}^{K}\sum_{j=1}^{J} {w_{jk}^{(2)}}^2, \tag{4.28}$$

donde d_{js} es la salida deseada para la neurona de salida j y $\bar{d}_{js} = f_j^{-1}(d_{js})$. De este modo, la función de coste para la red completa viene dada por la suma de las funciones

objetivo parciales,

$$Q(\mathbf{z}) = Q^{(1)}(\mathbf{z}) + Q^{(2)}(\mathbf{z}) = \tag{4.29}$$
$$\sum_{s=1}^{S}\left[\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)^2 + \sum_{j=1}^{J}\left(f_j'(\bar{d}_{js})\left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - \bar{d}_{js}\right)\right)^2\right] +$$
$$+\alpha\sum_{i=0}^{I}\sum_{k=1}^{K} {w_{ki}^{(1)}}^2 + \alpha\sum_{k=0}^{K}\sum_{j=1}^{J} {w_{jk}^{(2)}}^2.$$

De igual forma que en el método SBLLM original, derivando ambas funciones de coste $Q^{(1)}$ y $Q^{(2)}$ con respecto a los pesos e igualando a cero se calculan los pesos óptimos, al resolver los siguientes sistemas de ecuaciones lineales independientes:

$$\sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} + \alpha w_{kp}^{(1)} = b_{pk}^{(1)}; \quad p = 0, 1, \ldots, I; \quad k = 1, \ldots, K,$$
$$\sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} + \alpha w_{jq}^{(2)} = b_{qj}^{(2)}; \quad q = 0, 1, \ldots, K; \quad j = 1, \ldots, J,$$

donde $A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is} x_{ps} g_k'^2(\bar{z}_{ks})$; $b_{pk}^{(1)} = \sum_{s=1}^{S} \bar{z}_{ks} x_{ps} g_k'^2(\bar{z}_{ks})$ y $A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks} z_{qs} f_j'^2(\bar{d}_{js})$; $b_{qj}^{(2)} = \sum_{s=1}^{S} \bar{d}_{js} z_{qs} f_j'^2(\bar{d}_{js})$.

Posteriormente se calculan las sensibilidades de la nueva función de coste con respecto a las salidas de la capa intermedia z_{ks} de la siguiente manera:

$$\frac{\partial Q}{\partial z_{ks}} = \frac{\partial Q^{(1)}}{\partial z_{ks}} + \frac{\partial Q^{(2)}}{\partial z_{ks}}$$

donde el primer término de la suma viene dado por,

$$\frac{\partial Q^{(1)}}{\partial z_{ks}} = -2\left(g_k'(\bar{z}_{ks})(g_k^{-1})'(z_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)$$
$$\left(g_k'(\bar{z}_{ks}) - g_k''(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right), k = 1, \ldots, K, \forall s,$$

mientras que, el segundo término se define como,

$$\frac{\partial Q^{(2)}}{\partial z_{ks}} = 2\sum_{j=1}^{J}\left(f_j^{'}(\bar{d}_{js})\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - \bar{d}_{js}\right)\right) f_j^{'}(\bar{d}_{js}) w_{jk}^{(2)}, j = 1, \ldots, J, \forall s.$$

Cabe destacar, que las fórmulas de las sensibilidades que se acaban de mostrar son las mismas que en el algoritmo original debido a que el término de regularización no influye en este cálculo. De nuevo, se modifican los valores de las salidas de la capa intermedia empleando la aproximación de la serie de Taylor,

$$Q(\mathbf{z} + \Delta\mathbf{z}) = Q(\mathbf{z}) + \sum_{k=0}^{K}\sum_{s=1}^{S}\frac{\partial Q(\mathbf{z})}{\partial z_{ks}}\Delta z_{ks} \approx 0, \tag{4.30}$$

esto permite obtener los incrementos, $\Delta\mathbf{z} = -\rho\dfrac{Q(\mathbf{z})}{||\nabla Q||^2}\nabla Q$, que se emplean para actualizar los valores z_{ks}.

Teniendo en cuenta las consideraciones que se acaban de mencionar se describe el SBLLM con regularización en el Algoritmo 4.1. Cabe mencionar que las principales diferencias de esta versión del SBLLM con respecto al método original son, en primer lugar, la modificación de las funciones de coste de la red debido a la incorporación del término de regularización y, por otro lado, la inicialización y reevaluación continuada del valor del parámetro de regularización a lo largo del proceso de aprendizaje.

Algoritmo 4.1 Algoritmo SBLLM regularizado con estimación automática del parámetro de regularización.

Entradas: $\mathbf{x}_s = (x_{1s}, x_{2s}, \ldots x_{Is}),\ s = 1, \ldots, S,$
$\mathbf{d}_s = (d_{1s}, d_{2s}, \ldots d_{Js}),$
sesgos $x_{0s} = 1, z_{0s} = 1,$
umbrales de error ϵ y ϵ' para control de convergencia,
paso de aprendizaje ρ.

Salidas: pesos de ambas capas, $\mathbf{W}^{(l)}, l = 1, 2,$
sensibilidades con respecto a datos de entrada y salida.

Fase de inicialización.

- Asignar a las salidas de la capa intermedia la salida asociada con ciertos pesos aleatorios $\mathbf{w}^{(1)}(0)$ más un cierto error, es decir:

$$z_{ks} = f_k^{(1)}\left(\sum_{i=0}^{I} w_{ki}^{(1)}(0) x_{is}\right) + \epsilon_{ks};$$

  donde $\epsilon_{ks} \sim U(-\eta, \eta); k = 1, \ldots, K$, y η es un número pequeño.

- Inicializar también $Q_{anterior}$, $ECM_{anterior}$ y, $\mathbf{z}_{anterior}$ a números grandes, donde ECM mide el error cuadrático medio entre la salida obtenida y la deseada. Estas variables se emplearán para restaurar el sistema a un estado previo cuando tras un cambio en los pesos se produzca un deterioro en el comportamiento.

- Establecer un valor inicial pequeño para el parámetro de regularización α.

Paso 1: Solucionar los subproblemas correspondientes a cada capa, calculando los pesos de ambas capas mediante la resolución de sistemas de ecuaciones lineales independientes para cada una de ellas,

1. Para cada k neurona oculta de la red ($k = 1, \ldots, K$), y siendo $\bar{z}_{ks} = g_k^{-1}(z_{ks})$
2. Calcular $A_{pi}^{(1)}$ como $A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is} x_{ps} g_k'^2(\bar{z}_{ks})$, $p = 0, 1, \ldots, I; k = 1, 2, \ldots, K$.
3. Calcular $b_{pk}^{(1)}$ como $b_{pk}^{(1)} = \sum_{s=1}^{S} \bar{z}_{ks} x_{ps} g_k'^2(\bar{z}_{ks})$, $p = 0, 1, \ldots, I; k = 1, 2, \ldots, K$.
4. Aprender $w_{ki}^{(1)}$, mediante el sistema de ec. lineales $\sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} + \alpha w_{kp}^{(1)} = b_{pk}^{(1)}$.
5. fin del Para.
6. Para cada salida j de la red ($j = 1, \ldots, J$), y siendo $\bar{d}_{js} = f_j^{-1}(d_{js})$,
7. Calcular $A_{qk}^{(2)}$ como $A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks} z_{qs} f_j'^2(\bar{d}_{js})$, $q = 0, 1, \ldots, K; \forall j$
8. Calcular $b_{qj}^{(2)}$ como $b_{qj}^{(2)} = \sum_{s=1}^{S} \bar{d}_{js} z_{qs} f_j'^2(\bar{d}_{js})$, $q = 0, 1, \ldots, K; \forall j$
9. Calcular $w_{jk}^{(2)}$ mediante el sistema de ec. lineales $\sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} + \alpha w_{jq}^{(2)} = b_{qj}^{(2)}$.
10. fin del Para.

Paso 2: Calcular la suma de errores al cuadrado. Empleando los pesos obtenidos en el paso anterior, y asumiendo que las salidas de la capa intermedia **z** son conocidas, se calcula el valor de la función de coste Q para la red completa mediante la ecuación definida en 4.29, y también el ECM a la salida de la red.

Paso 3: Estimación del valor del parámetro de regularización α. Empleando el valor del ECM que ha sido calculado en el paso 2 se obtiene el nuevo valor de α, mediante la ecuación 4.23.

Paso 4: Comprobar la convergencia.

- Si $|Q - Q_{anterior}| < \epsilon$ ó $|ECM_{anterior} - ECM| < \epsilon'$ parar el proceso y devolver los pesos y las sensibilidades.

- En otro caso, el método continua en el paso 5.

Paso 5: Comprobar la mejora de la función de coste Q.

- Si $Q > Q_{anterior}$ se reduce el valor de ρ y se regresa a la situación anterior. Esto quiere decir que se restauran los pesos $\mathbf{z} = \mathbf{z}_{anterior}$, $Q = Q_{anterior}$, y $ECM = ECM_{anterior}$.

- En otro caso, almacenar los valores $Q_{anterior} = Q$, $ECM_{anterior} = ECM$ y $\mathbf{z}_{anterior} = \mathbf{z}$. Calcular además las sensibilidades $\dfrac{\partial Q}{\partial z_{ks}}$ de la función de coste Q con respecto a la salida $\mathbf{z}$ de la capa oculta.

Paso 6: Actualizar las salidas intermedias z, utilizando los siguientes incrementos

$$\Delta \mathbf{z} = -\rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q, \tag{4.31}$$

El procedimiento continua en el Paso 1 hasta que se alcanzan las condiciones de convergencia.

Esta versión del SBLLM mantiene la complejidad computacional del método original, que como se comentó en el Capítulo 2, viene dada por la complejidad del Paso 1 donde se resuelven los sistemas de ecuaciones lineales (uno para cada neurona de salida de cada subred). Existen diferentes métodos eficientes con una complejidad de $O(N^2)$ que pueden emplearse para su resolución, siendo N el número de parámetros de la red neuronal. Por tanto, la complejidad del método de aprendizaje es $O(MN^2)$, siendo $M = K + J$.

4.3. Resultados experimentales

En esta sección se ilustra el comportamiento del método propuesto mediante su aplicación a diferentes problemas de identificación de sistemas. El objetivo del estudio experimental es comprobar si la versión con regularización desarrollada mejora el comportamiento del SBLLM original en aquellas situaciones en las que es posible el problema del sobreajuste a los datos. Para contrastar si existen mejoras sustanciales en el comportamiento del algoritmo, las simulaciones se realizaron para el SBLLM original y la versión propuesta con regularización. Hay que tener en cuenta, que para ello, el método que se propone es una generalización del algoritmo original que se corresponde con el caso concreto de $\alpha = 0$. Con el objeto de obtener resultados comparables, las condiciones de ejecución son las mismas para las distintas aproximaciones del algoritmo SBLLM y para todos los conjuntos de datos. Estas condiciones experimentales se especifican brevemente a continuación,

- Se normaliza el conjunto de datos de entrada (media=0 y desviación típica=1). Además, se añade un ruido aleatorio $\epsilon \in N(0, \sigma)$ a las series temporales (donde σ es la desviación típica de cada conjunto de datos) para comprobar cómo se comporta el método frente al ruido. Bajo estas condiciones se comprueba si el método es capaz de modelar la generalidad de los datos o, por el contrario, se adapta al ruido. Un buen modelo debe filtrar el ruido superpuesto y adaptarse adecuadamente a los datos de entrenamiento.

- En cuanto a las funciones de activación de las neuronas, en todos los casos se emplea una función logística sigmoide para las neuronas de la capa oculta, mientras que para las unidades de la capa de salida se aplican funciones lineales siguiendo las recomendaciones para problemas de regresión [21].

- Con respecto a la topología de las redes hay que indicar que no se pretende encontrar la topología óptima para cada conjunto de datos, sino comprobar los beneficios que se pueden obtener como consecuencia de aplicar la técnica de re-

gularización al algoritmo SBLLM. De este modo, las topologías se eligieron de manera arbitraria buscando en todo momento un número elevado de pesos con respecto al número de ejemplos de entrenamiento con el fin de forzar situaciones favorables para el fenómeno del sobreajuste a los datos.

- El paso de aprendizaje ρ se establece a un valor de $0,2$. A pesar de que este parámetro puede tomar valores dentro del rango $(0, 1]$, se emplea el mismo valor para todas las simulaciones realizadas.

- Con el objeto de obtener resultados significativos se realizaron 5 simulaciones para cada experimento, utilizando en cada una de ellas diferentes conjuntos de valores iniciales de $\mathbf{z}$. Los resultados presentados son la media de las simulaciones realizadas.

- Finalmente, se realizaron tests estadísticos para comprobar si las diferencias encontradas en los resultados son significativas. Para ello, en primer lugar hay que conocer si las muestras siguen una distribución normal mediante el test de Kolmogorov-Smirnov [95]. Si el test no resulta ser significativo se aplica el test Anova [46]. En caso contrario, se emplea su homólogo no paramétrico Kruskal-Wallis [51, 64]. Ambos permiten comprobar la hipótesis de que todas las medias de las medidas de rendimiento son iguales. En todos los casos, se empleó un nivel de significación de 0,05.

En la tabla 4.1 se presentan los conjuntos de datos empleados en la experimentación junto con sus principales características. Se pueden observar cinco columnas que, respectivamente, indican el nombre del conjunto de datos, una breve descripción del mismo, el número total de muestras considerado, la estructura de red empleada para resolver el problema y la técnica de validación que se aplica para obtener un resultado de rendimiento estadísticamente fiable, una validación cruzada de 10 paquetes (*VC 10-paquetes*) ó validación cruzada dejando una muestra fuera (*leave one out*, *LOO*).

Datos	Descripción	Tamaño	Topología	Validación
Lorenz [142]	serie caótica generada mediante código	500	8-10-1	VC 10-paquetes
K.U. Leuven [142]	serie Laser, competición Santa Fe	500	8-10-1	VC 10-paquetes
Mackey-Glass [142]	serie caótica generada por código	50	5-12-1	LOO
Henon [142]	serie caótica generada por código	150	7-15-1	VC 10-paquetes
Dow-Jones [3]	valores índice bursátil Dow-Jones	500	8-10-1	VC 10-paquetes
Kobe [71]	registros terremoto en Tasmania	100	5-10-1	VC 10-paquetes
CO2 [71]	medidas de CO2 en Mauna Loa (Hawaii)	40	8-15-1	LOO
Oscillation [71]	presión en superficie mar Darwin-Tahiti	20	4-15-1	LOO
Blowfly [71]	población moscas azules en cristal	50	7-10-1	LOO
TreeRings [71]	ancho anillos árboles	110	5-12-1	VC 10-paquetes
Waves [71]	fuerzas cilindro suspendido en tanque	150	7-15-1	VC 10-paquetes

Tabla 4.1: Características de los conjuntos de datos empleados en la comparativa del SBLLM original y su versión con regularización.

En la tabla 4.2 se presentan las medidas de rendimiento empleadas en el estudio. Los resultados corresponden a los valores alcanzados por el SBLLM sin regularización y su versión regularizada en las fases de entrenamiento y test. Para el método regularizado, el valor final estimado para el parámetro de regularización se presenta en la columna etiquetada como $\alpha_{\text{ópt.}}$. A la vista de los resultados presentados se puede comprobar como, para todos los conjuntos de datos, el uso del parámetro de regularización α, y en consecuencia la aplicación de la técnica de regularización (*weight decay*) mejora el rendimiento del SBLLM original. Además, con el objeto de comprobar si estas mejoras en el comportamiento del método son estadísticamente significativas, se aplicó el test de Kruskal-Wallis. En todos los casos, el resultado del test indicó que se podía rechazar la hipótesis de que los errores medios de ambas aproximaciones eran iguales. Este resultado nos permite asegurar que existe una mejora en el comportamiento del algoritmo regularizado frente a la versión original sin regularización.

A mayores, el comportamiento de la versión regularizada del SBLLM se muestra de manera gráfica mediante un ejemplo representativo de todos los conjuntos de datos, la serie temporal *Oscillation*. En la figura 4.3 se representan las curvas medias del ECM en función del valor del parámetro de regularización tanto para el proceso de entrenamiento como para el de test. Cabe destacar que en las curvas de error, el méto-

		Entrenamiento			Test	
		$ECM_{\alpha=0}$	$ECM_{\alpha_{\text{ópt.}}}$	$\alpha_{\text{ópt.}}$	$ECM_{\alpha=0}$	$ECM_{\alpha_{\text{ópt.}}}$
Datos	*Lorenz*	$7{,}90 \times 10^{-3}$	$8{,}00 \times 10^{-3}$	$4{,}78 \times 10^{-4}$	$1{,}45 \times 10^{-4}$	$8{,}52 \times 10^{-5}$
	K.U. Leuven	$9{,}20 \times 10^{-3}$	$9{,}30 \times 10^{-3}$	$4{,}28 \times 10^{-4}$	$2{,}13 \times 10^{-4}$	$1{,}01 \times 10^{-4}$
	Mackey	$1{,}21 \times 10^{-2}$	$1{,}26 \times 10^{-2}$	$2{,}17 \times 10^{-2}$	$1{,}80 \times 10^{-3}$	$1{,}20 \times 10^{-3}$
	Henon	$2{,}61 \times 10^{-2}$	$2{,}62 \times 10^{-2}$	$2{,}24 \times 10^{-2}$	$5{,}50 \times 10^{-3}$	$5{,}40 \times 10^{-3}$
	Dow-Jones	$1{,}17 \times 10^{-2}$	$1{,}16 \times 10^{-2}$	$1{,}00 \times 10^{-3}$	$6{,}82 \times 10^{-4}$	$5{,}93 \times 10^{-4}$
	Kobe	$2{,}64 \times 10^{-2}$	$2{,}66 \times 10^{-2}$	$1{,}40 \times 10^{-3}$	$1{,}40 \times 10^{-3}$	$8{,}59 \times 10^{-4}$
	CO2	$1{,}02 \times 10^{-2}$	$1{,}22 \times 10^{-2}$	$9{,}78 \times 10^{-2}$	$4{,}50 \times 10^{-3}$	$2{,}10 \times 10^{-3}$
	Oscillation	$1{,}13 \times 10^{-2}$	$1{,}21 \times 10^{-2}$	$2{,}60 \times 10^{-2}$	$1{,}53 \times 10^{-2}$	$1{,}07 \times 10^{-2}$
	Blowfly	$1{,}27 \times 10^{-2}$	$1{,}27 \times 10^{-2}$	$2{,}80 \times 10^{-3}$	$5{,}10 \times 10^{-3}$	$3{,}60 \times 10^{-3}$
	TreeRings	$2{,}91 \times 10^{-2}$	$3{,}00 \times 10^{-2}$	$3{,}00 \times 10^{-2}$	$9{,}40 \times 10^{-3}$	$8{,}10 \times 10^{-3}$
	Waves	$1{,}23 \times 10^{-2}$	$1{,}24 \times 10^{-2}$	$3{,}10 \times 10^{-3}$	$2{,}40 \times 10^{-3}$	$2{,}05 \times 10^{-3}$

Tabla 4.2: Comparativa del SBLLM original y su versión regularizada con estimación automática de α para conjuntos de regresión. Valores del ECM obtenidos sin regularización ($\alpha = 0$) y con el valor óptimo de α.

do original es equivalente al del punto $\alpha = 0$. Como se puede observar en la gráfica el método original obtiene un error de test elevado pero al mismo tiempo, alcanza un error de entrenamiento bajo. Este hecho es consecuencia del problema del sobreajuste a los datos, el método alcanza buenos resultados en el entrenamiento pero no consigue una buena generalización para los datos del conjunto de test. Sin embargo, a medida que el valor de α aumenta se observa cómo el error de entrenamiento crece progresivamente debido a la disminución del problema del *overfitting*. Al mismo tiempo, el error de test desciende de manera monótona hasta un cierto valor de α momento en que éste es lo suficientemente elevado como para causar una degradación en el rendimiento del método. Este hecho es consecuencia de la excesiva influencia del término de regularización en la función de coste. Para esta serie temporal el valor óptimo de α viene dado por 0,026.

Hasta ahora se ha comprobado el comportamiento de la versión del SBLLM regularizada y su rendimiento se comparó con el método original. Una vez que se ha

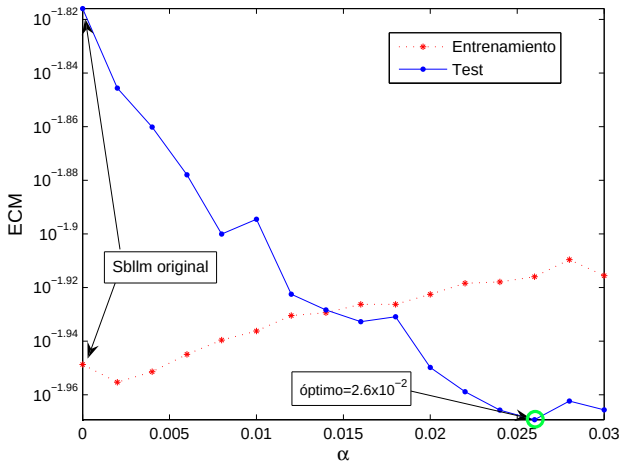


Figura 4.3: Conjunto de datos Oscillation. Curva media del ECM para las fases de entrenamiento y test en función del valor del parámetro de regularización α.

demostrado que el método con regularización mejora el comportamiento del SBLLM original en situaciones en las que es posible el fenómeno del sobreajuste a los datos, la atención se centra ahora en comprobar si la técnica de estimación automática del valor del parámetro de regularización del método propuesto es adecuada. Por tanto, para finalizar el estudio experimental se comparan los resultados obtenidos mediante estimación manual (validación cruzada) y la estimación automática propuesta. El empleo de la técnica de validación cruzada implica tantear diferentes valores del parámetro y estimar el error cometido para cada uno de ellos. El valor del parámetro que se selecciona como óptimo es aquel que obtiene el menor error de validación. A pesar del inconveniente de su lentitud y su elevado coste computacional (requiere entrenar varios modelos), se selecciona el método de validación cruzada por su robustez. Debido al coste computacional de la técnica de validación cruzada, para esta última parte del estudio se seleccionan únicamente algunos de los conjuntos de datos empleados en la experimentación previa. Los conjuntos de datos seleccionados corresponden a las series temporales de *Lorenz*, *K.U. Leuven*, *Dow-Jones*, y *Kobe*. Cabe mencionar que para esta comparativa se mantienen las condiciones experimentales comentadas previamente.

De este modo, en la tabla 4.3 se muestran los resultados obtenidos por el método regularizado cuando el valor de α se estima mediante validación cruzada, y de manera automática con la aproximación desarrollada. A la vista de los resultados presentados vale la pena comentar dos cuestiones de interés que son comunes a las cuatro series temporales. En primer lugar, en cuanto al valor estimado para el parámetro de regularización es posible observar cómo independientemente de la técnica empleada, ambas aproximaciones obtienen valores similares, del mismo orden en todos los casos. Por otro lado, con respecto a las medidas de rendimiento obtenidas se comprueba como los valores ECM medios alcanzados son también muy similares tanto para la fase de entrenamiento como para la de test.

Por tanto, se puede concluir que la versión regularizada con estimación automática del parámetro de regularización proporciona una buena solución de compromiso entre el rendimiento alcanzado en entrenamiento y test, el valor del parámetro α y la demanda de recursos computacionales. Esta última característica es muy importante, ya que la estimación manual es crítica en este aspecto. Como ejemplo se puede mencionar que para la serie temporal de *Dow-Jones* el tiempo medio de CPU, en segundos, que necesita la técnica de validación cruzada para completar el proceso de aprendizaje es de 111,34 mientras que, con el método propuesto es suficiente únicamente con 1,74.

4.4. Discusión

En este capítulo se ha presentado una versión regularizada del método de aprendizaje lineal basado en sensibilidad estadística, al añadir un término de regularización en su función de coste. Esta modificación permite mantener su rendimiento en aquellas situaciones en las que el conjunto de entrenamiento no es suficientemente representativo del problema a resolver o cuando se dispone de un conjunto limitado de muestras, evitando o paliando de este modo el problema del sobreentrenamiento. El algoritmo de aprendizaje original se modifica para incluir la técnica de regularización del decai-

miento de pesos. La estimación del parámetro de regularización se realiza de manera automática por medio de una sencilla fórmula. Este valor se recalcula de manera continua a lo largo del proceso de aprendizaje. Dicha estimación se obtiene gracias a una adaptación de la aproximación desarrollada por Guo et al. [56].

El estudio experimental realizado ha permitido comprobar cómo el método regularizado mejora el rendimiento del algoritmo original en aquellas situaciones en las que es posible el problema del sobreentrenamiento. Los tests estadísticos confirman que, en situaciones como éstas, la versión regularizada obtiene mejores resultados que los que alcanza el algoritmo original. Una vez que se ha comprobado que el comportamiento del algoritmo es adecuado en situaciones de posible sobreajuste a los datos, se ha verificado si la estimación del parámetro de regularización realizada por el método propuesto es adecuada. Con este objetivo se realiza una comparativa con una estimación manual del valor de α mediante la técnica de validación cruzada. Este método no es una aproximación óptima debido a su excesivo coste computacional, ya que implica establecer de manera manual un rango para los posibles valores del parámetro de regularización, ejecutar el algoritmo para cada uno de ellos y seleccionar el valor de α que alcanza el error de validación mínimo, sin embargo, se selecciona en primer lugar por su robustez y en segundo lugar porque permite analizar de manera detallada la evolución del comportamiento del método en función de los distintos valores que toma el parámetro. De este manera se comprueba como el método propuesto obtiene una buena aproximación en cuanto al valor óptimo estimado de α y también con respecto al rendimiento. A mayores, como cabe esperar, la aproximación propuesta presenta una demanda computacional considerablemente inferior debido a que la técnica de validación cruzada requiere entrenar varios modelos. Por tanto, el método desarrollado permite alcanzar buenos resultados optimizando el uso de los recursos computacionales y temporales disponibles.

		$\alpha_{opt.}$		$ECM_{Entrenamiento}$		ECM_{Test}	
		CV	Autom.	CV	Autom.	CV	Autom.
Datos	*Lorenz*	$9{,}59 \times 10^{-4}$	$4{,}78 \times 10^{-4}$	$8{,}00 \times 10^{-3} \pm 8{,}68 \times 10^{-8}$	$7{,}97 \times 10^{-3} \pm 1{,}48 \times 10^{-7}$	$6{,}67 \times 10^{-5} \pm 3{,}92 \times 10^{-7}$	$8{,}52 \times 10^{-5} \pm 1{,}69 \times 10^{-6}$
	K. U. Leuven	$4{,}27 \times 10^{-4}$	$4{,}28 \times 10^{-4}$	$9{,}00 \times 10^{-3} \pm 3{,}74 \times 10^{-6}$	$9{,}30 \times 10^{-3} \pm 1{,}79 \times 10^{-6}$	$9{,}03 \times 10^{-5} \pm 5{,}80 \times 10^{-7}$	$1{,}01 \times 10^{-4} \pm 1{,}50 \times 10^{-6}$
	Dow-Jones	$2{,}30 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	$1{,}17 \times 10^{-2} \pm 1{,}20 \times 10^{-6}$	$1{,}16 \times 10^{-2} \pm 9{,}02 \times 10^{-7}$	$5{,}77 \times 10^{-4} \pm 1{,}17 \times 10^{-6}$	$5{,}93 \times 10^{-4} \pm 6{,}69 \times 10^{-6}$
	Kobe	$1{,}43 \times 10^{-3}$	$1{,}40 \times 10^{-3}$	$2{,}64 \times 10^{-2} \pm 1{,}44 \times 10^{-5}$	$2{,}66 \times 10^{-2} \pm 1{,}14 \times 10^{-5}$	$7{,}27 \times 10^{-4} \pm 1{,}45 \times 10^{-5}$	$8{,}59 \times 10^{-4} \pm 3{,}13 \times 10^{-5}$

Tabla 4.3: Comparativa de los resultados obtenidos mediante las técnicas de validación cruzada y automática para conjuntos de regresión. Media y desviación típica de los valores ECM para las fases de entrenamiento y test.

Capítulo 5

Algoritmo de aprendizaje *online* para redes multicapa en entornos no estacionarios

En un número importante de problemas reales, los algoritmos de aprendizaje automático actúan en entornos dinámicos donde los datos de entrenamiento fluyen de manera continua o llegan en bloques (*batches*) separados en el tiempo, como por ejemplo en análisis de datos financieros o meteorológicos, protección del fraude de tarjetas bancarias, monitorización de tráfico, comportamiento predictivo de clientes, etc. [42, 143]. Por tanto, la información implicada en el aprendizaje no está disponible desde el principio del proceso sino que se recibe continuamente y debe ser procesada secuencialmente y en *tiempo real*. Esta nueva información puede afectar al modelo aprendido y por tanto, los algoritmos de aprendizaje deben ser capaces de adaptarse de manera dinámica cuando llegan nuevos datos. Los algoritmos clásicos de aprendizaje *batch* no son apropiados para manejar situaciones como éstas, ya que reaprenden el concepto desde el principio, empezando una nueva sesión de aprendizaje que considera todas las observaciones disponibles tanto antiguas como recientes [44]. Esta aproximación presenta

varios problemas, siendo el más importante su elevada demanda de recursos computacionales (tanto espaciales como temporales), un handicap importante a la hora de tratar conjuntos de datos de ciertas dimensiones ya que su manejo puede ser inviable si los recursos computacionales son limitados. Una alternativa adecuada son las técnicas de aprendizaje *online* que asumen que la información disponible en cualquier momento es incompleta y susceptible de cambios. Algunas de las ventajas más importantes que presenta el modo de aprendizaje *online* con respecto al *batch* son [84]:

- Un sistema de aprendizaje *online* actualiza sus hipótesis cada vez que recibe una nueva muestra sin que sea necesario reexaminar patrones antiguos.
- Es crucial para sistemas de aprendizaje que reciben muestras continuamente y deben procesar la información en tiempo real.
- Puede emplearse para tratar cambios de tendencia.
- A menudo disminuye el tiempo de convergencia, particularmente cuando los conjuntos de datos son grandes y contienen información redundante.
- Su demanda de recursos espaciales y temporales es considerablemente menor, debido a que no necesita almacenar ni reprocesar datos pasados.

El problema del aprendizaje se puede complicar todavía más debido a que, como ocurre en algunas aplicaciones del mundo real, la salida deseada puede evolucionar en el tiempo. Cabe la posibilidad de que el proceso no sea estrictamente estacionario representando, de este modo, un reto considerable para los sistemas de aprendizaje. Un sistema de aprendizaje *online* se puede aplicar en dominios en los que la distribución evoluciona en el tiempo debido a que, como se especificó anteriormente, una de las características de este tipo de aprendizaje es que permite tratar cambios de tendencia. En entornos no estacionarios, el sistema no deberá considerar igualmente todas las muestras de entrenamiento ya que es posible que solamente las muestras más recientes sean relevantes para el contexto actual de la salida deseada. Si el cambio de contexto es

considerable el sistema necesita olvidar, o ponderar de forma distinta, las muestras más antiguas y ajustar el modelo que se deriva de ellas. El tema es más complejo de lo que aparenta ya que, en dominios que varían en el tiempo, el sistema necesitará modificar la representación interna no sólo al aumentar el número de muestras disponibles, sino también en respuesta a los cambios en la definición del contexto a aprender [79].

El método de aprendizaje basado en sensibilidad estadística SBLLM, que se ha presentado en el Capítulo 2 para redes de neuronas de dos capas (véase figura 2.2), trabaja en modo *batch*, ya que en cada iteración del proceso de entrenamiento la modificación de los pesos depende del conjunto completo de datos (véase sección 2.2.1). Debido a la importancia del aprendizaje *online*, y considerando las características favorables del SBLLM, se plantea el desarrollo de una versión *online* del mismo, con el objetivo de obtener una adaptación que permita el aprendizaje en tiempo real. Sin embargo, al intentar el desarrollo de la versión *online* se plantea una cuestión importante. Ahora, en cada iteración de la fase de entrenamiento se emplea un único patrón, de manera que las sensibilidades puntuales del error para cada muestra de entrenamiento s con respecto a las entradas, definidas por

$$\frac{\partial Q^{(1)}}{\partial z_{ks}} = -2 \quad \left(g_k^{'}(\bar{z}_{ks})(g_k^{-1})^{'}(z_{ks}) \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks} \right) \right) \left(g_k^{'}(\bar{z}_{ks}) - g_k^{''}(\bar{z}_{ks}) \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks} \right) \right),$$

y también las sensibilidades con respecto a las salidas de la capa intermedia, definidas por

$$\frac{\partial Q^{(2)}}{\partial z_{ks}} = 2 \sum_{j=1}^{J} \left(f_j^{'}(\bar{d}_{js}) \left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - \bar{d}_{js} \right) \right) f_j^{'}(\bar{d}_{js}) w_{jk}^{(2)},$$

toman valores pequeños. Consecuentemente, el empleo de estas sensibilidades para obtener los incrementos $\Delta \mathbf{z} = -\rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q$, que permiten actualizar las salidas de la capa oculta $\mathbf{z} = \mathbf{z} - \Delta \mathbf{z}$, provoca que estas modificaciones sean prácticamente nulas. Este hecho conlleva un deterioro importante del rendimiento del método debido a que

las salidas $\mathbf{z}$ de la capa oculta son las que permiten ir aproximando los pesos de la red a sus valores óptimos. Por tanto, se ha comprobado que en el caso del aprendizaje *online* el uso de las sensibilidades no proporciona beneficios a la hora de actualizar las salidas de la capa oculta debido a que éstas se mantienen prácticamente constantes entre iteraciones. Este hecho obliga a plantear diferentes alternativas para solucionar el problema.

Considerando las características que han de tener los sistemas de aprendizaje para satisfacer las necesidades de las aplicaciones reales adaptativas, el método de aprendizaje *online* propuesto en este capítulo incluye un término de olvido en su función de coste, al igual que se presentó previamente para la red de una capa, (veáse Capítulo 3). Esta función permite asignar una importancia creciente a los nuevos datos, facilitando que la red olvide en presencia de un cambio, mientras mantiene un comportamiento estable cuando el contexto es estacionario. En definitiva, en este capítulo se propone un nuevo algoritmo de aprendizaje *online* para redes de neuronas de dos capas con alimentación hacia delante, con capacidad de adaptación para trabajar en entornos no estacionarios.

5.1. Descripción del método propuesto

Supongamos una red de neuronas de dos capas con alimentación hacia delante como la que se presenta en la figura 5.1. Como se indicó con anterioridad, la red se considera como la composición de dos redes de una sola capa. Teniendo en cuenta la explicación presentada en el Capítulo 2, en este contexto los pesos se calculan de manera independiente minimizando para cada una de las capas l una función de coste, $Q^{(l)}$. Esta función mide el error de entrenamiento como la suma de los errores al cuadrado antes de las funciones de activación no lineales (véanse g_k y f_j en la figura 5.1), en lugar de después de ellas como ya se comentó previamente. Sin embargo, las funciones de coste presentadas en la sección 2.2 (véanse ecuaciones 2.9 y 2.10) son para aprendizaje *batch*.

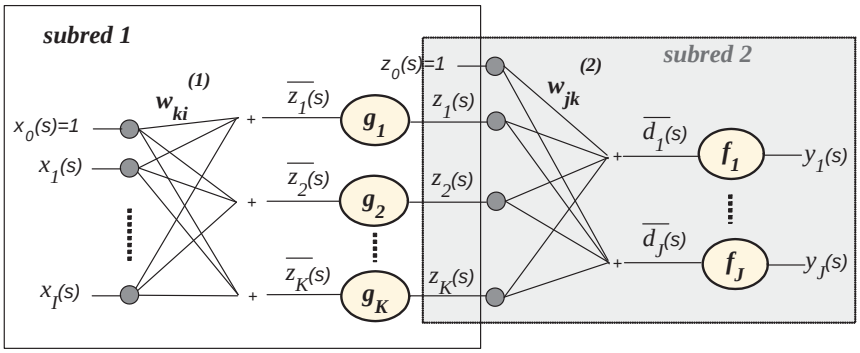


Figura 5.1: Red de neuronas de dos capas con alimentación hacia delante.

Para trabajar en modo *online* se plantea la misma aproximación pero considerando un único patrón en cada iteración (muestra actual). De esta manera, la función de coste para cada salida k de la primera capa de la red viene dada por,

$$Q_k^{(1)}(s) = g_k^{'}(\bar{z}_k(s)) \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_i(s) - \bar{z}_k(s) \right)^2 , \tag{5.1}$$

mientras que para cada salida j de la segunda de las subredes,

$$Q_j^{(2)}(s) = f_j^{'}(\bar{d}_j(s)) \left(\sum_{k=0}^{K} w_{jk}^{(2)} z_k(s) - \bar{d}_j(s) \right)^2 . \tag{5.2}$$

El empleo de estas funciones de coste permite el aprendizaje muestra a muestra, ponderando del mismo modo todos los errores cometidos y concediendo la misma importancia tanto al conocimiento previo como a la información más reciente. Sin embargo, si el proceso modifica la función intrínseca que genera los datos o existen cambios de contexto, es necesario adaptar la red para que conceda mayor importancia al conocimiento reciente. Por este motivo es necesario adaptar la función anterior para que sea válida en un entorno no estacionario. Por tanto, siendo $z_k(s)$ la salida deseada para la neurona oculta k y el patrón de entrenamiento s y $\bar{z}_k(s) = g_k^{-1}(z_k(s))$, se propone una nueva función de coste para resolver ambas subredes. Así, la función objetivo para cada salida de la primera subred se puede escribir como,

$$Q_k^{(1)}(s) = h_k(s) \left(g_k^{'}(\bar{z}_k(s)) \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_i(s) - \bar{z}_k(s) \right) \right)^2 . \tag{5.3}$$

De manera análoga, para cada salida de la segunda subred la función de coste se define del siguiente modo,

$$Q_j^{(2)}(s) = h_j(s)\left(f_j'(\bar{d}_j(s))\left(\sum_{k=0}^{K} w_{jk}^{(2)} z_k(s) - \bar{d}_j(s)\right)\right)^2. \tag{5.4}$$

Los términos $h_j(s)$ y $h_k(s)$ determinan la importancia del error en la $s-ésima$ muestra y tienen el efecto de una función de olvido. Estas funciones permiten establecer la forma y la velocidad de adaptación a las muestras recientes en un contexto de trabajo no estacionario. Vale la pena mencionar que en un entorno estacionario debería emplearse una función constante, para dar la misma importancia a todos los datos analizados durante el proceso de aprendizaje. Por contra, en un entorno no estacionario la función debería ser monótona creciente para tener en cuenta el aumento de la importancia de la información más reciente en contraposición con la más antigua. Como demostrarán los resultados, gracias a la combinación de la propiedad de aprendizaje incremental y la asignación de la importancia creciente, la red mantiene un comportamiento estable cuando el contexto es estático y al mismo tiempo es capaz de olvidar rápidamente en presencia de un cambio.

Al igual que en el SBLLM, las funciones de coste para ambas subredes (ecuaciones 5.3 y 5.4) son convexas con un óptimo global que se puede obtener de manera sencilla al derivarlas con respecto a los pesos e igualando las derivadas a cero. En este caso se obtienen los siguientes sistemas de ecuaciones lineales,

$$\sum_{i=0}^{I} A_{pi}^{(1)}(s) w_{ki}^{(1)}(s) = b_{pk}^{(1)}(s); \quad p = 0, 1, \ldots, I; \quad k = 1, \ldots, K; \tag{5.5}$$

$$\sum_{k=0}^{K} A_{qk}^{(2)}(s) w_{jk}^{(2)}(s) = b_{qj}^{(2)}(s); \quad q = 0, 1, \ldots, K; \quad j = 1, \ldots, J; \tag{5.6}$$

que permiten obtener los pesos de la primera capa y segunda capa, respectivamente, y donde

$$A_{pi}^{(1)}(s) = A_{pi}^{(1)}(s-1) + h_k(s) x_i(s) x_p(s) g_k'^2(\bar{z}_k(s)), \tag{5.7}$$

$$b_{pk}^{(1)}(s) = b_{pk}^{(1)}(s-1) + h_k(s) \bar{z}_k(s) x_p(s) g_k'^2(\bar{z}_k(s)), \tag{5.8}$$

$$A_{qk}^{(2)}(s) = A_{qk}^{(2)}(s-1) + h_j(s)z_k(s)z_q(s)f_j'^2(\bar{d}_j(s)), \tag{5.9}$$

$$b_{qj}^{(2)}(s) = b_{qj}^{(2)}(s-1) + h_j(s)\bar{d}_j(s)z_q(s)f_j'^2(\bar{d}_j(s)), \tag{5.10}$$

A su vez, $b^{(l)}(s-1)$ y $A^{(l)}(s-1)(l=1,2)$ son, respectivamente, los vectores y las matrices calculados para la muestra anterior y que irán acumulando los coeficientes de los sistemas de ecuaciones lineales obtenidos en las iteraciones previas para calcular los valores de los pesos. Estos viejos coeficientes se emplean para calcular los pesos en la iteración actual. De este modo, es posible manejar el conocimiento previamente adquirido y emplearlo para aproximar progresivamente el valor óptimo de los pesos. Este hecho permite que el algoritmo trabaje en modo *online* actualizando su conocimiento en función de la información recibida a lo largo del tiempo.

De este modo, empleando las salidas z_{ks} es posible aprender de manera independiente los pesos para ambas capas de la red al resolver sistemas de ecuaciones lineales independientes. En este punto, el algoritmo SBLLM en su versión original obtiene las sensibilidades de la función de coste con respecto a z_{ks} y las utiliza para calcular los incrementos que permiten modificar los valores de las salidas $\mathbf{z}$ de la capa oculta. Sin embargo, como ya se ha comentado previamente, en el caso de aprendizaje *online* debido al empleo de una única muestra en cada iteración del proceso, las sensibilidades puntuales toman valores muy pequeños dando lugar a incrementos mínimos que hacen que la actualización de las salidas de la capa oculta sea prácticamente inexistente. De este modo, en el aprendizaje *online* no resulta muy adecuado el empleo de las sensibilidades para la actualización de las salidas de la capa oculta y, en consecuencia, se hace necesario establecer otra aproximación que permita obtener los valores de estas salidas. El empleo de valores aleatorios para las salidas $\mathbf{z}$ de la capa oculta alcanza resultados satisfactorios. Sin embargo, estos valores aleatorios presentan una restricción, y es que deben obtenerse en base a una inicialización previa de los pesos de la red, tarea para la cual se puede emplear algún método conocido como por ejemplo Nguyen-Widrow [104]. Por tanto, en lugar de realizar una adaptación continuada de las salidas de la capa oculta a lo largo del proceso de aprendizaje, como hace el SBLLM, ahora estos valores se inicializan para cada muestra empleando unos pesos aleatorios.

Todo el desarrollo anterior se puede expresar en notación matricial. En este caso las ecuaciones 5.5 y 5.6 se escriben como,

$$\mathbf{A}_k^{(1)}(s)\mathbf{w}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s), \quad k = 1, \ldots, K, \tag{5.11}$$

$$\mathbf{A}_j^{(2)}(s)\mathbf{w}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s), \quad j = 1, \ldots, J, \tag{5.12}$$

donde s indica la iteración de aprendizaje actual que se corresponde con la muestra de entrenamiento $s - ésima$. También se reescriben en notación matricial las ecuaciones 5.7 y 5.8 para la subred 1,

$$\mathbf{A}_k^{(1)}(s) = \mathbf{A}_k^{(1)}(s-1) + h_k(s)\mathbf{x}(s)\mathbf{x}^T(s)g_k'^2(\bar{z}_k(s)), \tag{5.13}$$

$$\mathbf{b}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s-1) + h_k(s)g_k^{-1}(z_k(s))\mathbf{x}(s)g_k'^2(\bar{z}_k(s)). \tag{5.14}$$

De manera análoga, las ecuaciones 5.9 y 5.10 correspondientes a la segunda subred se escriben ahora como,

$$\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s)f_j'^2(\bar{d}_j(s)), \tag{5.15}$$

$$\mathbf{b}_k^{(2)}(s) = \mathbf{b}_k^{(2)}(s-1) + h_j(s)f_j^{-1}(d_j(s))\mathbf{z}(s)f_j'^2(\bar{d}_j(s)). \tag{5.16}$$

Finalmente,

$$\mathbf{w}_k^{(1)}(s) = \mathbf{A}_k^{(1)^{-1}}(s)\mathbf{b}_k^{(1)}(s), \tag{5.17}$$

$$\mathbf{w}_j^{(2)}(s) = \mathbf{A}_j^{(2)^{-1}}(s)\mathbf{b}_j^{(2)}(s). \tag{5.18}$$

La complejidad de este método viene dada por la complejidad para resolver cada uno de los sistemas de ecuaciones lineales, uno para cada neurona de salida de cada capa. Como ya se ha comentado en capítulos previos, existen diferentes métodos computacionalmente eficientes que se pueden emplear para resolver este tipo de problemas (excepto para matrices mal condicionadas), con una complejidad de $O(N^2)$ [23, 27] donde N el número de parámetros desconocidos. En el caso de matrices mal condicionadas, el problema puede resolverse aplicando la pseudoinversa Moore-Penrose [111]. Teniendo en cuenta estas consideraciones se puede decir que la complejidad del método de aprendizaje viene dada por $O(MN^2)$ siendo $M = K + J$.

Finalmente, el Algoritmo 5.1 detalla el método de aprendizaje propuesto empleando los conceptos definidos previamente.

5.2. Resultados experimentales

En esta sección se comprueba el rendimiento del algoritmo de aprendizaje *online* al trabajar en distintos entornos. El objetivo es verificar que el método propuesto, además de trabajar en modo *online*, es capaz de adaptar su respuesta a contextos no estacionarios gracias a la incorporación del término de olvido (h) en la función de coste. Si los resultados obtenidos son significativos, el método podría aplicarse en entornos que tienen que trabajar en tiempo real y que presentan un comportamiento evolutivo en el tiempo.

A continuación, se indican las principales motivaciones del estudio experimental que se presenta. En primer lugar, se desea comparar el rendimiento del método propuesto con dos configuraciones diferentes, con y sin capacidad de adaptación a los cambios. Para ello se emplea el mismo algoritmo pero considerando una función de olvido diferente según cada caso: monótona creciente, para la configuración con capacidad de adaptación; una función constante, en caso contrario. Esto nos permite comprobar si el factor de olvido mejora el rendimiento del algoritmo *online*, sin perjudicarlo cuando el entorno de trabajo es estacionario. Tras un estudio experimental del comportamiento del método propuesto con distintas funciones de olvido, se seleccionó una función exponencial. La función empleada se define como,

$$h_k(s) = h_j(s) = e^{\mu s}, k = 1, \dots K; j = 1, \dots, J, \tag{5.19}$$

siendo K y J las salidas de las subredes de la figura 5.1, s la $s-$ésima muestra, y μ un parámetro real positivo que controla el crecimiento de la función y, por tanto, la respuesta de la red a los cambios del entorno. La red no dispondrá de capacidad de adaptación a los cambios cuando el valor correspondiente del parámetro μ sea cero. En

Algoritmo 5.1 Algoritmo de aprendizaje para redes de neuronas de dos capas *online* e incremental con capacidad de adaptación a cambios de contexto.

Entradas: $\mathbf{x}(s) = (x_1(s), x_2(s), \ldots x_I(s)),\ s = 1, \ldots, S$

$\mathbf{d}(s) = (d_1(s), d_2(s), \ldots d_J(s))$

sesgos, $x_0(s) = 1, z_0(s) = 1$

1. Fase de inicialización:
2. $\mathbf{A}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times(I+1)}$, $\mathbf{b}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times 1}$, $\forall k = 1, \ldots, K$,
3. $\mathbf{A}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times(K+1)}$, $\mathbf{b}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times 1}$, $\forall j = 1, \ldots, J$,
4. Calcular los pesos iniciales $\mathbf{w}_k^{(1)}(0)$, con algún método de inicialización.
5. Para cada muestra s $(s = 1, \ldots, S)$,
6. $z_k(s) = g(\mathbf{w}_k^{(1)}(0), \mathbf{x}(s))$,
7. Para cada salida k de la subred 1 $(k = 1, \ldots, K)$,
8. $\mathbf{A}_k^{(1)}(s) = \mathbf{A}_k^{(1)}(s-1) + h_k(s)\mathbf{x}(s)\mathbf{x}^T(s)g_k'^2(\bar{z}_k(s))$ (ec. 5.13),
9. $\mathbf{b}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s-1) + h_k(s)g_k^{-1}(z_k(s))\mathbf{x}(s)g_k'^2(\bar{z}_k(s))$ (ec. 5.14),
10. Calcular $\mathbf{w}_k^{(1)}(s)$ resolviendo el sistema de ec. lineales (ec. 5.17),
11. fin del Para.
12. Para cada salida j de la subred 2 $(j = 1, \ldots, J)$,
13. $\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s)f_j'^2(\bar{d}_j(s))$ (ec. 5.15),
14. $\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s)f_j^{-1}(d_j(s))\mathbf{z}(s)f_j'^2(\bar{d}_j(s))$ (ec. 5.16),
15. Calcular $\mathbf{w}_j^{(2)}(s)$ resolviendo el sistema de ec. lineales (ec. 5.18),
16. fin del Para.
17. fin del Para.

otro caso, el valor establecido de μ permitirá que la red adapte su comportamiento al entorno de trabajo correspondiente.

Para realizar un estudio experimental completo, el algoritmo propuesto se compara con otros métodos de aprendizaje *online*. Tras un análisis de la bibliografía disponible en el campo se seleccionan, el *Online Sequential Extreme Learning Machine* (OS-ELM) [68, 90] y el *Online Support Vector Regression* (SVR Online) [92, 109]. La elección de ambos sistemas, OS-ELM y SVR Online, se debe a que son dos métodos importantes dentro del estado del arte del aprendizaje automático y que presentan la característica de ser *online*. El primero de ellos, OS-ELM, es un método rápido y eficiente, aplicable a redes de neuronas de dos capas con alimentación hacia delante que presenta la particularidad de realizar el aprendizaje en dos fases diferenciadas. La primera fase es la de inicialización, en ella se asignan valores aleatorios a los pesos de la primera capa para obtener las salidas iniciales de la red. Posteriormente, en la fase de aprendizaje secuencial se aproximan, de manera continuada, los valores de los pesos de la segunda de las capas. El SVR es un método típico para sistemas de aprendizaje diferentes a las redes de neuronas, en este caso se elige una versión que permite aprendizaje *online*. El método permite construir máquinas de vectores soporte para problemas de regresión, con la posibilidad de añadir o eliminar muestras sin necesidad de realizar el reentrenamiento desde el principio.

A continuación se presentan los resultados obtenidos en diferentes escenarios de trabajo, contextos estacionarios y no estacionarios, con el objeto de evaluar el comportamiento y las capacidades de los diferentes algoritmos.

5.2.1. Contextos estacionarios

El primer conjunto de datos empleado corresponde al ratio de cambio monetario de Dólares frente a Libras (*US-UK*) y se obtiene de la página web de datos de la Reserva Federal [1]. El histórico de datos del conjunto *US-UK* tiene frecuencia mensual

y corresponde al periodo comprendido entre los años 1971 y 2009, con un total de 456 observaciones. El objetivo de la red es predecir la muestra actual en base a ocho patrones previos, la salida deseada se puede observar en la figura 5.2.

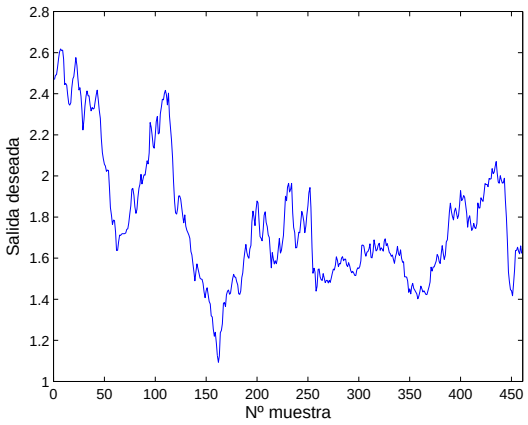


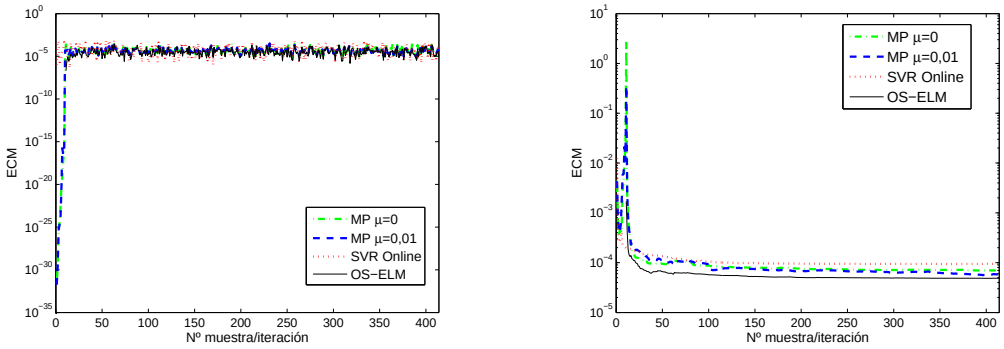
Figura 5.2: Contextos estacionarios. Señal deseada para el conjunto de datos US-UK.

Las condiciones experimentales se resumen brevemente a continuación. Con el objeto de obtener resultados estadísticamente significativos, se realizan 5 simulaciones y, para cada una de ellas, se calcula el rendimiento por medio de una validación cruzada de 10 paquetes, de manera que los resultados que se presentan son valores medios. Con respecto a los métodos de redes de neuronas (algoritmo propuesto y OS-ELM) se emplean 10 neuronas ocultas. Cabe mencionar que el objetivo no es investigar el número óptimo de neuronas ocultas para el conjunto de datos, sino mejorar el rendimiento del método para una topología dada. Para seleccionar el valor adecuado del factor μ de la función de olvido se realizó un estudio comparativo con diferentes valores, al igual que el que se presentó en el Capítulo 3 (sección 3.2.1). Finalmente, para este conjunto de datos se establece el valor de μ=0,01 para la configuración del método propuesto con capacidad de olvido. En cuanto al método SVR Online, los parámetros cuyos valores hay que determinar son 1) el límite para los multiplicadores de Lagrange (C) y, 2) el margen de tolerancia para los errores (ε). Además, en todos los experimentos de este capítulo se seleccionó como función kernel una gaussiana. Para contextos estacionarios

se estableció $C = 100$ y $\varepsilon = 0,001$. En todos los casos estos parámetros se seleccionaron mediante un procedimiento de prueba y error hasta encontrar los más favorables.

La figura 5.3 muestra el error cuadrático medio (ECM) obtenido en las fases de entrenamiento y test por cada uno de los cuatro métodos comparados: 1) el método propuesto sin capacidad de adaptación a los cambios (etiquetado como MP μ=0), 2) el mismo método incluyendo la capacidad de adaptación (MP μ=0,01), 3) el SVR Online, y 4) el OS-ELM. Como se puede observar en la figura 5.3(b) debido a la ausencia de cambios estadísticos significativos en el conjunto de datos, no se aprecian diferencias en el rendimiento de los métodos, concluyendo que el método propuesto con capacidad de adaptación a los cambios trabaja de manera adecuada también en este tipo de entornos, aunque en este caso no se obtenga ninguna ventaja en su aplicación. Por otro lado, en las gráficas de la fase de test se puede observar un pico al comienzo de la curva del error correspondiente al algoritmo propuesto. Este problema se debe a la relación entre el número de muestras de entrenamiento (S) y el número de parámetros libres en el sistema de ecuaciones lineales, el cual está relacionado con el número de unidades ocultas K de la red. Cuando el número de muestras disponible es menor que el número de unidades ocultas ($S < K$) el sistema de ecuaciones puede encontrar una solución de mínimo error asociando a cada parámetro libre un patrón de entrenamiento. A medida que el número de muestras de entrenamiento aumenta, los errores de entrenamiento y test decrecen hasta un instante en el que el número de muestras es mayor que el de parámetros libres. En este momento ($S > K$), no es posible realizar una correspondencia uno a uno entre las muestras y los parámetros libres del sistema. Ahora, el sistema debe encontrar una solución de compromiso que implica una ligera degradación del error debido a que el sistema tiene prácticamente la misma información (un único ejemplo a mayores), pero ya no es posible realizar una correspondencia uno a uno. Esta es la razón por la que aparece el pico en la curva de test siempre que el número de ejemplos y el de neuronas ocultas coincide. A partir de este momento, el error continúa descendiendo a medida que el sistema dispone de nuevas muestras con las que generalizar la solución.

Con respecto al tiempo de CPU necesario por cada método para completar el apren-



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 5.3: Curvas ECM para las fases de entrenamiento y test para el conjunto de datos US-UK.

dizaje, vale la pena indicar que cualquiera de los métodos para redes de neuronas aprenden más rápido que el SVR Online. Esta comparativa (en segundos) se presenta en la tabla 5.1.

Algoritmo	**Tiempo total de CPU (s)**
MP μ=0	1,458
MP μ=0,01	1,477
SVR Online	283,300
OS-ELM	0,397

Tabla 5.1: Tabla de tiempos de CPU (en segundos) requerido por cada algoritmo de aprendizaje en el conjunto de datos US-UK.

5.2.2. Contextos no estacionarios

En entornos no estacionarios el concepto reflejado en la salida deseada puede cambiar en el tiempo. Las modificaciones entre contextos pueden ser bruscas cuando la distribución cambia rápidamente, o graduales si existe una transición suave entre distribuciones [15, 147]. Para comprobar el rendimiento del método propuesto en diferentes

situaciones, se consideraron diferentes ejemplos para ambos tipos de entorno.

Los primeros conjuntos de datos que se presentan corresponden a series artificiales generadas mediante ecuaciones. En estos casos, como la señal evoluciona en el tiempo, se generan varios cambios de contexto. Como resultado se obtiene un conjunto de test diferente para cada uno de ellos, de manera que cada muestra de entrenamiento tiene asociado un conjunto de test, aquel que representa el contexto al que pertenece la muestra.

5.2.2.1. Cambios bruscos de contexto

En esta sección consideramos dos conjuntos de datos artificiales, ambos se generan para simular una distribución que presenta cambios repentinos en el tiempo.

Conjunto de datos artificial 1. El primer conjunto de datos está formado por 4 variables de entrada aleatorias que contienen valores de una distribución normal de media 0 y desviación típica 0,1. La salida deseada se obtiene mediante una mezcla lineal de funciones no lineales aplicadas sobre todas las variables de entrada. Las funciones no lineales empleadas son sigmoide tangente hiperbólica, exponencial, seno y sigmoide logarítmica. Para cada contexto o estado, se aplican diferentes combinaciones lineales de estas funciones. Para generar la salida deseada, y con el fin de provocar diferentes contextos, los coeficientes utilizados para la combinación de funciones se modifican cada 150 muestras, empleando cada vez una fila diferente de la siguiente matriz de mezcla.

$$M1 = \begin{pmatrix} 0{,}1 & 0{,}2 & 0{,}5 & 0{,}8 \\ 0{,}2 & 0{,}3 & 0{,}5 & 0{,}8 \\ 0{,}3 & 0{,}4 & 0{,}6 & 0{,}7 \\ 0{,}4 & 1{,}2 & -0{,}1 & 0{,}2 \end{pmatrix}.$$

De este modo, se obtiene un conjunto de entrenamiento de 600 muestras y 4 con-

juntos de test, uno para cada uno de los cambios de la mezcla lineal del conjunto de entrenamiento. La figura 5.4 contiene la señal empleada como salida deseada durante el proceso de entrenamiento, las líneas verticales que se pueden observar indican el momento en el aparece un cambio de contexto. En este ejemplo, las redes neuronales empleadas contienen 15 neuronas ocultas. Además, en el método propuesto se establece a 0,05 el valor del factor de capacidad de olvido. Por último el SVR Online emplea los valores de $\varepsilon = 0,1$ y $C = 100$.

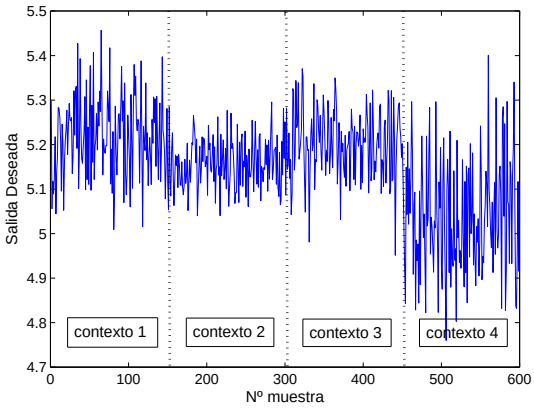
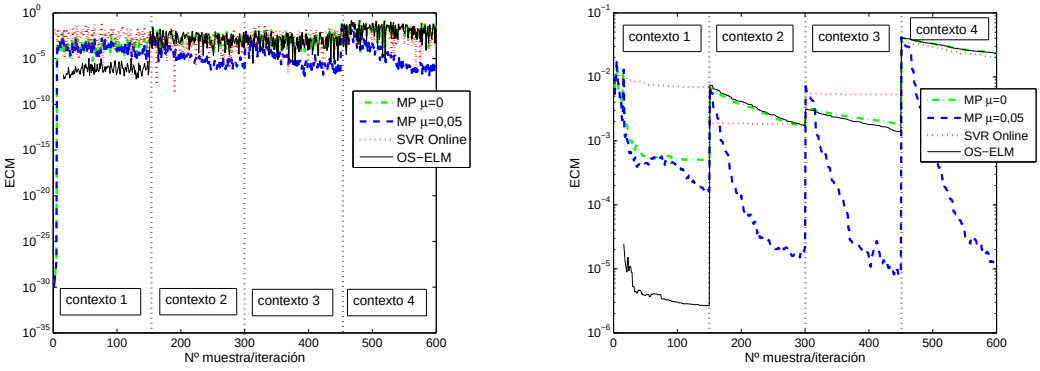


Figura 5.4: Contextos no estacionarios con cambios bruscos de contexto. Conjunto de datos artificial 1: Salida deseada para conjunto de entrenamiento.

La figura 5.5 muestra el error obtenido por cada uno de los métodos durante las fases de entrenamiento y test. En las curvas de test que se presentan, cada punto de la señal representa el valor medio obtenido en el conjunto de test correspondiente a la muestra de entrenamiento actual. En ambas figuras, y en mayor proporción en la fase de test, se pueden observar ciertos picos que se deben a la existencia de cambios en la señal de entrada y corresponden a diferentes contextos. Como se puede ver en la figura 5.5(b), el error cometido por el método propuesto con μ=0,05 aumenta (nunca más que el de los otros métodos), pero enseguida decrece hasta el momento en que la red es capaz de adaptarse al nuevo contexto. Tanto las dos configuraciones del método propuesto como el OS-ELM disminuyen el error cometido tras un cambio del sistema, sin embargo, es el método propuesto con capacidad de adaptación el que consigue un descenso más

pronunciado, siendo la diferencia estadísticamente significativa. La configuración del método sin capacidad de adaptación (μ=0) y el OS-ELM disminuyen el error pero no consiguen alcanzar su rendimiento previo. En cuanto al SVR Online no es capaz de adaptarse al nuevo contexto.



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 5.5: Contextos no estacionarios con cambios bruscos de contexto. Conjunto de datos artificial 1: Curvas ECM para las fases de entrenamiento y test.

Conjunto de datos artificial 2. El segundo conjunto de datos se genera de la misma manera que el primero, pero empleando una combinación lineal diferente de las funciones no lineales antes mencionadas por medio de la siguiente nueva matriz de mezcla.

$$M2 = \begin{pmatrix} 2{,}1 & 1{,}3 & -1{,}1 & 1{,}3 \\ -1{,}1 & -0{,}3 & 0{,}7 & -1{,}1 \\ -0{,}1 & 0{,}2 & 1{,}8 & -2{,}3 \\ -3{,}1 & -1{,}5 & 0{,}4 & 1{,}8 \\ 1{,}5 & -0{,}9 & 1{,}2 & 0{,}6 \end{pmatrix}.$$

De nuevo, la señal de salida se modifica cada 150 muestras empleando cada vez una fila diferente de la matriz de mezcla, de modo que se obtiene un conjunto de entrenamiento de 750 muestras y 5 conjuntos de test, uno para cada uno de los cambios

de la mezcla lineal sobre el conjunto de entrenamiento. La figura 5.6 contiene la señal empleada como salida deseada durante el proceso de entrenamiento. Con respecto a las condiciones experimentales, las redes neuronales emplean 20 neuronas ocultas, y en el método propuesto se establece a 0,05 el valor del factor de olvido. El SVR Online utiliza los valores de $\varepsilon = 0,1$ y $C = 10$.

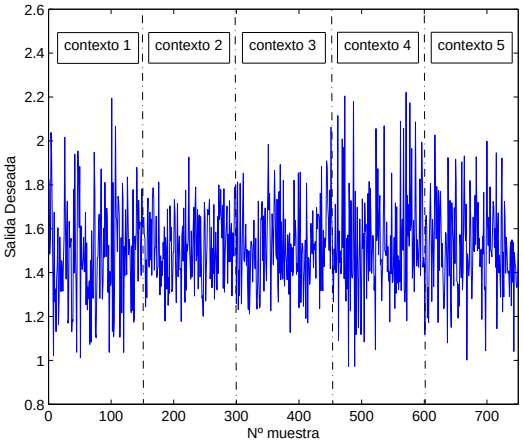
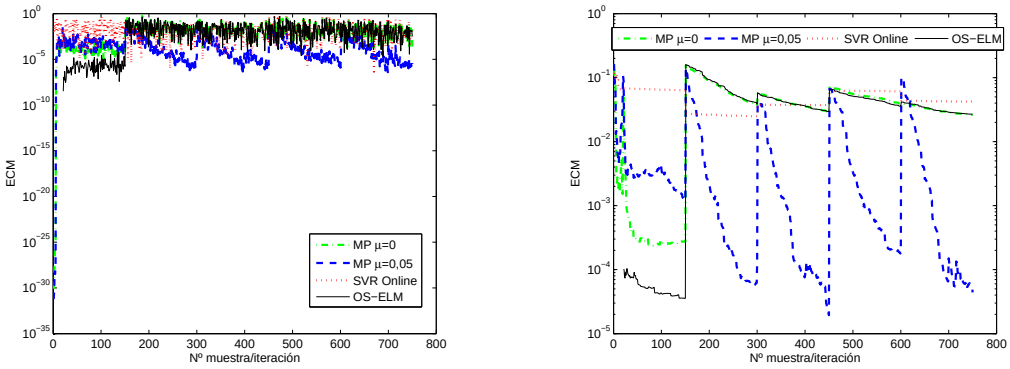


Figura 5.6: Contextos no estacionarios con cambios bruscos de contexto. Conjunto de datos artificial 2: Salida deseada para conjunto de entrenamiento.

La figura 5.7 muestra el error cuadrático medio obtenido por cada uno de los métodos en las fases de entrenamiento y test. Al igual que en el ejemplo previo, la configuración con capacidad de adaptación del método propuesto es capaz de reponerse cuando aparece un cambio, el OS-ELM y el método propuesto sin adaptación a cambios disminuyen el error cometido tras un cambio pero en una proporción mucho menor, mientras que el SVR incrementa sus error debido a la naturaleza de la distribución.

Los tiempos medios de ejecución que requiere cada uno de los métodos para completar el aprendizaje para los conjuntos de datos con cambios de contextos bruscos se presentan en la tabla 5.2. Como se puede observar tanto el método propuesto como el OS-ELM completan el proceso de aprendizaje en un tiempo reducido, mientras que el SVR Online es un método lento que presenta un consumo elevado de recursos



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 5.7: Contextos no estacionarios con cambios bruscos de contexto. Conjunto de datos artificial 2: Curvas ECM para las fases de entrenamiento y test.

temporales.

Algoritmo	**Tiempo CPU Conjunto 1**	**Tiempo CPU Conjunto 2**
MP μ=0	2,206	3,659
MP μ=0,05	2,214	3,619
SVR Online	304	513
OS-ELM	0,702	1,351

Tabla 5.2: Tabla de valores medios del tiempo total de CPU (en segundos) requerido por cada algoritmo de aprendizaje en contextos no estacionarios para conjuntos con cambios bruscos de contexto.

5.2.2.2. Cambios de tendencia o cambios graduales de contexto

En esta sección consideramos, en primer lugar, dos series temporales artificiales. En este caso, las distribuciones correspondientes presentan cambios continuos en cada una de las muestras como consecuencia de la evolución de los coeficientes de la matriz de mezcla empleada para la generación del conjunto de datos. En segundo lugar, el estudio se realiza para una aplicación del mundo real, la predicción del comportamiento de un

rodamiento en un proceso industrial.

Conjunto de datos artificial 3. Se genera un nuevo conjunto de datos artificial que, en esta ocasión, presenta una evolución *gradual* y continua en cada muestra del conjunto de entrenamiento. El conjunto está formando por 4 variables de entrada generadas por medio de una distribución normal con media cero y desviación típica 0,1. La salida se obtiene por medio de una mezcla no lineal sobre las mismas fijando los coeficientes iniciales del vector de mezcla como,

$$a(0) = \left[\begin{array}{cccc} 0{,}5 & 0{,}2 & 0{,}7 & 0{,}8 \end{array} \right],$$

y evolucionando estos coeficientes en el tiempo de acuerdo a la siguiente ecuación,

$$a_j(s) = a_j(s-1) + 2 \times 10^{-4} \sqrt{exp(a_j(s-1))}, s = 1, ..., S,$$

donde el subíndice j indica la componente del vector de mezcla. El conjunto dispone de un total de 500 muestras de entrenamiento y un conjunto diferente de test (de 150 muestras), para cada una de ellas debido a la evolución continua de la señal que origina un contexto diferente para cada una de las muestras de entrenamiento. Después de este proceso se obtiene un conjunto de datos cuya salida deseada durante el proceso de entrenamiento se presenta en la figura 5.8. Para este conjunto de datos, las redes de neuronas emplean 30 unidades en su capa oculta, y el valor de μ para la configuración con capacidad de adaptación se establece a 0,05. A su vez los valores de los parámetros asociados al SVR Online se seleccionaron como $\varepsilon = 0,001$ y C=100.

Las curvas de error para las fases de entrenamiento y test se presentan en la figura 5.9. Como se puede observar en los resultados de test, tanto el método propuesto sin capacidad de adaptación como el SVR Online y el OS-ELM sufren un deterioro de su rendimiento a lo largo del tiempo. Sin embargo el método propuesto con capacidad de adaptación obtiene un buen error a pesar de la naturaleza de la distribución de la señal.

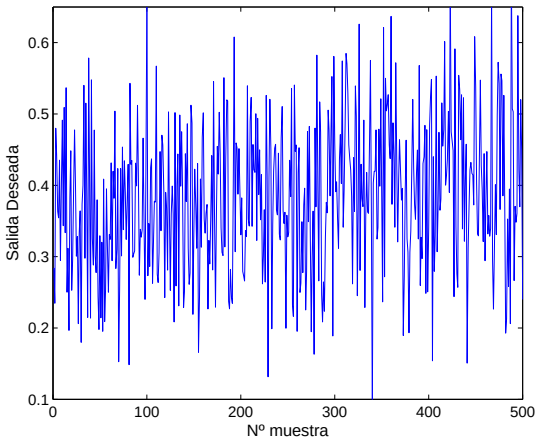
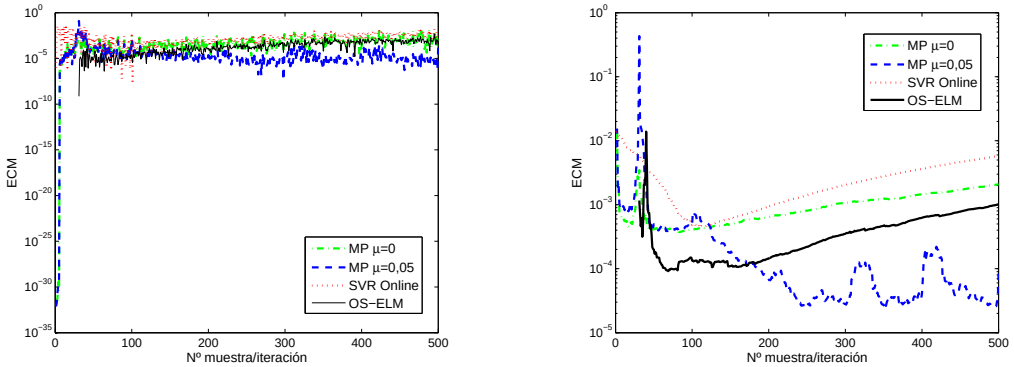


Figura 5.8: Contextos no estacionarios con cambios graduales de contexto. Conjunto de datos artificial 3: Salida deseada para el conjunto de entrenamiento.



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 5.9: Contextos no estacionarios con cambios graduales de contexto. Conjunto de datos artificial 3: Curvas ECM para las fases de entrenamiento y test.

Conjunto de datos artificial 4. En este caso se genera otro conjunto de datos artificial que también presenta una evolución en cada muestra de entrenamiento. La manera de generar la serie es la misma que en el ejemplo previo, pero ahora la ecuación que se emplea para la evolución de la distribución partiendo de los coeficientes que se establecen como iniciales es la siguiente,

$$a_j(s) = a_j(s-1) + \frac{s}{10^{4,7}} logsig(a_j(s-1)), s = 1, \ldots, S,$$

donde j es el índice que implica el componente del vector. El conjunto dispone de un total de 500 muestras de entrenamiento y un conjunto diferente de test de 150 muestras para cada una de ellas, debido a la evolución continua de la señal que origina un contexto diferente para cada una de las muestras de entrenamiento. Finalmente se obtiene un conjunto de datos cuya salida deseada durante el proceso de entrenamiento se presenta en la figura 5.10. Las redes de neuronas emplean 15 unidades en su capa oculta, el valor de μ para la capacidad de adaptación se fija a 0,1 y los parámetros asociados al SVR Online toman los valores $\varepsilon = 0,001$ y C=100.

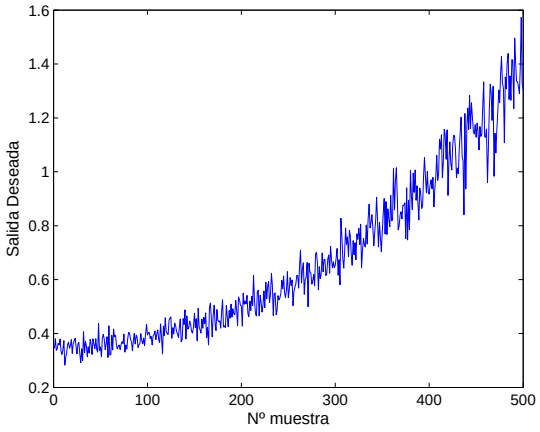
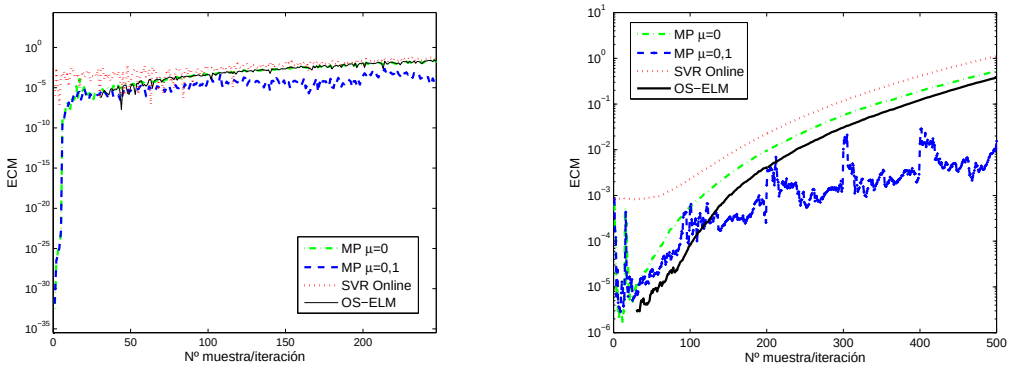


Figura 5.10: Contextos no estacionarios con cambios graduales de contexto. Conjunto de datos artificial 4: Salida deseada para el conjunto de entrenamiento.

En la figura 5.11 se muestran las curvas de error. Como se puede observar en la gráfica de test, de nuevo, es el método propuesto el que obtiene el mejor error incluso en este caso en el que se presentan cambios continuos.

Finalmente, con respecto a las demandas temporales de los algoritmos en estudio en la tabla 5.3 se puede observar el tiempo medio que requiere cada método para los ejemplos con cambios graduales de contexto. Como se puede comprobar al igual que en los casos anteriores, el SVR Online necesita mucho más tiempo que los otros métodos y obtiene peores resultados en todos los casos.



(a) Fase de entrenamiento.

(b) Fase de test.

Figura 5.11: Contextos no estacionarios con cambios graduales de contexto. Conjunto de datos artificial 4: Curvas ECM para las fases de entrenamiento y test.

Algoritmo	**Tiempo CPU Conjunto 3**	**Tiempo CPU Conjunto 4**
MP μ=0	1,800	1,851
MP μ	1,834	1,884
SVR Online	216	179
OS-ELM	1,257	0,549

Tabla 5.3: Tabla de valores medios del tiempo total de CPU (en segundos) requerido por cada algoritmo de aprendizaje en contextos no estacionarios para conjuntos con cambios graduales de contexto.

Conjunto de datos 5: Aplicación a la predicción de fallos en sistemas mecánicos. En último lugar se plantea una prueba en un escenario real, una aplicación en un campo de interés como es la predicción de fallos basada en sistemas inteligentes. En concreto se trata de analizar datos reales de vibraciones de sistemas mecánicos de tipo rotativo con el objeto de conocer las capacidades de predicción de los sistemas inteligentes desarrollados. Para este conjunto de datos y en vista de los resultados experimentales anteriormente presentados, se prescinde del algoritmo SVR Online debido a dos motivos. En primer lugar, no alcanza un rendimiento adecuado según los resultados (en todos los casos) anteriormente presentados, y por otro lado requiere una elevada demanda computacional.

Con el objeto de comprobar la idoneidad de los algoritmos en estudio para la elaboración de un modelo de pronóstico de fallos en componentes mecánicos, se emplea el conjunto de datos de vibraciones facilitado por el Centro de Mantenimiento de Sistemas Inteligentes (*Center for Intelligent Maintenance Systems*, *IMS*) de la Universidad de Cincinnati [87]. Para obtener los datos, se instalaron 4 rodamientos en una plataforma colocando en cada uno de ellos 2 acelerómetros que permiten adquirir las vibraciones producidas en diferentes direcciones. La figura 5.12 muestra la plataforma de prueba donde se encuentran los rodamientos e ilustra también la colocación de los sensores. Los rodamientos se encuentran colocados sobre un eje que está conectado a un motor que dirige su movimiento. La velocidad de rotación se mantuvo constante a 2.000 rpm y se aplicó una carga radial de 6.000 lb sobre el eje y los rodamientos. Se obliga a todos los rodamientos a lubricar y se emplea un sistema de circulación del aceite que regula el flujo y la temperatura del lubricante. En los cables de retroalimentación del aceite se instala un mecanismo que permite recoger los posibles restos de aceite debidos al deterioro de los rodamiento. La prueba finaliza cuando el aceite acumulado adherido al mecanismo excede un cierto nivel causando por tanto un fallo eléctrico.

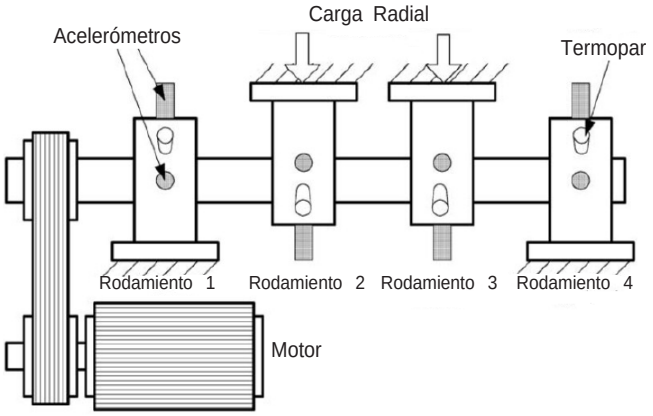


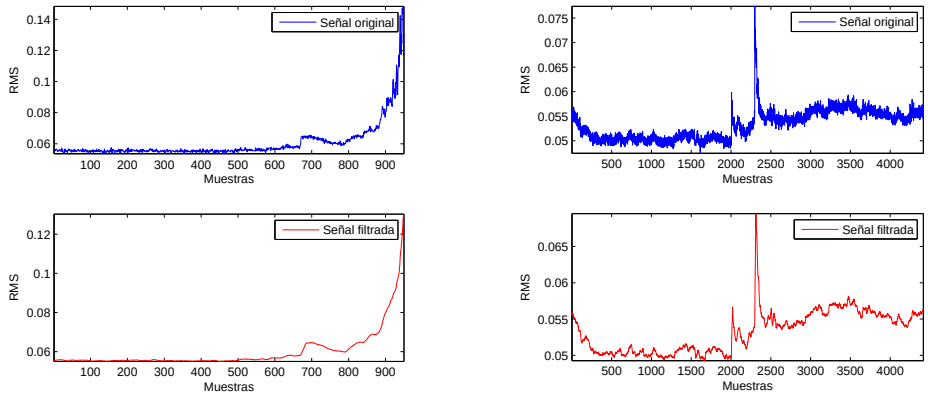
Figura 5.12: Plataforma de prueba de los rodamientos y colocación de los sensores para la obtención del conjunto de datos empleado en la experimentación.

Para este estudio experimental se emplearon dos de los conjuntos de datos que se proporcionan en el estudio y que contienen medidas de las vibraciones correspondientes a los 8 acelerómetros empleados:

- El primer conjunto contiene los datos recogidos cada 10 minutos durante 7 días. Al final del experimento aparece un fallo en el anillo externo del rodamiento 1.

- Un segundo conjunto que contiene los datos de vibraciones recogidos a lo largo de 30 días. En este caso, es el rodamiento 3 el que falla al finalizar la fase de prueba.

En monitorización de vibraciones, la práctica común es extraer de la señal de aceleración, la raíz cuadrada de los valores medios de la vibración al cuadrado (RMS), y emplear esta medida como parámetro global en el asesoramiento del estado de un componente. Como el funcionamiento normal del sistema de predicción es en tiempo real, a medida que se van obteniendo los ejemplos se aplica el algoritmo correspondiente sobre la señal RMS. En este estudio se realizó también un filtrado previo de la señal de RMS con el objetivo de observar la tolerancia al ruido en la señal. En las figuras 5.13(a) y 5.13(b) se presentan las señales (original y filtrada) correspondientes a los dos conjuntos de datos en estudio.



(a) Fallo en el rodamiento 1.

(b) Fallo en el rodamiento 3.

Figura 5.13: Aplicación a la predicción de fallos en sistemas mecánicos. Raíz cuadrada de los valores medios de la vibración al cuadrado (RMS) para los dos conjuntos de datos empleados.

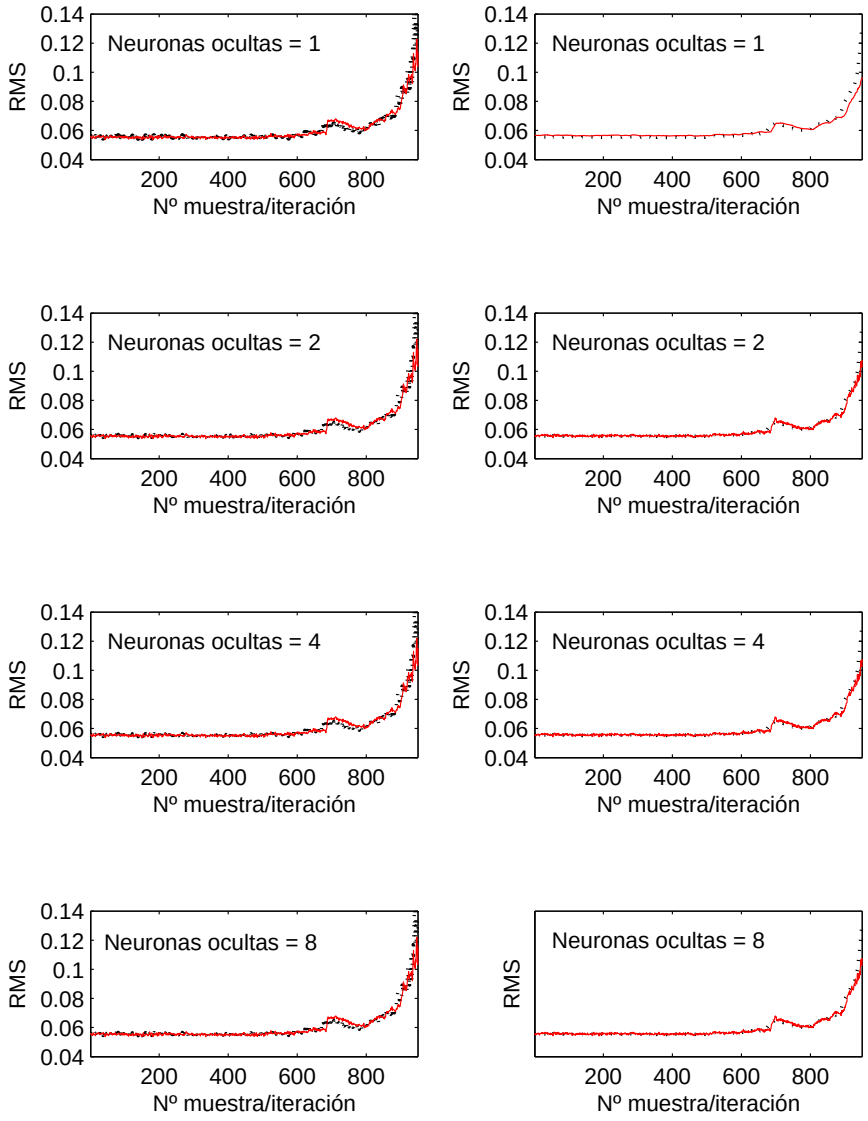
El objetivo que se persigue en este caso es predecir la muestra $t + 15$ empleando únicamente las muestras t y $t-1$. Se exploraron dos situaciones diferentes, (a) señal RMS

sin filtrar con el objeto de reproducir el escenario de monitorización de fallos original, y (b) señal RMS previamente filtrada. Para realizar una comparación exhaustiva se realizan diferentes pruebas modificando el número de unidades de la capa oculta de la red.

Las figuras 5.14, 5.15 y 5.16 muestran, respectivamente, los resultados obtenidos en el primer conjunto (fallo en rodamiento 1) después de aplicar las configuraciones del algoritmo propuesto sin y con capacidad de adaptación (valor de μ=0,01) y el OS-ELM. La columna de la izquierda corresponde a los resultaoda para la señal RMS sin filtrar mientras que, la de la derecha presenta las curvas para la señal RMS previamente filtrada. Como se puede observar el comportamiento del algoritmo propuesto es muy estable, obteniendo errores muy bajos para ambas señales, RMS original y filtrada, e independientemente del número de unidades ocultas empleadas. La configuración del método propuesto sin capacidad de adaptación a los cambios obtiene buenos resultados, pero la configuración que incorpora esta capacidad es capaz de alcanzar un ajuste mayor especialmente en la señal sin filtrar. Con respecto al algoritmo OS-ELM, se puede observar un comportamiento adecuado tanto para la señal RMS original como para la filtrada, únicamente cuando el número de neuronas es similar al número de entradas de la señal RMS (en este caso 2). Sin embargo, el comportamiento del método es muy inestable cuando el número de neuronas ocultas difiere del número de entradas, especialmente en el caso de la señal RMS original. Este comportamiento es debido en parte a la fase previa de inicialización que incluye el algoritmo. El método emplea al menos tantas muestras como neuronas ocultas para inicializar, de manera aleatoria, los pesos de la primera capa de la red. A partir de este momento, el proceso de aprendizaje persigue optimizar los valores de los pesos de la segunda capa. Esta aleatoriedad en la inicialización produce que el método presente un comportamiento inestable en algunas ocasiones que, como se puede observar en las figuras presentadas, parece disminuir en el caso de que el número de entradas y neuronas ocultas coincida.

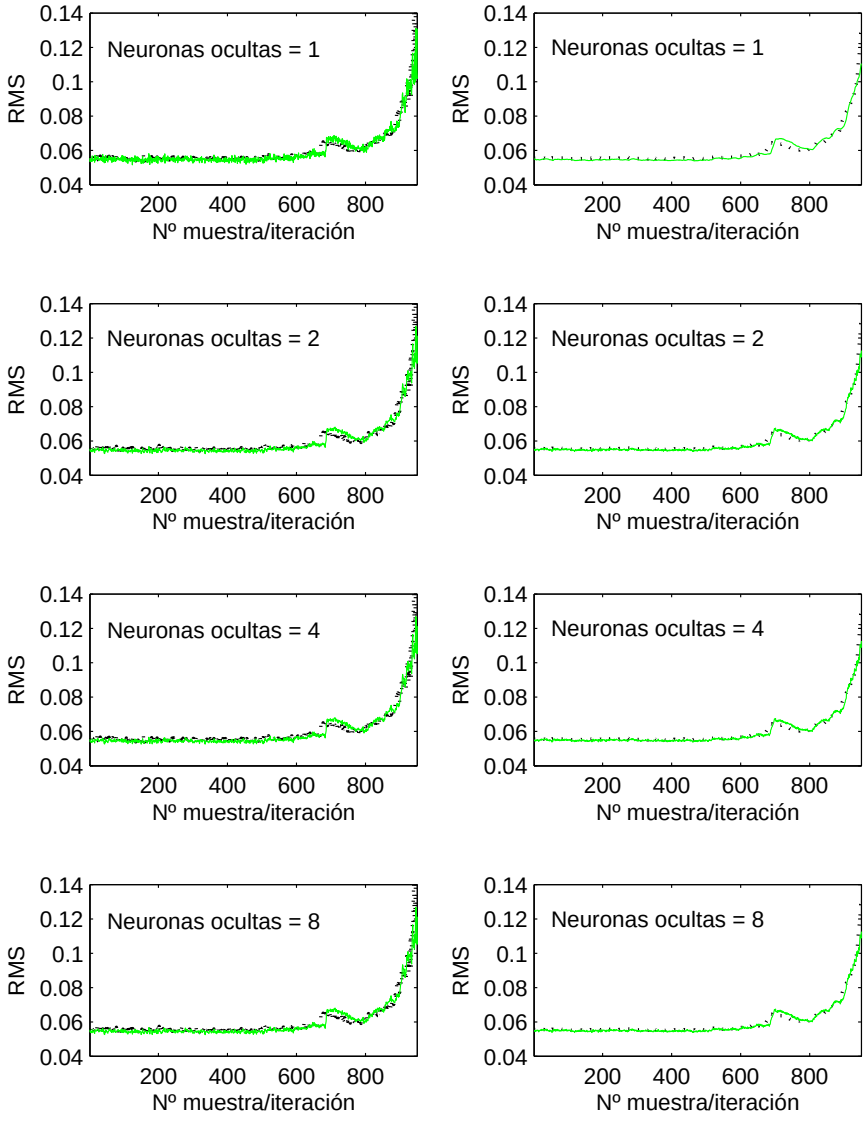
Los resultados obtenidos para el segundo conjunto de datos, en el que se produce un fallo en el anillo externo del rodamiento 3, se presentan en las figuras 5.17, 5.18 y 5.19 para cada uno de los métodos en estudio. Como se puede observar los resultados

obtenidos siguen la misma línea de comportamiento que en el conjunto previo pero se aprecia una excepción. En este caso, cuando la entrada es la señal filtrada el OS-ELM también se comporta de manera adecuada cuando emplea 4 neuronas ocultas. En el resto de los casos al igual que con el conjunto anterior su comportamiento es muy inestable, especialmente en el caso de manejar la señal original con sus posibles distorsiones.



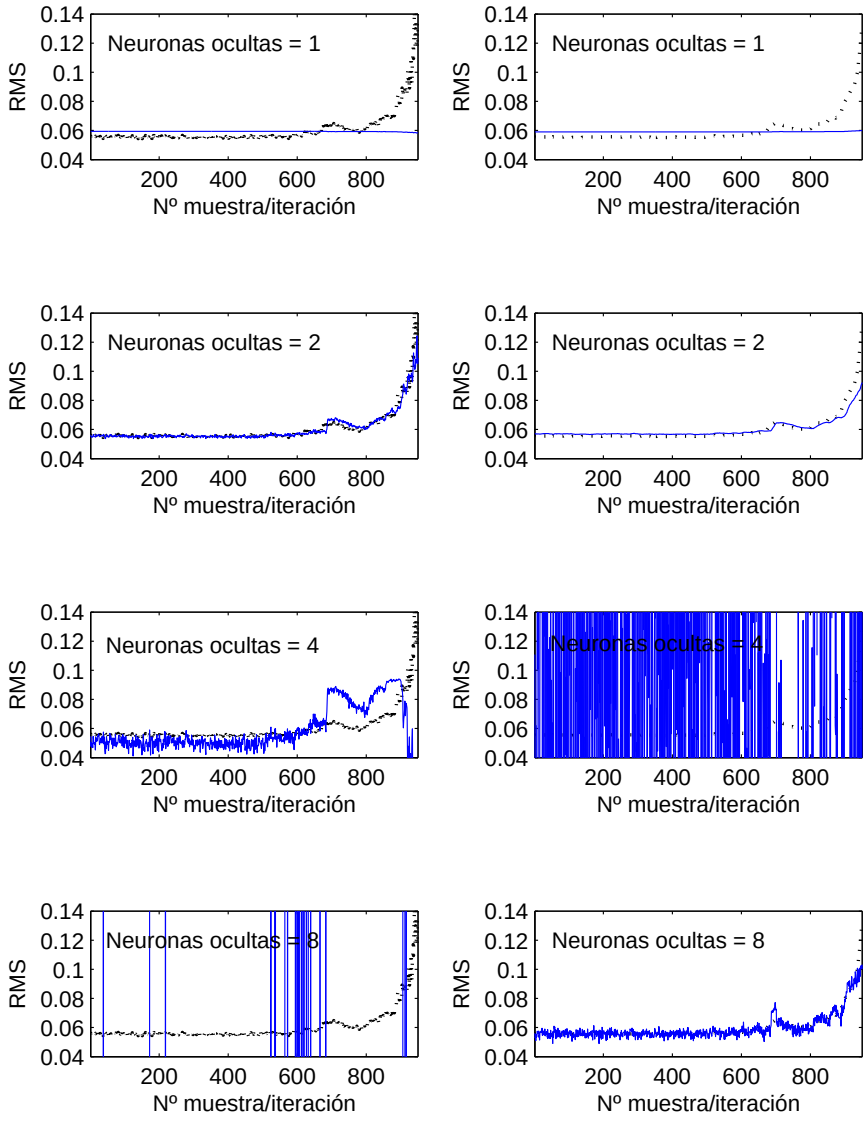
(a) Señal original (b) Señal filtrada

Figura 5.14: Resultados obtenidos empleando el método propuesto (μ=0) como predictor para el análisis de vibraciones en el rodamiento 1. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en rojo.



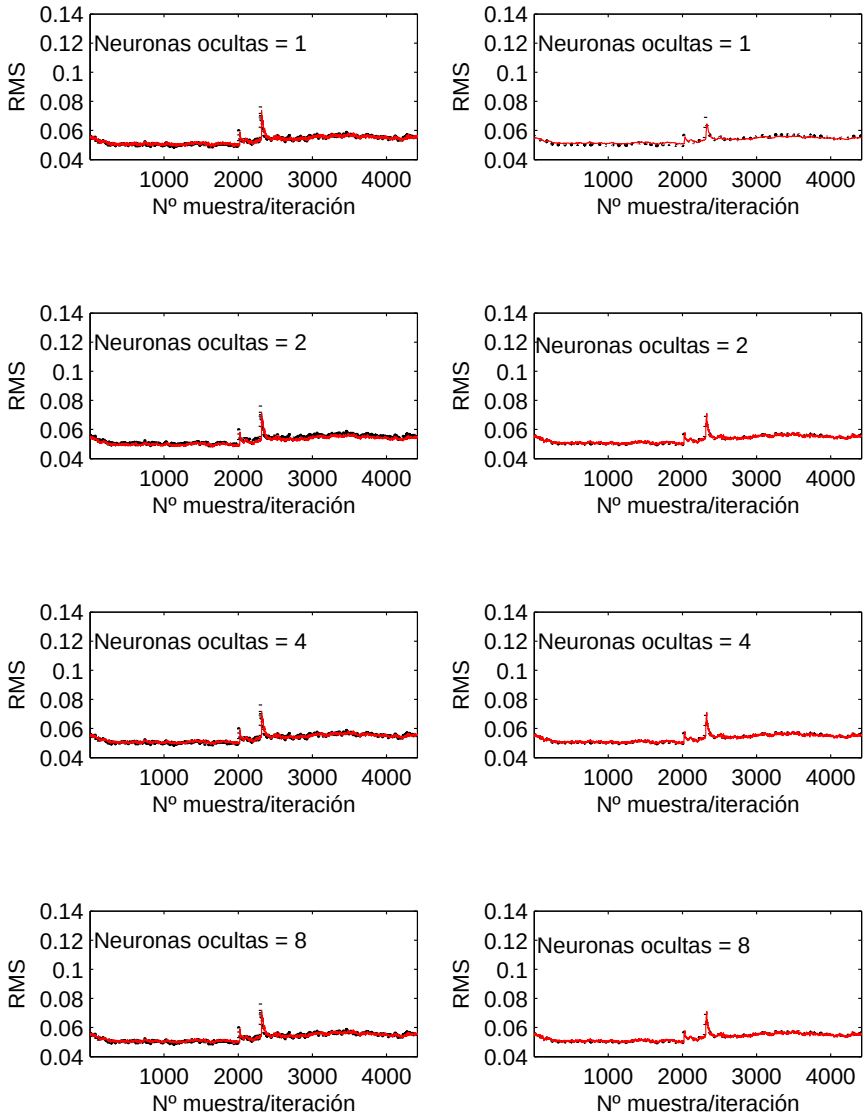
(a) Señal original (b) Señal filtrada

Figura 5.15: Resultados obtenidos empleando el método propuesto (μ=0,01) como predictor para el análisis de vibraciones en el rodamiento 1. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en verde.



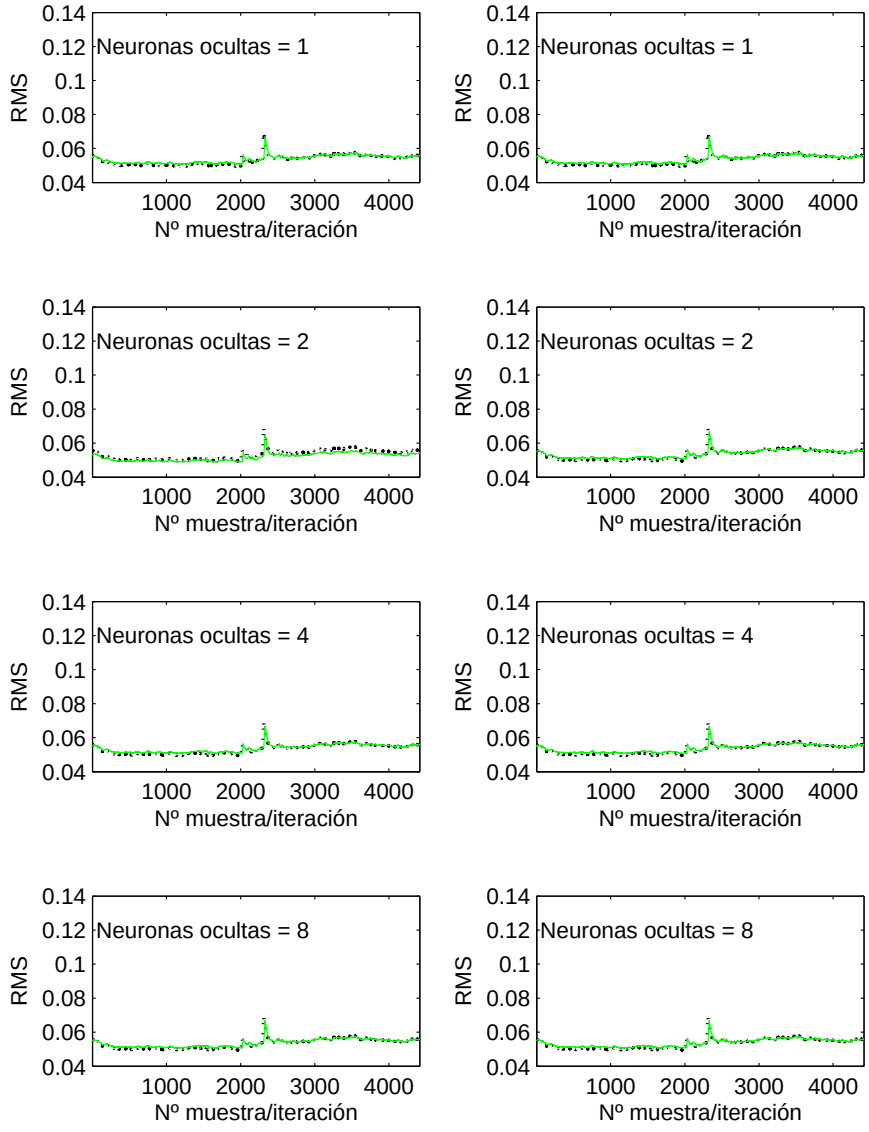
(a) Señal original (b) Señal filtrada

Figura 5.16: Resultados obtenidos empleando el OS-ELM como predictor para el análisis de vibraciones en el rodamiento 1. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en azul.



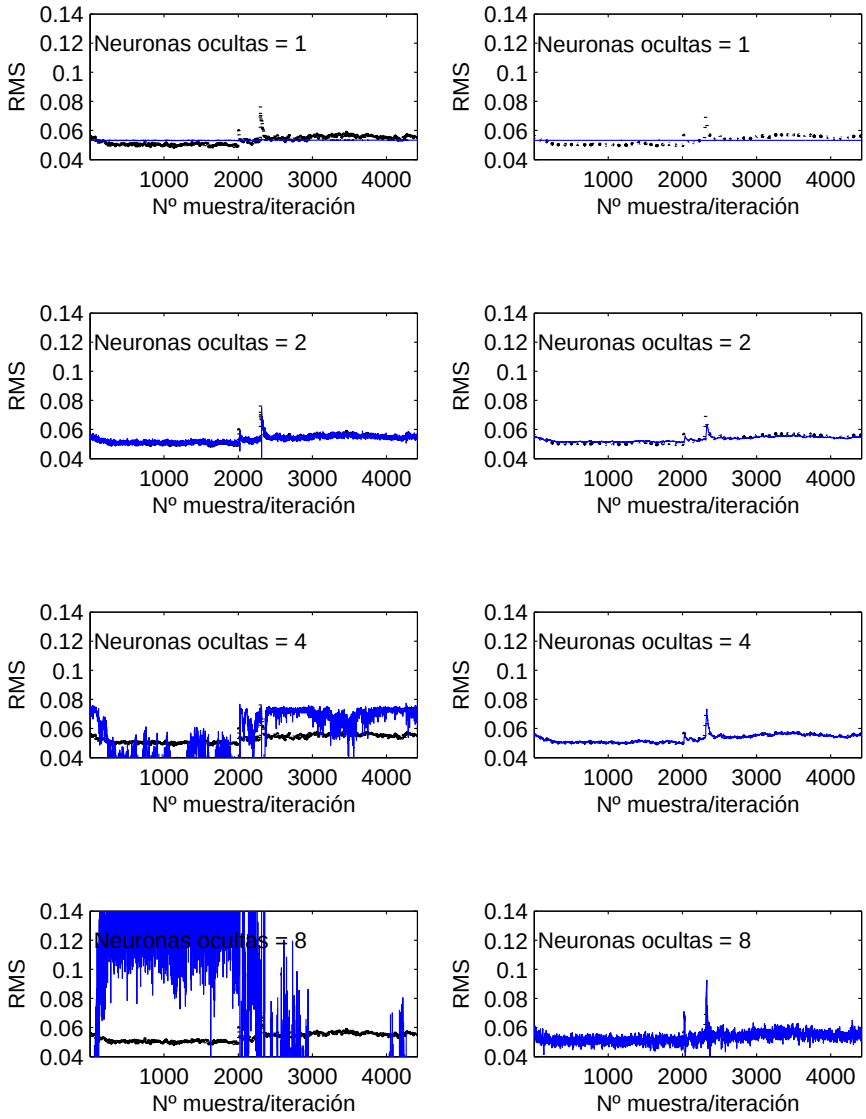
(a) Señal original (b) Señal filtrada

Figura 5.17: Resultados obtenidos empleando el método propuesto (μ=0) como predictor para el análisis de vibraciones en el rodamiento 3. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en rojo.



(a) Señal original (b) Señal filtrada

Figura 5.18: Resultados obtenidos empleando el método propuesto (μ=0,01) como predictor para el análisis de vibraciones en el rodamiento 3. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en verde.



(a) Señal original (b) Señal filtrada

Figura 5.19: Resultados obtenidos empleando el OS-ELM como predictor para el análisis de vibraciones en el rodamiento 3. En negro se presenta la señal RMS (original o filtrada) mientras que, la predicción obtenida aparece en azul.

5.3. Discusión

A la vista de los resultados presentados en la sección previa, se deduce que el algoritmo de aprendizaje propuesto es un método adecuado para trabajar con problemas de identificación de sistemas, tanto estacionarios como no estacionarios que presentan distintos tipos de evolución a lo largo del tiempo. La incorporación de un término de olvido en la función de coste mejora el comportamiento del método en aquellas situaciones en las que el entorno de trabajo evoluciona en el tiempo sin que se produzca un deterioro del buen rendimiento alcanzado al trabajar en contextos estacionarios. Esto quiere decir que el método propuesto trabaja de manera adecuada en entornos no estacionarios, debido a la adaptación dinámica de sus habilidades de olvido.

Con respecto a su comparación con los otros dos algoritmos de aprendizaje presentes en el estudio experimental, SVR Online y OS-ELM, se pueden extraer las siguientes conclusiones:

1. *Entornos estacionarios:* El método propuesto trabaja adecuadamente y obtiene resultados similares a los alcanzados por cualquiera de los otros dos métodos. En este tipo de contextos no se aprecia ventaja alguna al emplear el método adaptativo debido a la ausencia de cambios estadísticos importantes en los conjuntos de datos.

2. *Escenarios no estacionarios:* El método adaptativo propuesto mejora el rendimiento alcanzado por los otros métodos en todos los conjuntos de datos. El SVR Online no consigue adaptar su comportamiento a los cambios en la señal de entrada. A su vez, el OS-ELM y la configuración del método propuesto sin capacidad de adaptación presentan comportamientos y resultados semejantes entre sí.

3. *Tiempo de ejecución:* Los requisitos del SVR Online para llevar a cabo el entrenamiento son considerablemente mayores con respecto a las necesidades del OS-ELM y del algoritmo propuesto. El método propuesto presenta unos resultados competitivos, y similares en orden de magnitud, a los del OS-ELM.

Para finalizar se puede concluir que, en este capítulo se ha presentado un nuevo algoritmo de aprendizaje *online* e incremental con capacidad de adaptación a los cambios para redes de neuronas de dos capas con alimentación hacia delante, con el objetivo de que el método propuesto sea capaz de trabajar en entornos que evolucionan en el tiempo. Las principales características del método que se acaba de presentar son,

- Adapta sus capacidades de olvido de manera dinámica. La inclusión del control de cambio mejora el rendimiento del algoritmo al trabajar en entornos que evolucionan en el tiempo, sin perjudicar el rendimiento del método en entornos estacionarios.
- Es capaz de aprender adecuadamente la información reciente que se le proporciona, al mismo tiempo que retiene el conocimiento previo relevante.
- Presenta requisitos de memoria reducidos ya que únicamente necesita almacenar y actualizar, para cada neurona de salida de cada una de las subredes, la matriz A y el vector b. El tamaño de cada matriz A es $(I+1)\times(I+1)$ y de los vectores b es $(I+1)$, siendo I el número de entradas de la red. Además, la complejidad para resolver cada uno de los sistemas de ecuaciones lineales (uno para cada salida de cada subred) es $O(N^2)$, siendo N el número de parámetros asociados. Por tanto, la complejidad del método viene dada por $O(MN^2)$, donde M es $K+J$.
- Comparado con el SVR Online, el método propuesto obtiene un rendimiento superior en todos los casos y al mismo tiempo su demanda de recursos temporales es significativamente inferior.
- Comparado con el algoritmo OS-ELM, presenta un comportamiento más estable demostrando un buen comportamiento de manera generalizada.

Capítulo 6

Método de aprendizaje *online* con topología adaptativa para redes multicapa

En el capítulo previo se ha presentado un algoritmo de aprendizaje *online* para redes de neuronas de dos capas con alimentación hacia delante. El método incorpora un término de ponderación de los errores en la función de coste que permite disponer de un mecanismo de olvido de la información pasada. Esto permite alcanzar un buen rendimiento en situaciones en las que el entorno de trabajo evoluciona a lo largo del tiempo. A mayores, el método mantiene un comportamiento adecuado cuando trabaja en contextos estacionarios.

De manera general se asume que durante la fase de aprendizaje, las redes se adaptan mediante la actualización tanto de sus parámetros como de sus estructuras. Por tanto, además de que la red disponga de la capacidad de adaptar su comportamiento cuando se producen cambios en el entorno también sería beneficioso que su topología se adaptase en función de las necesidades del proceso de aprendizaje. La arquitectura de una red

neuronal debe estar en armonía con la cantidad y complejidad de los datos analizados. Si la red es demasiado pequeña, ésta no es capaz de aprender de manera adecuada sin embargo, una red demasiado grande tenderá a sobreajustar los datos [82]. Existen varias razones que justifican la búsqueda de una estructura óptima para una red de neuronas entre ellas, mejorar y acelerar la predicción, obtener una mejor generalización y reducir las necesidades computacionales, especialmente cuando se trabaja con grandes conjuntos de datos.

Teniendo en cuenta las consideraciones que se acaban de mencionar, en la siguiente sección se presenta el estudio que permite comprobar y justificar la viabilidad del algoritmo de aprendizaje *online* desarrollado para trabajar con estructuras de red incrementales. Una vez que se demuestre el funcionamiento del algoritmo al añadir nuevas neuronas ocultas durante el aprendizaje, se incorporará un mecanismo para decidir la adaptación de la topología de la red de forma automática. En otras palabras, al intentar el desarrollo de una topología incremental se deben resolver dos problemas fundamentales. En primer lugar hay que verificar que el algoritmo propuesto puede adaptar la topología para incorporar nuevas neuronas ocultas manteniendo, en la medida de lo posible, el conocimiento adquirido en las etapas previas del aprendizaje. En segundo lugar es preciso saber cuándo es el momento adecuado para añadir una nueva unidad a la red neuronal.

Es importante mencionar que la incorporación de esta capacidad de modificar la topología de la red permitirá obtener finalmente, un algoritmo de aprendizaje incremental no sólo con respecto a su capacidad de aprendizaje sino también, respecto a su topología de red.

6.1. Justificación de la capacidad de topología incremental y mecanismo para añadir unidades ocultas

Como se ha comentado, el primer problema a resolver para poder desarrollar un algoritmo de aprendizaje capaz de adaptar la estructura de red es encontrar un mecanismo que permita añadir nuevas neuronas ocultas sin necesidad de reinicializar el entrenamiento y perder el conocimiento aprendido anteriormente. En esta sección se describen las variaciones que se han incluido en el algoritmo inicial presentado en el Capítulo 5, y que permiten comprobar la idoneidad del método para adaptar de manera dinámica la topología de red.

El método de aprendizaje descrito en el Capítulo 5 presenta ciertas características que permiten pensar en la posibilidad de realizar una modificación dinámica de la topología de red durante el proceso de aprendizaje, manteniendo el conocimiento adquirido en las iteraciones anteriores. La inclusión de nuevas unidades ocultas en una red de neuronas implica la creación de nuevos pesos que, por un lado, conecten cada entrada de la red con cada nueva unidad, y por otro, que relacionen las nuevas neuronas ocultas con las unidades de salida. En la figura 6.1 se presenta de manera gráfica la situación de una red de neuronas al incluir nuevas unidades en la capa oculta. Las líneas de puntos indican las nuevas conexiones que han de establecerse como consecuencia de la ampliación de la capa intermedia. Como se ha explicado en el capítulo previo, el método de aprendizaje *online* para redes de neuronas de dos capas (considerada como la composición de dos redes de una sola capa), emplea sistemas de ecuaciones lineales para calcular los pesos de ambas subredes. Estos sistemas vienen descritos por las ecuaciones,

$$\begin{aligned} \mathbf{A}_k^{(1)}(s)\mathbf{w}_k^{(1)}(s) &= \mathbf{b}_k^{(1)}(s), \quad k = 1, \ldots, K, \\ \mathbf{A}_j^{(2)}(s)\mathbf{w}_j^{(2)}(s) &= \mathbf{b}_j^{(2)}(s), \quad j = 1, \ldots, J. \end{aligned}$$

Las matrices $\mathbf{A}^{(l)}$ y los vectores $\mathbf{b}^{(l)}$ de coeficientes ($l = 1, 2$) asociados a cada neurona de salida de cada una de las subredes, contienen la información necesaria para

calcular los pesos de la red. Existe la posibilidad de modificar, de manera sencilla, tales matrices y vectores al añadir filas y/o columnas de elementos. De este modo, sus dimensiones se adaptan en función de las necesidades del proceso de aprendizaje. Estas nuevas estructuras de datos (matrices **A** y vectores **b** modificados) permiten obtener los pesos para la topología de red actual.

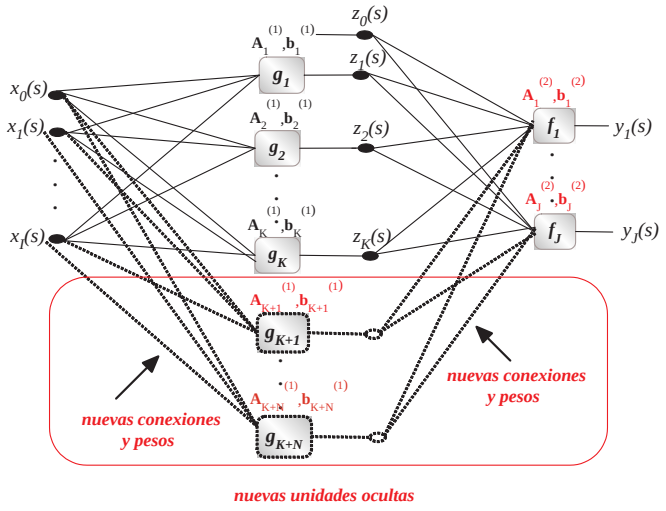


Figura 6.1: Adaptación de la estructura de red cuando se incorpora una o varias unidades en su capa oculta.

Aunque de diferente manera, el incremento en el número de unidades afecta tanto a la primera como a la segunda de las subredes. En el caso de la primera subred implica un aumento de sus neuronas de salida, mientras que es el número de unidades de entrada el que varía para la segunda de las subredes. Por tanto, las modificaciones que deben llevarse a cabo difieren en función de la subred que se considere, aunque en ambos casos, matrices **A** y vectores **b** deben ser redimensionados de manera que permitan calcular adecuadamente los pesos para la nueva topología de la red.

Teniendo presentes las consideraciones mencionadas se describe el método de aprendizaje con topología incremental. Hay que tener en cuenta que el algoritmo sigue las líneas del método presentado en el capítulo previo pero ahora se incluyen las mejoras necesarias para la modificación de la topología. De este modo, se considera una red de

neuronas de dos capas, como la que se presenta en la figura 6.2 como la composición de dos redes de una sola capa.

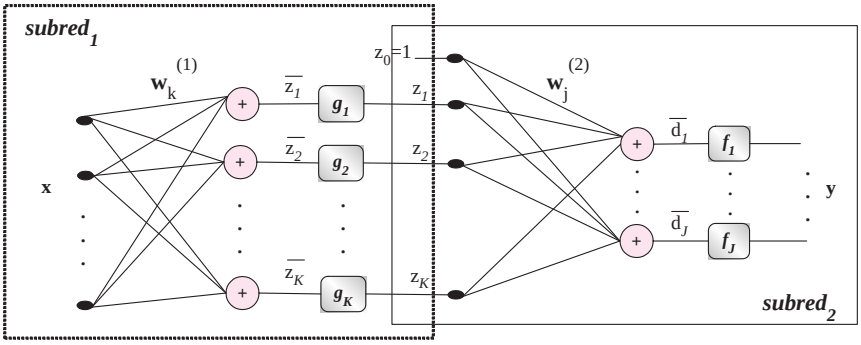


Figura 6.2: Red de neuronas de dos capas con alimentación hacia delante.

Como se explicó en el capítulo 5 teniendo en cuenta que $z_k(s)$ es la salida deseada de la neurona oculta k para el patrón de entrenamiento s, $\bar{z}_k(s) = g_k^{-1}(z_k(s))$ y $\mathbf{w}_k^{(1)}$ son los pesos asociados a la neurona oculta k, se emplea la siguiente la función objetivo para cada salida k de la subred 1:

$$Q_k^{(1)}(s) = h_k(s)\left(g_k^{'}(\bar{z}_k(s))\left(\mathbf{w}_k^{(1)^T}(s)\mathbf{x}(s) - \bar{z}_k(s)\right)\right)^2, \quad k = 1, \ldots, K, \tag{6.1}$$

donde s indica la iteración de entrenamiento actual que se corresponde con la muestra de entrenamiento $s - ésima$. De manera análoga, la función de coste para cada salida j de la segunda subred se define como,

$$Q_j^{(2)}(s) = h_j(s)\left(f_j^{'}(\bar{d}_j(s))\left(\mathbf{w}_j^{(2)^T}(s)\mathbf{z}(s) - \bar{d}_j(s)\right)\right)^2, \quad j = 1, \ldots, J. \tag{6.2}$$

siendo $\bar{d}_j(s) = f_j^{-1}(d_j(s))$, y los términos $h_j(s)$, $h_k(s)$ las funciones de olvido que determinan la importancia del error en la $s - ésima$ muestra.

Los pesos de las capas 1 y 2 se obtienen minimizando la función de coste correspondiente, lo que equivale a resolver los sistemas de ecuaciones lineales para cada una de las subredes, de la forma

$$\mathbf{A}\mathbf{w} = \mathbf{b}, \tag{6.3}$$

donde, para la primera subred, los componentes de $\mathbf{A}$ y $\mathbf{b}$ se definen como

$$\mathbf{A}_k^{(1)}(s) = \mathbf{A}_k^{(1)}(s-1) + h_k(s)\mathbf{x}(s)\mathbf{x}^T(s)g_k'^2(\bar{z}_k(s)); k = 1, \ldots, K, \quad (6.4)$$

$$\mathbf{b}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s-1) + h_k(s)g_k^{-1}(z_k(s))\mathbf{x}(s)g_k'^2(\bar{z}_k(s)); k = 1, \ldots, K, \quad (6.5)$$

y para la segunda subred, como

$$\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s)f_j'^2(\bar{d}_j(s)); j = 1, \ldots, J,, \quad (6.6)$$

$$\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s)f_j^{-1}(d_j(s))\mathbf{z}(s)f_j'^2(\bar{d}_j(s)); j = 1, \ldots, J. \quad (6.7)$$

En este momento, para que el algoritmo incluya la capacidad de adaptar la estructura de la red en función de las necesidades del proceso de aprendizaje hay que incluir ciertas modificaciones. El algoritmo requiere aumentar el número de neuronas ocultas en una o varias unidades y, por tanto, redimensionar los pesos de manera adecuada. El aumento del número de neuronas ocultas afecta a ambas capas de la red. En la figura 6.3 se puede observar cómo el incremento en el número de neuronas ocultas afecta a la primera subred al aumentar su número de unidades de salida, ya que es necesario considerar nuevos pesos que permitan conectar las unidades de entrada con la nueva neurona de salida de la subred. Por esta razón, y para cada nueva unidad oculta $k+1$, se crea una matriz de coeficientes $\mathbf{A}_{k+1}^{(1)}$, con $(I+1)\times(I+1)$ elementos y un vector $\mathbf{b}_{k+1}^{(1)}$ de tamaño $(I+1)$, donde I indica el número de entradas. Inicialmente estas estructuras contienen elementos de valor cero.

Además, el incremento de neuronas ocultas implica también modificaciones en la segunda subred. Como puede comprobarse en la figura 6.3 se produce un incremento en su número de entradas. En consecuencia, en este segundo caso, todas las matrices $\mathbf{A}_j^{(2)}$ y los vectores $\mathbf{b}_j^{(2)}$ $(j = 1, \ldots, J)$ previamente calculados deben modificar su tamaño. Para realizar esta adaptación se incluyen, en cada matriz $\mathbf{A}_j^{(2)}$ tantas filas y columnas de valores cero como número de unidades se añaden en la capa oculta de la red. Al mismo tiempo, cada vector $\mathbf{b}_j^{(2)}$ añade tantos elementos de valor cero como número de unidades se incorporan en la capa oculta. El resto de elementos de las estructuras se

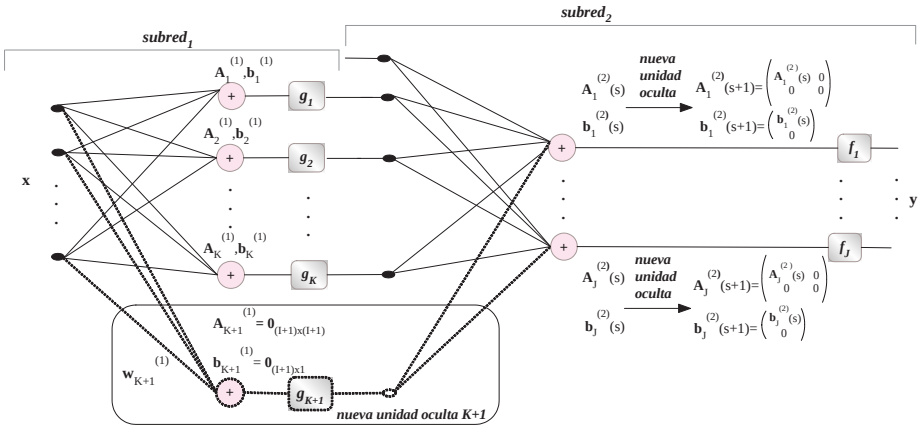


Figura 6.3: Modificación de los parámetros de la red para la topología incremental.

mantienen y esto permite conservar, en cierta medida, el conocimiento adquirido en las iteraciones previas con la topología anterior.

Tras estas modificaciones, el proceso de aprendizaje continua para obtener el nuevo conjunto de pesos para la nueva topología de la red. El Algoritmo 6.1 detalla el método de aprendizaje *online* e incremental con las modificaciones realizadas para la inclusión de nuevas unidades de la capa oculta. Los pasos del 17 al 26 corresponden a la fase de adaptación de la red cuando se fuerza el crecimiento de la estructura al aumentar las unidades en su capa oculta.

6.1.1. Resultados experimentales

En esta sección se comprueba la viabilidad del método de aprendizaje propuesto mediante su aplicación a diferentes problemas de identificación de sistemas. El estudio experimental que se plantea persigue un doble objetivo. En primer lugar, comprobar que el método propuesto es capaz de añadir nuevas unidades en la capa oculta sin deteriorar el rendimiento de forma significativa. En segundo lugar, se desea conocer si el rendimiento que alcanza es similar al que lograría empleando la topología final obtenida desde el principio del proceso de aprendizaje, es decir una topología fija.

Algoritmo 6.1 Algoritmo *online* e incremental con capacidad de adaptación a los cambios y topología adaptativa (manual) para redes de dos capas con alimentación hacia delante

Entradas: $\mathbf{x}(s) = (x_1(s), x_2(s), \ldots x_I(s)); \mathbf{d}(s) = (d_1(s), d_2(s), \ldots d_J(s));\ s = 1, \ldots, S.$

sesgos, $x_0(s) = 1, z_0(s) = 1$

1. Fase de Inicialización
2. $\mathbf{A}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times(I+1)}, \quad \mathbf{b}_k^{(1)}(0) = \mathbf{0}_{(I+1)}, \quad \forall k = 1, \ldots, K.$
3. $\mathbf{A}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times(K+1)}, \quad \mathbf{b}_j^{(2)}(0) = \mathbf{0}_{(K+1)}, \quad \forall j = 1, \ldots, J.$
4. Los pesos iniciales, $\mathbf{w}_k^{(1)}(0)$, se calculan por medio de algún método de inicialización.
5. Para la muestra actual s ($s = 1, 2, \ldots$,S)
6. $z_k(s) = g(\mathbf{w}_k^{(1)}(0), \mathbf{x}(s)), \forall k = 1, \ldots, K.$
7. Para cada salida k de la subred 1 ($k = 1, \ldots, K$),
8. $\mathbf{A}_k^{(1)}(s) = \mathbf{A}_k^{(1)}(s-1) + h_k(s)\mathbf{x}(s)\mathbf{x}^T(s)g_k^{'2}(\bar{z}_k(s))$ (véase ecuación 6.4).
9. $\mathbf{b}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s-1) + h_k(s)g_k^{-1}(z_k(s))\mathbf{x}(s)g_k^{'2}(\bar{z}_k(s))$ (véase ecuación 6.5).
10. Calcular $\mathbf{w}_k^{(1)}(s)$ resolviendo el sistema de ec. lineales $\mathbf{w}_k^{(1)}(s) = {\mathbf{A}_k^{(1)}}^{-1}(s)\mathbf{b}_k^{(1)}(s)$.
11. fin del Para.
12. Para cada salida j de la subred 2 ($j = 1, \ldots, J$),
13. $\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s)f_j^{'2}(\bar{d}_j(s))$ (véase ecuación 6.6).
14. $\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s)f_j^{-1}(d_j(s))\mathbf{z}(s)f_j^{'2}(\bar{d}_j(s))$ (véase ecuación 6.7).
15. Calcular $\mathbf{w}_j^{(2)}(s)$ resolviendo el sistema de ec. lineales $\mathbf{w}_j^{(2)}(s) = {\mathbf{A}_j^{(2)}}^{-1}(s)\mathbf{b}_j^{(2)}(s)$.
16. fin del Para.
17. Si se produce un cambio en la topología de la red,
18. Para cada nueva salida $m = 1, \ldots, M$ de la subred 1,
19. $\mathbf{A}_m^{(1)}(s) = \mathbf{0}_{(I+1)\times(I+1)}, \quad \mathbf{b}_m^{(1)}(s) = \mathbf{0}_{(I+1)},$
20. fin del Para.
21. Para cada salida j de la subred 2 ($j = 1, \ldots, J$),
22. $$\mathbf{A}_j^{(2)}(s) = \begin{pmatrix} \mathbf{A}_j^{(2)}(s-1) & 0 & \overset{M}{\cdots} & 0 \\ 0 & 0 & 0 & 0 \\ M\ \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{b}_j^{(2)}(s) = \begin{pmatrix} \mathbf{b}_j^{(2)}(s-1) \\ 0 \\ M\ \vdots \\ 0 \end{pmatrix},$$
23. fin del Para.
24. Obtener pesos iniciales para las nuevas conexiones por algún método de inicialización.
25. $K = K + M.$
26. fin del Si.
27. fin del Para.

Para comprobar el rendimiento del método propuesto comparamos diferentes configuraciones del mismo: (a) con topología fija y capacidad de adaptación a los cambios, (b) con topología adaptativa, (c) con topología adaptativa y capacidad de adaptación a los cambios, y finalmente (d) con topología adaptiva pero reinicializando las matrices **A** y los vectores **b** de coeficientes cada vez que se modifica la estructura de manera que el sistema olvida todo el conocimiento adquirido previamente cuando se produce una variación en la topología de red.

Para todos los experimentos realizados, las diferentes configuraciones del método de aprendizaje propuesto comparten las siguientes condiciones:

- En todos los casos se emplea la función logística sigmoide para las neuronas de la capa oculta, mientras que para las unidades de la capa de salida se aplican funciones lineales como se recomienda en el caso de problemas de regresión [21].
- Los datos de entrada que se proporcionan a la red están normalizados (con media 0 y desviación típica 1).
- Con el objeto de obtener resultados significativos se realizaron 5 simulaciones, de manera que los resultados presentados son la media de las pruebas realizadas.
- En todos los casos se empleó una función de olvido exponencial, definida como:

$$h_j(s) = h_k(s) = e^{\mu s}, j = 1, \ldots, J; k = 1, \ldots, K; s = 1, \ldots, S; \tag{6.8}$$

  para las J y K salidas de ambas subredes (véase figura 6.2).

A mayores, cabe mencionar que en cada experimento se emplea una topología arbitraria ya que el objetivo de la experimentación no es investigar la topología óptima de la red sino mejorar el rendimiento del método para una topología dada. En el estudio realizado se emplearon cinco series temporales diferentes, tres de ellas son reales y otras dos artificiales. Además se comprueba el comportamiento del algoritmo de aprendizaje cuando trabaja en diferentes tipos de entorno, estacionarios y no estacionarios.

Las siguientes secciones incluyen los resultados obtenidos para varios experimentos que simulan distintos escenarios.

6.1.1.1. Contextos estacionarios

En esta sección se consideran tres series temporales estacionarias diferentes: Lorenz [91], K.U. Leuven [132] y ratio de cambio monetario entre Dólares y Libras (US-UK). Los dos primeros se obtuvieron de una página web de series temporales [142] mientras que el último de ellos se extrajo de la página web de datos de la Reserva Federal [1]. En esta caso, a las condiciones experimentales establecidas anteriormente, se añaden las siguientes más específicas:

- Con el objeto obtener resultados significativos, se aplicó la técnica de validación cruzada 10-paquetes para calcular el rendimiento final. De este modo los resultados presentados se corresponden con valores medios.
- Para las topologías fijas se emplean 5 y 10 unidades ocultas y en el caso de la estructura adaptativa, la topología varía entre 5 y 10 unidades añadiendo una única neurona de cada vez.
- El valor del parámetro μ del término de olvido se establece a $0,01$ para los conjuntos Lorenz y Leuven y, a $0,05$ para el caso de US-UK. Para este último conjunto se seleccionó un valor más alto debido a la mayor variabilidad de la señal, debido a la cual se producen más cambios puntuales a los que se adaptaría la red si el valor de μ es más bajo.

A continuación, en la tabla 6.1 se presentan las características principales para los conjuntos de datos seleccionados.

En el estudio experimental y, para cada uno de los conjuntos de datos, las figuras presentadas muestran distintas gráficas. En la primera de ellas, véase la figura 6.4,

Datos	**Nº entradas**	**Nº muestras**
Lorenz	8	5.000
K. U. Leuven	8	1.800
US-UK	8	456

Tabla 6.1: Características de los conjuntos de regresión empleados como ejemplos de contextos estacionarios.

se presentan las curvas ECM para la fase de test obtenidas por dos configuraciones del método propuesto con capacidad de adaptación a los cambios: (1) con topología incremental, empezando con un número inicial de unidades ocultas y aumentando hasta alcanzar una topología final establecida, y (2) una topología fija, proporcionando en este caso los resultados de las topologías inicial y final utilizadas en el caso incremental. En esta figura se puede observar cómo en caso de que la topología inicial sea insuficiente para resolver el problema, el método con topología incremental puede adaptarse de tal manera que es capaz de alcanzar resultados similares a aquellos que obtiene la topología fija final entrenada desde el principio del proceso de aprendizaje. En la curva ECM correspondiente al método con topología adaptativa se observan varios picos en diferentes instantes del tiempo que son consecuencia del aumento de unidades ocultas en la red. Cuando la estructura de red modifica su tamaño el error sufre una ligera degradación debido a la incorporación de nuevos parámetros (pesos) que todavía no han sido ajustados.

En la figura 6.5, se presentan las curvas ECM del método propuesto con topología incremental para las configuraciones con y sin capacidad de adaptación a los cambios ($\mu = 0$) con el objetivo de comprobar si el factor de olvido es útil cuando se produce un cambio en la topología. Se puede observar cómo, a pesar de tratarse de un contexto estacionario, el método sin capacidad de adaptación deteriora el rendimiento puesto que no se recupera adecuadamente tras una modificación de la topología de red. Esto es debido a que acumula el conocimiento adquirido con otras topologías y no le concede mayor importancia al nuevo, este hecho provoca que el error quede estancado

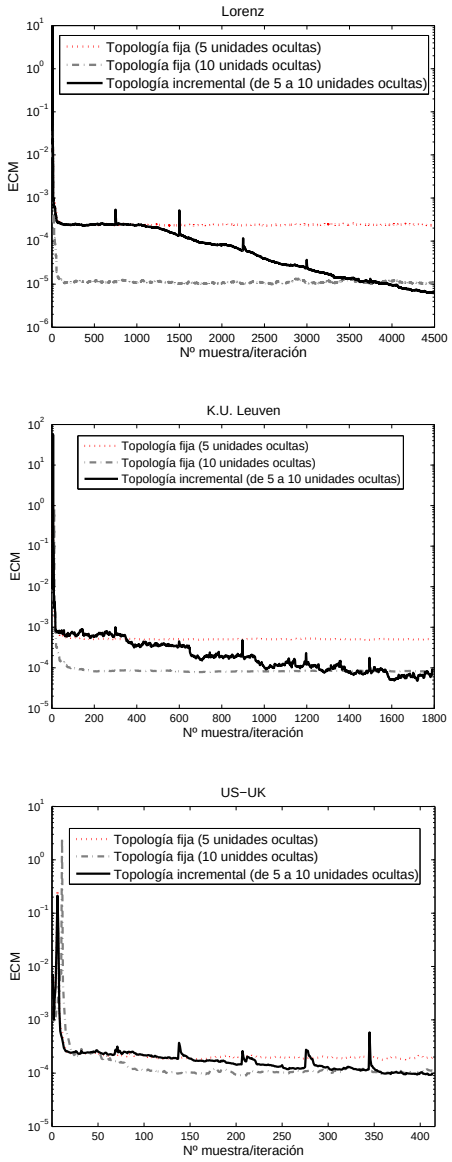


Figura 6.4: Contextos estacionarios. Curvas ECM para la fase de test comparando el método con topología fija y topología incremental.

en un determinado valor sin poder mejorar su rendimiento a lo largo del aprendizaje. Sin embargo, la configuración del método con capacidad de adaptación concede mayor importancia al conocimiento reciente frente al adquirido previamente y, de este modo, consigue que el aprendizaje avance de manera adecuada empleando el modelo de aprendizaje previo.

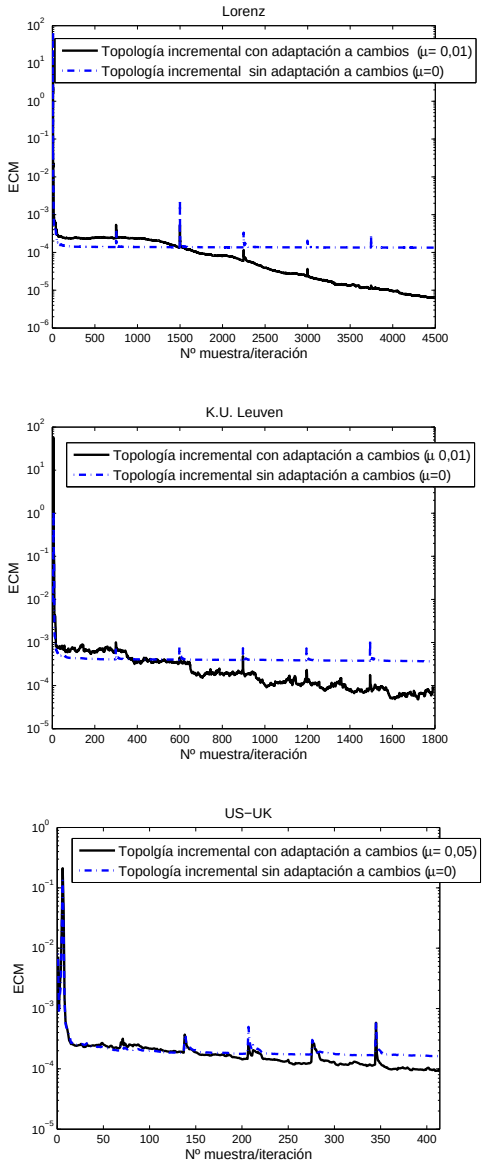


Figura 6.5: Contextos estacionarios. Curvas ECM para la fase de test comparando el método con topología incremental con y sin capacidad de adaptación a los cambios.

Finalmente, y con el objetivo de comprobar cómo influye el hecho de almacenar el conocimiento previamente adquirido se presenta la figura 6.6. En ella se muestran las curvas del ECM obtenidas por el método con topología incremental con capacidad de adaptación y una aproximación del método incremental que olvida el conocimiento previo cuando se produce un cambio de topología, debido al reinicio de los valores de

las matrices y vectores de coeficientes que almacenan el conocimiento previamente adquirido. En la figura se puede observar cómo la aproximación que reinicia las matrices obtiene un deterioro importante en el rendimiento cada vez que se produce una modificación en la topología ya que olvida todo el conocimiento previo y tiene que empezar un nuevo aprendizaje desde cero cada vez que se produce un cambio. Sin embargo, el método propuesto conserva la información aprendida con las topologías de red anteriores de manera que, gracias a ese conocimiento almacenado, cuando se produce una variación en la topología de red su rendimiento únicamente sufre un leve deterioro del que se repone rápidamente en pocas iteraciones.

En último lugar y para completar esta parte del estudio se realiza otro experimento con el último de los conjuntos de datos (US-UK). El objetivo es mostrar cómo el método propuesto es capaz de añadir varias unidades ocultas al mismo tiempo, en lugar de incluirlas de una en una como se ha visto hasta ahora. En este caso, la topología incremental varía su estructura de 5 a 25 neuronas ocultas, incluyendo las unidades de 5 en 5. El valor del factor μ se establece a 0,01. Los resultados, presentados en la figura 6.7, son similares a los de los experimentos previos pero se observa una pequeña diferencia en el comportamiento de la estructura adaptativa que refleja la figura 6.7(a). Cada vez que se modifica la topología de red y se añade un grupo de neuronas, el rendimiento sufre un deterioro mayor. Este hecho se debe a que la incorporación de muchos parámetros (pesos) al mismo tiempo, implica una variación mayor en las matrices de coeficientes (tantas filas/columnas de valores cero como unidades se incorporan). Por tanto, en este caso, el deterioro con respecto a la iteración previa es mayor que si se añade una única neurona (como ocurría en los ejemplos previos). A pesar de estas degradaciones puntuales el método con topología incremental se recupera rápidamente.

En cuanto al comportamiento de la configuración sin capacidad de adaptación a los cambios (figura 6.7(b)) se observa como cada vez que se modifica la topología se produce un incremento elevado del error. A su vez, en la figura 6.7(c) se comprueba que el reinicio de las matrices y vectores de coeficientes provoca una degradación muy importante en el error que comete el método y del que se recupera lentamente. Por tanto, ninguna de estas dos aproximaciones resulta ser adecuada.

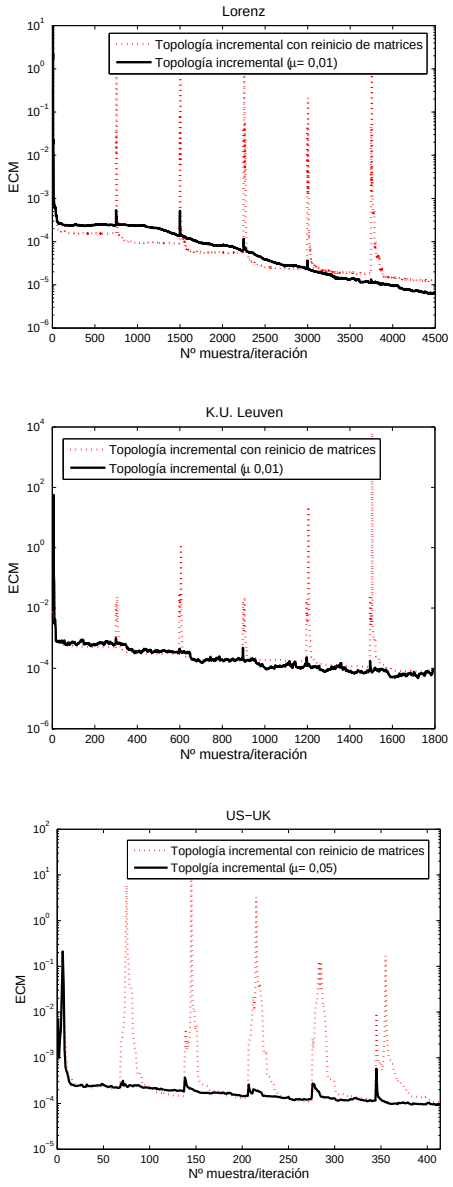
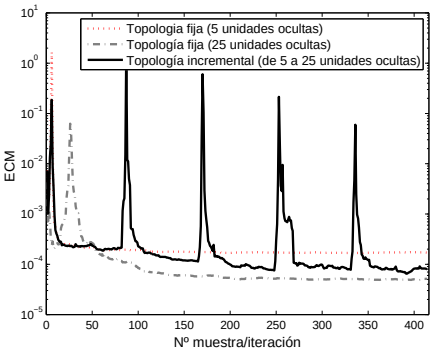
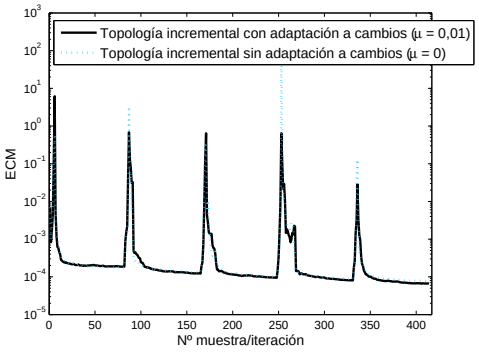


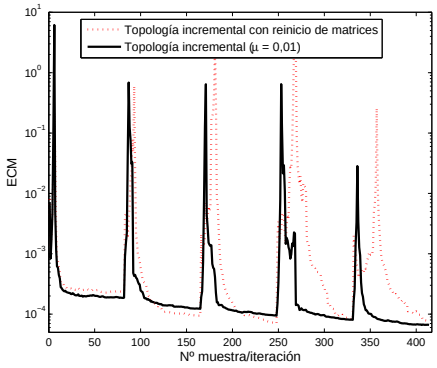
Figura 6.6: Contextos estacionarios. Curvas ECM para la fase de test comparando el método con topología incremental con capacidad de adaptación a los cambios y con reinicio del conocimiento previo.



(a)



(b)



(c)

Figura 6.7: Ejemplo para el conjunto de datos US-UK. Curvas ECM para la fase de test comparando diferentes versiones del método, (a) topología fija y topología incremental, (b) topología incremental con y sin capacidad de adaptación a los cambios, (c) topología incremental con capacidad de adaptación frente al método con reinicio del conocimiento previo.

6.1.1.2. Contextos no estacionarios

En entornos dinámicos el concepto de salida deseada puede cambiar en el tiempo. Los cambios entre contextos pueden ser bruscos cuando la distribución cambia rápidamente o graduales si existe una transición suave entre las distribuciones [15, 147]. Para comprobar el rendimiento del método propuesto en diferentes situaciones, se consideran ambos tipos de cambios. Como la señal evoluciona en el tiempo, se generan varios

cambios de contexto y como resultado, se dispone de un conjunto de test diferente para cada uno de ellos. Por tanto, cada muestra de entrenamiento tiene asociado un conjunto de test que representa el contexto al que pertenece la muestra. Para los siguientes experimentos, todos los conjuntos de test contienen 150 muestras.

Conjunto de datos artificial 1. El primer conjunto de datos está formado por 4 variables de entrada aleatorias que contienen valores procedentes de una distribución normal con media cero y desviación típica 0,1. La salida deseada se obtiene mediante una mezcla lineal de funciones no lineales sobre todas las variables de entrada. Las funciones no lineales empleadas son sigmoide tangente hiperbólica, exponencial, seno y sigmoide logarítmica. Para cada contexto, se aplican diferentes combinaciones de estas funciones no lineales. La señal de salida deseada se modifica cada 500 muestras empleando cada vez una fila diferente de la siguiente matriz de mezcla.

$$M = \begin{pmatrix} 2{,}1 & 1{,}3 & -1{,}1 & 1{,}3 \\ -1{,}1 & -0{,}3 & 0{,}7 & -1{,}1 \\ -0{,}1 & 0{,}2 & 1{,}8 & -2{,}3 \\ -3{,}1 & -1{,}5 & 0{,}4 & 1{,}8 \\ 1{,}5 & -0{,}9 & 1{,}2 & 0{,}6 \end{pmatrix}.$$

De este modo, se obtiene un conjunto de entrenamiento de 2.500 muestras y 5 conjuntos de test, uno por cada cambio de la mezcla lineal sobre el conjunto de entrenamiento. La figura 6.8 contiene la señal empleada como salida deseada durante el proceso de entrenamiento. En este ejemplo, se emplean 10 y 15 neuronas en la capa oculta para las topologías fijas, mientras que la topología incremental añade las unidades de una en una, realizando adaptaciones de la estructura cada cierto número de iteraciones del proceso de aprendizaje. Para comparar con el algoritmo con capacidad de adaptación a los cambios se establece el valor del factor μ del término de olvido a 0,01.

La figura 6.9(a) muestra los resultados de test obtenidos por el método propuesto con topologías fija e incremental. En las figuras de test presentadas, cada punto de la

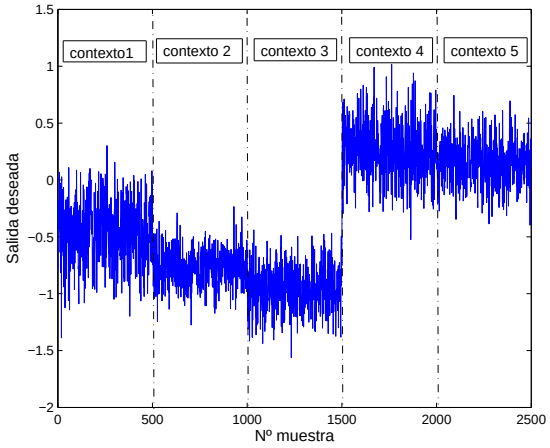


Figura 6.8: Contextos no estacionarios. Salida deseada para el conjunto de entrenamiento del conjunto de datos artificial 1 con cambios bruscos cada 500 muestras.

señal corresponde al valor medio obtenido en el conjunto de test correspondiente para la muestra de entrenamiento actual. Se puede observar como el método propuesto con topología incremental empezando desde una arquitectura de 10 unidades, es capaz de obtener mejores resultados que el método con topología fija con 15 neuronas ocultas. En las curvas de error del método se observan dos tipos de picos diferentes que indican degradaciones puntuales en el rendimiento. Los más bruscos aparecen cada 500 iteraciones como consecuencia de la aparición de un nuevo contexto debido a los cambios que sufre la señal de entrada. Los otros picos, más pequeños, se producen aproximadamente cada 420 muestras y se deben a un cambio en la estructura de red. En el primer caso, los picos se traducen en una degradación mayor debido a la aparición repentina de un cambio en la señal a modelar. Sin embargo, el método es capaz de recuperarse rápidamente gracias a su capacidad de adaptación a los cambios. Las variaciones de topología implican un deterioro ligero del error, en muchos casos casi inapreciable, como consecuencia de la modificación de matrices y vectores de coeficientes debido a la incorporación de una nueva unidad oculta.

La figura 6.9(b) presenta las curvas de error para la topología incremental con y sin capacidad de olvido. Como se puede comprobar, cuando el método no dispone de

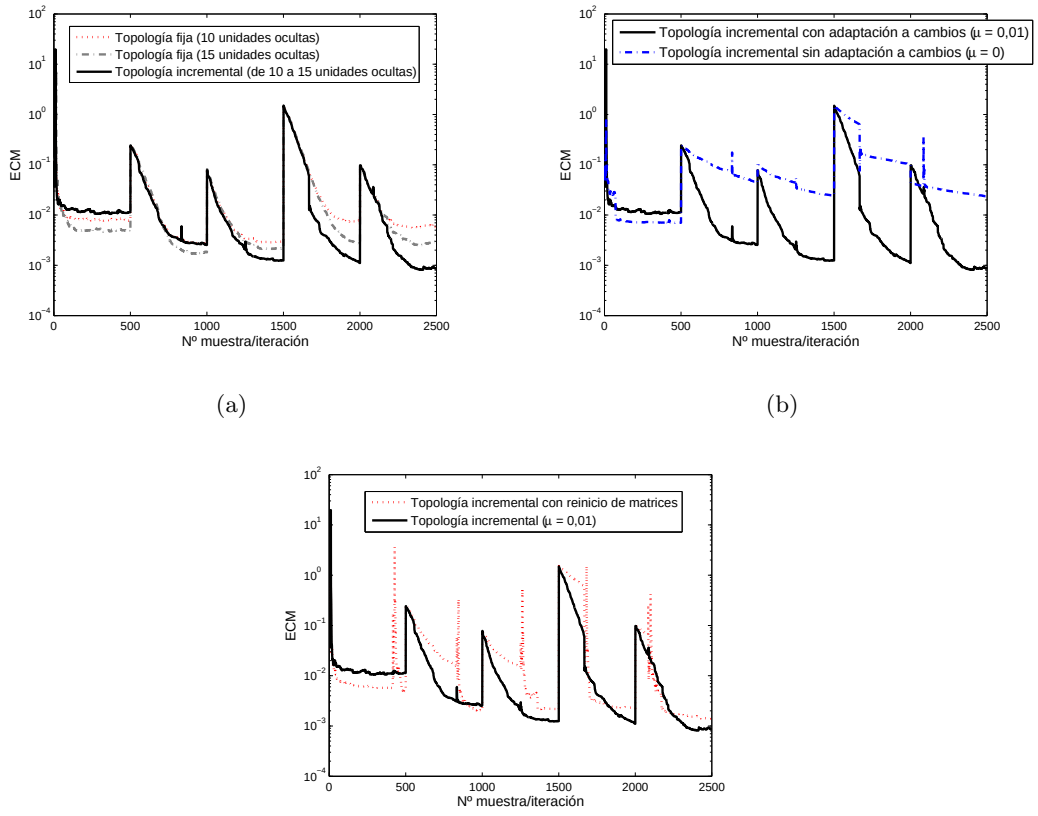


Figura 6.9: Conjunto de Datos Artificial 1. Curvas ECM para la fase de test comparando diferentes configuraciones del método, (a) topología fija y topología incremental, (b) topología incremental con y sin capacidad de adaptación a los cambios, (c) topología incremental con capacidad de adaptación frente al método con reinicio del conocimiento previo.

la capacidad de adaptación a los cambios se produce un deterioro en el rendimiento. En la figura se puede observar como después de un cambio de contexto el método no es capaz de recuperarse debido a que pondera del mismo modo todos los errores cometidos. Por otro lado, si se modifica la topología de red el incremento del error es menor, pero aún así el hecho de otorgar la misma importancia a todos los errores supone una recuperación lenta. Sin embargo, cuando el método dispone de la capacidad de adaptación concede mayor importancia al conocimiento reciente y consecuentemente

a pesar de la aparición de cambios de contexto o a variaciones en la topología de red, es capaz de recuperarse permitiendo que el aprendizaje avance de manera adecuada empleando el modelo previamente aprendido.

Finalmente, en la figura 6.9(c), se presentan las curvas ECM obtenidas para el método con topología incremental con capacidad de adaptación y un aproximación del mismo que olvida el conocimiento previo cuando se modifica la topología de red. El hecho de reiniciar las matrices de coeficientes que representan el conocimiento previo cuando se modifica la topología lleva asociado un deterioro importante en el rendimiento, ya que se reinicia el proceso de aprendizaje cada vez que se añade una nueva unidad a la capa oculta y se maneja una nueva topología.

Conjunto de datos artificial 2. En este caso se genera un conjunto de datos artificial que presenta una evolución *gradual* continua en cada muestra de entrenamiento. El conjunto está formado por 4 variables generadas mediante una distribución normal con media cero y desviación típica 0,1. La salida se obtiene por medio de una mezcla no lineal sobre las mismas fijando los coeficientes iniciales del vector de mezcla como,

$$a(0) = \left[\begin{array}{cccc} 0{,}5 & 0{,}2 & 0{,}7 & 0{,}8 \end{array} \right],$$

y evolucionando estos coeficientes en el tiempo de acuerdo a la siguiente ecuación,

$$a_j(s) = a_j(s-1) + \frac{i}{10^{4,7}} logsig(a_j(s-1)), s = 1, \ldots, S,$$

donde el subíndice j indica la componente del vector de mezcla. El conjunto de datos dispone de un total de 500 muestras de entrenamiento y un conjunto diferente de test para cada una de ellas, debido a la evolución continua de la señal que origina un estado diferente para cada una de las muestras de entrenamiento. Finalmente obtenemos un conjunto de datos cuya salida deseada durante el proceso de entrenamiento se presenta en la figura 6.10. Para este conjunto de datos se emplean 10 y 15 neuronas ocultas y las unidades se añaden de una en una en el caso de la estructura adaptativa. Para la capacidad de olvido se establece un valor de μ=0,01.

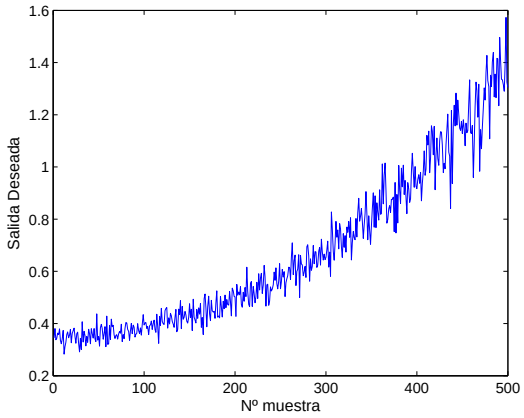
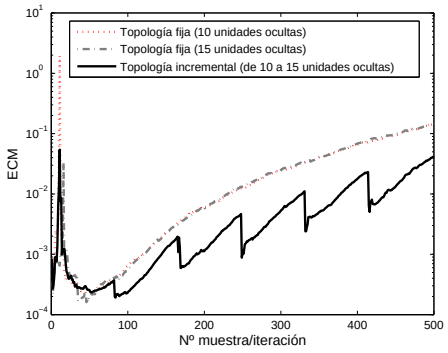


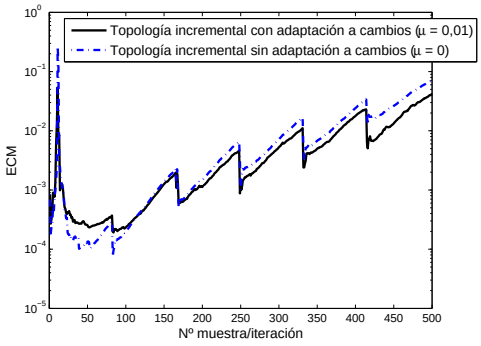
Figura 6.10: Contextos no estacionarios. Salida deseada para el conjunto de entrenamiento del conjunto de datos artificial 2 con cambios graduales continuos.

Las curvas de error obtenidas por el método propuesto con topologías fija e incremental se presentan en la figura 6.11(a). Como se puede observar, en ambos casos, el error crece de manera continuada como consecuencia de los continuos cambios graduales que sufre la señal de entrada. En este caso, debido a la rápida evolución del proceso a modelar, el hecho de ir almacenando el conocimiento previo no aporta tantas ventajas como en los anteriores conjuntos de datos. Aun así, el método incremental supera los resultados obtenidos por la topología fija. Además, hay que señalar que al añadir una nueva neurona oculta se produce una mejora puntual en su rendimiento (en la gráfica se refleja mediante ciertos picos hacia abajo). Esto se debe a la incorporación de nuevas filas/columnas de elementos con valor cero en las matrices y vectores de coeficientes que almacenan el conocimiento previo. Estas modificaciones causan un efecto de olvido restando importancia al conocimiento previamente adquirido.

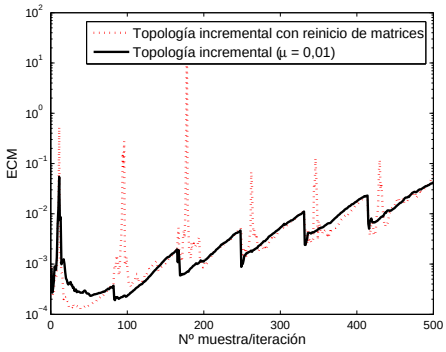
Con respecto a las curvas de error obtenidas por el método con topología incremental con y sin capacidad de adaptación a los cambios (véase la figura 6.11(b)) se observa cómo las diferencias en el rendimiento no son de magnitud importante como ocurría en los ejemplos anteriores. Estos resultados avalan las conclusiones extraídas en el párrafo anterior respecto a la menor importancia que, en casos de evolución rápida del proceso, tiene el conocimiento previo aprendido. No obstante, en la figura 6.11(c) se



(a)



(b)



(c)

Figura 6.11: Contextos no estacionarios: Conjunto de Datos Artificial 2. Curvas ECM para la fase de test comparando diferentes configuraciones del método, (a) topología fija y topología incremental, (b) topología incremental con y sin capacidad de adaptación a los cambios, (c) topología incremental con capacidad de adaptación frente al método con reinicio del conocimiento previo.

observa cómo la configuración propuesta es ligeramente mejor cuando se la compara con la aproximación que reinicia el conocimiento previo al completo y que produce degradaciones importantes en el rendimiento cuando se modifica la topología debido a que debe empezar un nuevo proceso de aprendizaje.

6.2. Aprendizaje incremental con adaptación automática de la topología de red

En la sección anterior se ha demostrado y justificado la idoneidad del algoritmo propuesto de aprendizaje *online* para trabajar con estructuras de red adaptativas. Para ello, en el estudio experimental la modificación de las topologías de red se llevó a cabo de manera manual, realizando adaptaciones de la estructura cada cierto número constante de iteraciones del proceso de aprendizaje. En este punto, se hace necesario completar el método incremental de aprendizaje dotándolo de un mecanismo para controlar de manera automática el crecimiento de la estructura. De este modo, se consigue reducir la dificultad de la aplicación del método especialmente en aquellas situaciones en las que se dispone de un número de datos considerable o se requiere un elevado número de unidades ocultas. La topología debe ser modificada sólo si sus capacidades son insuficientes para satisfacer las necesidades del proceso de aprendizaje. Éste es el objetivo de la investigación que se presenta en esta sección.

El desconocimiento de la topología de red apropiada para modelar un proceso presenta dos inconvenientes importantes, que se solventan en gran medida al incluir una técnica automática para la determinación de una estructura de red adecuada. Los problemas mencionados son:

- Debido a que se desconoce cómo seleccionar el número óptimo de unidades ocultas que componen la estructura de la red, la decisión se basa en un método de prueba y error.
- Si el número disponible de datos de entrenamiento es grande y el número necesario de unidades ocultas también es elevado, resulta conveniente reducir en la medida de lo posible la demanda de recursos computacionales.

Recientemente, distintos trabajos de investigación han propuesto aproximaciones que alteran la estructura de la red a medida que evoluciona el proceso de aprendizaje.

Existen dos estrategias generales para conseguirlo. La primera implica emplear una red más grande de lo necesario y entrenar hasta que se encuentre una solución aceptable y, después de esto, eliminar aquellas unidades y pesos ocultos innecesarios. Los métodos que emplean esta aproximación se denominan métodos de podado (*pruning methods*) [118]. La otra aproximación realiza la búsqueda de la red adecuada en otra dirección, son los procedimientos constructivos (*constructive algorithms*) [21]. Estos métodos comienzan con una red de pequeñas dimensiones que posteriormente aumenta su tamaño al añadir unidades ocultas y pesos hasta encontrar una solución satisfactoria. De manera general se considera que la técnica de podado presenta varios inconvenientes frente a la aproximación constructiva, los más destacados se enumeran brevemente a continuación [81, 82]:

1. En los algoritmos constructivos se especifica una red inicial de manera directa mientras que en las técnicas de podado no se sabe, en la práctica, cómo de grande debe ser la red inicialmente.

2. Las técnicas constructivas buscan en primer lugar soluciones con redes pequeñas, esto supone un ahorro computacional frente a los métodos de podado que malgastan la mayor parte del tiempo de entrenamiento con redes más grandes de lo necesario.

3. Como es probable que redes de diferente tamaño implementen soluciones aceptables, es habitual que los algoritmos constructivos encuentren redes más pequeñas que las técnicas de podado.

Es difícil estimar, de manera teórica, el número exacto de unidades ocultas pero sin embargo existen diferentes trabajos que incorporan estructuras variables de red durante el proceso de entrenamiento. Así, Ash [10] desarrolló un algoritmo para la creación dinámica de nodos. En su propuesta, se genera una nueva unidad en la capa oculta cuando el ratio del error de entrenamiento está por debajo de un valor crítico (seleccionado de manera arbitraria). Hirose [63] adoptó una aproximación similar para la creación

de nodos pero al mismo tiempo también elimina unidades cuando se alcanzan valores pequeños del error. Aylward y Anderson [12] plantean un conjunto de reglas basadas en el ratio de error, el criterio de convergencia y la distancia al nivel de error deseado, de manera que se añade una nueva unidad oculta cuando las reglas no se satisfacen de manera continuada. Masters [96] propone la regla de la pirámide geométrica para establecer el número de unidades ocultas, basándose en el hecho de que la estructura de muchas redes sigue una forma piramidal, ya que el número de unidades decrece desde la capa de entrada a la de salida. Así, esta regla calcula el número de unidades de la capa oculta como $\sqrt{nm}$, siendo n el número de entradas y m el número de salidas de la red. Yao [154] y Fiesler [45] investigaron la aplicación de algoritmos evolutivos para optimizar el número de unidades ocultas y el valor de los pesos en un perceptrón multicapa. Mientras, Murata [101] investigó el problema de determinar el número óptimo de parámetros en las redes neuronales desde un punto de vista estadístico.

En la siguiente sección se explica la técnica de estimación automática que incorpora el método de aprendizaje propuesto en la sección anterior para adaptar apropiadamente su topología de red. De este modo se presenta un algoritmo completo de aprendizaje incremental respecto a su capacidad de adaptar, por un lado los parámetros y por otro, la estructura de la red a medida que se dispone de nuevos ejemplos de entrenamiento.

6.2.1. Técnica de adaptación automática de la topología de red

En la teoría de aprendizaje estadístico la dimensión de Vapnik-Chervonenkis (d_{VC}) [137] es una medida de la capacidad de un algoritmo de clasificación estadística y se define como la cardinalidad del conjunto de datos de mayor tamaño que el algoritmo puede dividir.

Se dice que un modelo de clasificación g con parámetros θ puede dividir un conjunto de datos $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ si, para todas las posibles etiquetas de clase de estos datos, existe un conjunto θ^* que evita que el modelo cometa errores en la clasificación de los

mismos. Consideremos un ejemplo sencillo en el que se consideraron dos clases y en el que emplea una recta como modelo de clasificación para separar los datos positivos de los negativos. Como puede comprobarse de manera representativa en la figura 6.12, con este modelo se pueden separar todos los posibles conjuntos de datos de tres elementos y 2 clases o etiquetas. Sin embargo, este modelo no permite dividir adecuadamente todos los conjuntos de cuatro puntos, tal y como se observa en la última gráfica de la figura 6.12. Por tanto, se dice que la d_{VC} del clasificador es 3, dado que esta es la cardinalidad máxima del conjunto de datos que este modelo puede clasificar a la perfección.

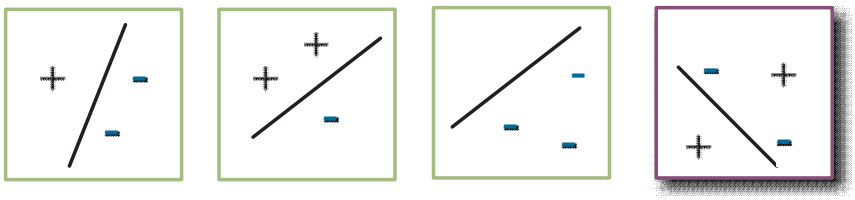


Figura 6.12: Dicotomías calculadas por una recta para conjuntos de 3 y 4 puntos.

Debido a que la d_{VC} de un modelo es la cardinalidad máxima del conjunto de datos que puede dividir, este término presenta una utilidad importante al permitir realizar una predicción de una cota superior para el error de test de un modelo de clasificación en función de su complejidad. Se trata de una cota ideal calculada de acuerdo al error de entrenamiento y la topología de red considerada. En caso de que el modelo esté generalizando de manera adecuada, la cota indica el peor error de test que puede alcanzar el modelo. En definitiva, el valor de la cota calculada permite establecer el margen en el que debe encontrarse el error de test, y por tanto es posible emplear dicho valor para conocer si la topología actual resulta suficiente.

De este modo, se puede decir que un número adecuado de unidades ocultas es aquel que permite obtener el menor riesgo esperado, es decir, el menor error de test posible. Teniendo en cuenta estas consideraciones se establece que, con una probabilidad de $1-\eta$, el menor error de test posible (R, riesgo esperado) está delimitado por la siguiente

desigualdad [137],

$$R(d_{VC},\theta) \leq \frac{\frac{1}{n}\sum_{i=1}^{n}\left(d_i - y_i\left(d_{VC},\theta\right)\right)^2}{\left[1-\sqrt{\frac{d_{VC}\left(ln\left(\frac{n}{d_{VC}}\right)+1\right)-ln\left(\frac{\eta}{4}\right)}{n}}\right]_+} \tag{6.9}$$

donde θ es el conjunto de parámetros ajustables del sistema de aprendizaje, d son las salidas deseadas, y las salidas obtenidas por el sistema de aprendizaje, n el número de patrones de entrenamiento y, la notación $[.]_+$ indica el valor máximo entre 0 y el término entre corchetes. Podemos observar que el numerador de la ecuación 6.9 es simplemente el ECM de entrenamiento, mientras que el denominador es una función de corrección.

De manera informal se puede decir que la capacidad de generalización de un modelo de clasificación está relacionada con la complejidad de su estructura. Concretamente, a mayor complejidad mayor valor de la d_{VC} y menor capacidad de generalización. En [17], Baum y Haussler establecen ciertos límites para una arquitectura de red particular. En concreto, proporcionan un límite superior para una red de neuronas alimentada hacia delante con un total de M unidades ($M \geq 2$) y W pesos (incluyendo sesgos) de la forma,

$$d_{VC} \leq 2Wlog_2(eM), \tag{6.10}$$

donde e es la base de los logaritmos naturales. Aunque la ecuación 6.10 establece un límite superior del valor d_{VC}, es una aproximación válida porque permite calcular la cota del error en el peor de los casos.

Teniendo en cuenta todas estas consideraciones, se desarrolla una técnica automática para controlar el crecimiento de la estructura de la red en el algoritmo de aprendizaje *online*. El método de estimación se basa en la cota ideal para el error de test que se puede calcular gracias a la d_{VC} de la red de neuronas considerada. De esta manera, se obtiene un algoritmo constructivo que añade unidades a la capa intermedia de la red siempre y cuando el error de test cometido sea mayor que el límite que establece la cota ideal calculada en la ecuación 6.9. El método de aprendizaje propuesto comienza con

una red de pequeñas dimensiones (inicialmente la red tiene dos unidades en su capa oculta), y adapta su topología (añade una única neurona de cada vez) en función de las circunstancias del proceso de aprendizaje.

De este modo, el algoritmo de aprendizaje propuesto, *online* e incremental con respecto a su capacidad de aprendizaje y su topología, se presenta de manera detallada en el Algoritmo 6.2. La variación principal con respecto al algoritmo descrito en la sección previa se concentra en los pasos del 17 al 27. Ese bloque de código hace referencia a la comprobación de la idoneidad de la estructura de red. Cuando la topología actual es insuficiente para resolver el problema planteado se realiza una adaptación de los parámetros oportunos para obtener una red dimensionada como corresponde a la nueva topología siguiendo el mecanismo descrito en la sección anterior. En caso contrario el proceso de aprendizaje continua sin realizar modificaciones.

6.2.2. Resultados experimentales

En esta sección se ilustra el método de aprendizaje con topología de red adaptativa mediante su aplicación a diferentes problemas de identificación de sistemas. El objetivo es comprobar el rendimiento del método y comparar su capacidad de generalización con respecto al método con topología de red fija.

6.2.2.1. Contexto estacionario.

Se consideran dos series temporales estacionarias, Henon y Mackey-Glass [142]. Para obtener resultados significativos se realizaron 5 simulaciones de manera que los resultados que se presentan a continuación corresponden con valores medios. El objetivo de la red es predecir la muestra actual en base a 7 muestras previas, en el caso de la serie de Henon, y en función de los 8 patrones inmediatamente anteriores para el conjunto

Algoritmo 6.2 Algoritmo *online* e incremental con topología adaptativa para redes de neuronas de dos capas con alimentación hacia delante.

Entradas: $\mathbf{x}(s) = (x_1(s), x_2(s), \ldots x_I(s)); \mathbf{d}(s) = (d_1(s), d_2(s), \ldots d_J(s));\ s = 1, \ldots, S.$

sesgos, $x_0(s) = 1, z_0(s) = 1$

1. Fase de Inicialización
2. $\mathbf{A}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times(I+1)}, \quad \mathbf{b}_k^{(1)}(0) = \mathbf{0}_{(I+1)}, \quad \forall k = 1, \ldots, K.$
3. $\mathbf{A}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times(K+1)}, \quad \mathbf{b}_j^{(2)}(0) = \mathbf{0}_{(K+1)}, \quad \forall j = 1, \ldots, J.$
4. Calcular los pesos iniciales, $\mathbf{w}_k^{(1)}(0)$, por medio de algún método de inicialización.
5. Para cada muestra de entrenamiento disponible s $(s = 1, \ldots, S)$,
6. $z_k(s) = g(\mathbf{w}_k^{(1)}(0), x(s)), \forall k = 1, \ldots, K.$
7. Para cada salida k de la subred 1 $(k = 1, \ldots, K)$,
8. $\mathbf{A}_k^{(1)}(s) = \mathbf{A}_k^{(1)}(s-1) + h_k(s)\mathbf{x}(s)\mathbf{x}^T(s)g_k^{'2}(\bar{z}_k(s))$ (véase ecuación 6.4).
9. $\mathbf{b}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s-1) + h_k(s)g_k^{-1}(z_k(s))\mathbf{x}(s)g_k^{'2}(\bar{z}_k(s))$ (véase ecuación6.5).
10. Calcular $\mathbf{w}_k^{(1)}(s)$ resolviendo el sistema de ec. lineales $\mathbf{A}_k^{(1)}(s)\mathbf{w}_k^{(1)}(s) = \mathbf{b}_k^{(1)}(s)$.
11. fin del Para.
12. Para cada salida j de la subred 2 $(j = 1, \ldots, J)$,
13. $\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s)f_j^{'2}(\bar{d}_j(s))$ (véase ecuación 6.6).
14. $\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s)f_j^{-1}(d_j(s))\mathbf{z}(s)f_j^{'2}(\bar{d}_j(s))$ (véase ecuación 6.7).
15. Calcular $\mathbf{w}_j^{(2)}(s)$ resolviendo el sistema de ec. lineales $\mathbf{A}_j^{(2)}(s)\mathbf{w}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s)$.
16. fin del Para.
17. Calcular la dimensión VC, $d_{VC} = 2Wlog_2(eM)$ (ecuación 6.10).
18. Calcular el error que comete la red sobre el conjunto de test, MSE_{Test}.
19. Obtener la cota del error de test de acuerdo a la ecuación 6.9.
20. Si $MSE_{Test} > cota$ entonces se añade una nueva unidad $K+1$ en la capa oculta,
21. $\mathbf{A}_{K+1}^{(1)}(s) = \mathbf{0}_{(I+1)\times(I+1)}, \quad \mathbf{b}_{K+1}^{(1)}(s) = \mathbf{0}_{(I+1)},$
22. Para cada salida j de la subred 2 $(j = 1, \ldots, J)$,
23. $\mathbf{A}_j^{(2)}(s) = \begin{pmatrix} \mathbf{A}_j^{(2)}(s-1) & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{b}_j^{(2)}(s) = \begin{pmatrix} \mathbf{b}_j^{(2)}(s-1) \\ 0 \end{pmatrix},$
24. fin del Para.
25. Obtener pesos iniciales para nuevas conexiones por algún método de inicialización.
26. $K = K + 1.$
27. fin del Si.
28. fin del Para.

de Mackey-Glass. La tabla 6.2 muestra el número de muestras de entrenamiento y test empleadas para cada una de las series.

Serie temporal	**Nº muestras entrenamiento**	**Nº muestras test**
Henon	3.000	1.000
Mackey-Glass	2.250	750

Tabla 6.2: Número de muestras de entrenamiento y test empleadas para las series temporales de Henon y Mackey-Glass.

En las figuras 6.13 y 6.14 se muestran las gráficas de test obtenidas para las series Henon y Mackey-Glass, respectivamente. En ellas se pueden observar las curvas de error para el método con topología fija e incremental. En el caso de las topologías fijas se incluyen las curvas correspondientes al ECM alcanzado con distinto número de neuronas ocultas. El objeto es comprobar cómo se comporta el método en función de su estructura y constatar si el comportamiento del método incremental es adecuado. Por motivos de legibilidad se presentan únicamente los resultados más significativos. Para ambos conjuntos de datos, la topología incremental obtiene resultados cercanos a los alcanzados por la topología fija (final) comenzando con una red inicial de dos neuronas en su capa oculta. Los puntos que aparecen sobre la curva de la topología incremental permiten conocer los instantes en los que el error cometido por la red supera el límite establecido para el error de test. En este momento la estructura se modifica para responder adecuadamente a las necesidades del aprendizaje.

6.2.2.2. Contextos dinámicos.

A continuación se consideran dos conjuntos artificiales diferentes. Como en entornos dinámicos los cambios pueden ser de diferentes tipos, se comprueba el rendimiento del método en varias situaciones. De este modo el primer conjunto considerado presenta cambios bruscos de contexto mientras que el segundo sufre cambios graduales continuos. La evolución temporal de la señal hace que se generen varios cambios de contextos y,

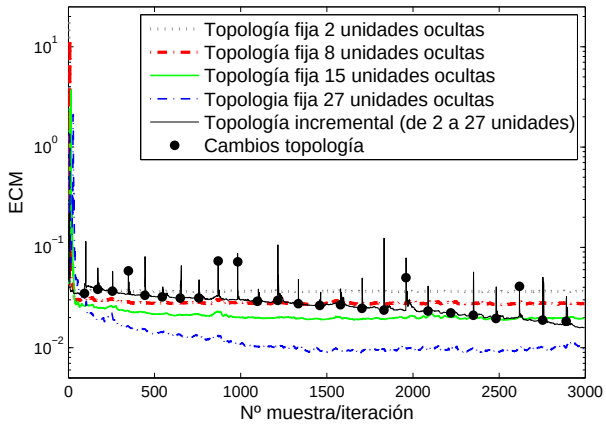


Figura 6.13: Entorno estacionario. Curvas de error de la fase de test para la serie temporal de Henon.

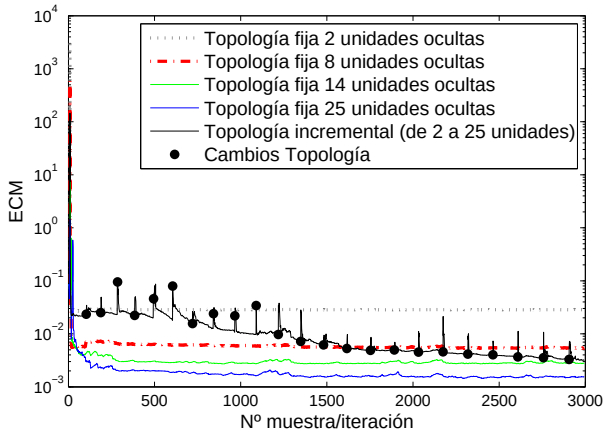


Figura 6.14: Entorno estacionario. Curvas de error de la fase de test para la serie temporal de Mackey-Glass.

por tanto, se dispone de un conjunto de test diferente para cada uno de ellos. De este modo, en las figuras que representan el error obtenido en la fase de test cada punto de la señal representa el valor medio calculado en el conjunto de test correspondiente a la muestra de entrenamiento actual.

Conjunto de datos artificial 1. El primer conjunto de datos está formado por 4 variables de entrada aleatorias, que contiene valores de una distribución normal de media 0 y desviación típica 0.1. La salida deseada se obtiene mediante una mezcla lineal de funciones no lineales aplicadas sobre dichas variables. La mezcla se modifica cada 600 muestras empleando para ello las filas de la siguiente matriz de mezcla,

$$M1 = \begin{pmatrix} 0.1 & 0.2 & 0.5 & 0.8 \\ 0.2 & 0.3 & 0.5 & 0.8 \\ 0.3 & 0.4 & 0.6 & 0.7 \\ 0.4 & 1.2 & -0.1 & 0.2 \end{pmatrix}. \tag{6.11}$$

dando lugar a cuatro contextos o estados distintos. Las funciones no lineales empleadas son sigmoide tangente hiperbólica, exponencial, seno y sigmoide logarítmica. El conjunto final contiene 2,400 muestras de entrenamiento y 4 conjuntos de test, uno para cada uno de los cambios en la mezcla lineal del conjunto de entrenamiento. La figura 6.15 contiene la señal empleada como salida deseada durante el proceso de entrenamiento.

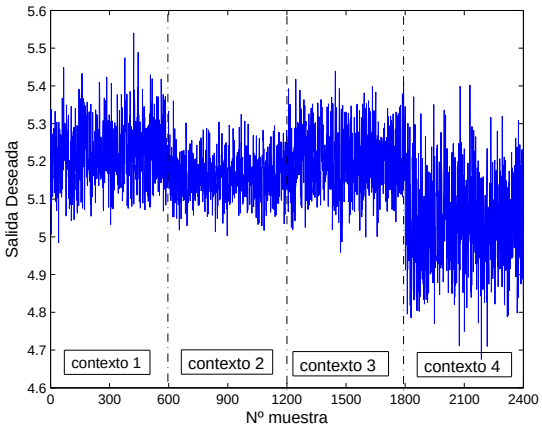


Figura 6.15: Conjunto de datos artificial 1: Ejemplo de salida deseada para conjunto de entrenamiento.

La figura 6.16 muestra las curvas del error cometido por el método para la fase de test. Por motivos de legibilidad, en esta figura sólo se presentan las curvas asociadas a las

topologías fija inicial y final. En la gráfica se puede observar como la topología fija inicial (2 unidades ocultas) comete un error elevado debido a que su estructura es insuficiente para realizar el aprendizaje de manera adecuada. En cuanto a la topología incremental se comprueba como alcanza un rendimiento medio superior al de la topología fija final, aún cuando en las primeras 600 muestras la topología fija obtiene mejores resultados al emplear desde el inicio 32 unidades ocultas, mientras que la topología incremental todavía no dispone de una estructura adecuada para el problema.

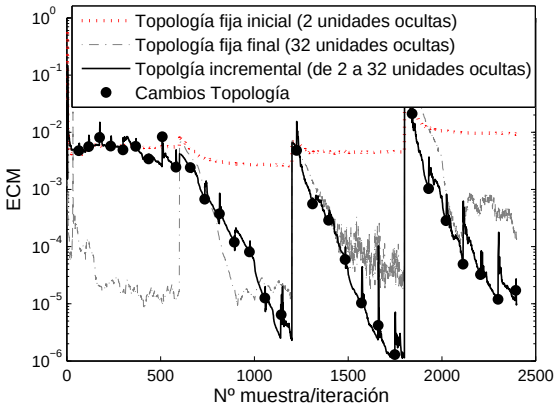


Figura 6.16: Conjunto de datos artificial 1: Curvas de error para la fase de test.

Conjunto de datos artificial 2. Se genera otro conjunto de datos artificial que presenta una evolución gradual en cada muestra de entrenamiento. El conjunto de datos está formado por 4 variables aleatorias que siguen una distribución normal de media 0 y desviación típica 0.1. La salida se obtiene por medio de una mezcla no lineal sobre las mismas fijando los coeficientes iniciales del vector de mezcla como,

$$a(0) = \left[\begin{array}{cccc} 0.5 & 0.2 & 0.7 & 0.8 \end{array} \right],$$

y evolucionando estos coeficientes en el tiempo de acuerdo a la siguiente ecuación,

$$a_j(s) = a_j(s-1) + \frac{s}{10^{4.7}} logsig(a_j(s-1)), s = 1, \ldots, S,$$

donde j es el índice que implica el componente del vector. Los datos contienen un conjunto de entrenamiento formado por 2,000 muestras y un conjunto de test para

cada una de ellas. La señal que se emplea como salida deseada se presenta en la figura 6.17.

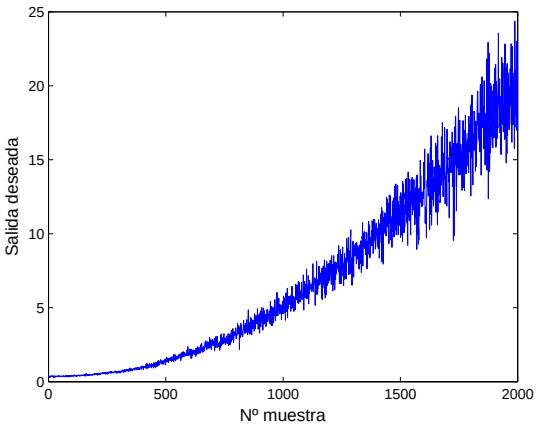


Figura 6.17: Conjunto de datos artificial 2: Ejemplo de salida deseada para conjunto de entrenamiento.

Al igual que en el ejemplo previo, en la figura 6.18 se comprueba como el método con topología incremental presenta un rendimiento superior al que alcanza el método con topología fija final. De nuevo la técnica automática permite que la estructura inicial de la red de 2 unidades evolucione en el tiempo adaptando su respuesta en función de las necesidades del proceso.

6.2.3. Discusión

En la primera parte de este capítulo se comprueba la idoneidad del método *online* presentado en el Capítulo 5 para trabajar con estructuras de red adaptativas. Los resultados obtenidos en el estudio experimental realizado verifican que es capaz de adaptar también de manera incremental la topología de la red durante el proceso de entrenamiento, sin degradar de forma significativa su rendimiento ya que permite mantener el conocimiento previo. La capa oculta puede modificar su número de unidades añadiendo

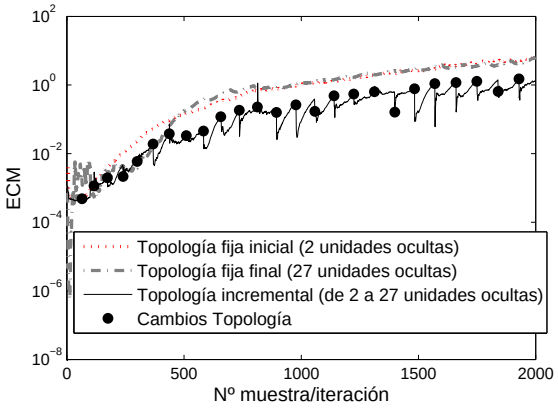


Figura 6.18: Conjunto de datos artificial 2: Curvas de error para la fase de test.

neuronas de una en una o bien en grupo obteniendo resultados satisfactorios en ambos casos, aunque lo más recomendable sería la incorporación gradual de las neuronas según las necesidades del aprendizaje. En aquellas situaciones en las que la topología inicial establecida para la red es insuficiente, el método consigue adaptarse alcanzando resultados similares a aquellos que se obtienen empleando la topología fija final desde el principio del proceso de aprendizaje. A pesar de que topologías fija e incremental obtienen un rendimiento similar, esta última aproximación implica el ahorro de tiempo y recursos computacionales debido a que por un lado, evita tener que estimar mediante prueba y error un número adecuado de unidades ocultas de la red y además, optimiza el consumo de tiempo y recursos computacionales durante el aprendizaje. Este consumo puede llegar a ser crítico especialmente cuando el proceso a modelar es complejo y necesita un tamaño importante de red para resolverlo, debido a que no se emplea una red de tamaño considerable desde el comienzo del aprendizaje.

Una vez demostrado el funcionamiento del algoritmo al añadir nuevas unidades ocultas, se incorpora un mecanismo para decidir la adaptación de la topología de la red de forma automática. De este modo en la sección 6.2 se presenta la versión final del método que incluye una técnica para controlar el crecimiento de la topología. La técnica se basa en la dimensión de Vapnik-Chervonenkis, una buena medida de la capacidad de generalización de un modelo de aprendizaje que además está relacionada con la

complejidad del mismo. Empleando esta medida como referencia es posible predecir un límite superior para el error de test cometido por el modelo. El crecimiento de la red de neuronas está supeditado en todo momento a las necesidades del proceso de aprendizaje.

A la vista de los resultados presentados para las series temporales estacionarias y los conjuntos de naturaleza dinámica, se puede decir que la técnica automática basada en la dimensión de Vapnik-Chervonenkis de una red neuronal permite obtener una estimación del tamaño adecuado de la red. Tomando como referencia los resultados alcanzados por la topología fija se comprueba cómo la aproximación incremental desarrollada obtiene un rendimiento similar, sin necesidad de estimar previamente la topología adecuada para resolver el problema. Además, el método propuesto permite optimizar los recursos computacionales disponibles ya que en ningún momento se emplea una red de tamaño mayor que el necesario para satisfacer las necesidades del aprendizaje.

Resumiendo, en este capítulo se presenta un nuevo algoritmo de aprendizaje *online* e incremental para redes de neuronas de dos capas con alimentación hacia delante. Sus principales innovaciones se comentan brevemente a continuación.

- En primer lugar, gracias a la combinación de la propiedad de aprendizaje incremental y la asignación de la importancia creciente, la red mantiene un comportamiento estable cuando el contexto es estático y al mismo tiempo es capaz de olvidar rápidamente en presencia de un cambio. Esta característica hace que el algoritmo propuesto sea un método útil para entornos que tienen que trabajar en tiempo real y presentan un comportamiento evolutivo a lo largo del tiempo. Asimismo, también es aplicable en aquellas situaciones en las que se disponga previamente de los datos realizando un entrenamiento *online*.

- Por otra parte, la capacidad de adaptar la estructura de red de manera incremental permite al método controlar y adaptar el crecimiento de la red de manera automática en función de las necesidades del aprendizaje. El rendimiento alcan-

zado por el método iguala e incluso supera los resultados obtenidos utilizando desde el inicio del aprendizaje la topología final alcanzada. Esta característica permite disponer de un método de complejidad reducida y adaptativo, apropiado para aquellas situaciones en las que se hace necesario manejar un gran número de datos de entrenamiento o incluso cuando el problema es complejo y requiere una red con un número elevado de nodos para su resolución.

De este modo, se puede decir que se propone un método capaz de manejar entornos que trabajan en tiempo real y que presentan cambios en el concepto de salida deseada a lo largo del proceso de aprendizaje. Además de adaptar sus pesos, la topología puede crecer de manera controlada adaptándose a las necesidades del proceso y evitando el problema de búsqueda de una topología óptima mediante prueba y error, y optimizando el empleo de los recursos computacionales disponibles a lo largo del proceso de aprendizaje.

Capítulo 7

Conclusiones, principales aportaciones y trabajo futuro.

A continuación se resumen las principales aportaciones y conclusiones que se obtuvieron de la investigación realizada y se apuntan algunas de las posibles líneas de trabajo futuro derivadas de esta Tesis Doctoral. En primer lugar y en cuanto a las conclusiones del trabajo se puede establecer que:

1. Se ha presentado una mejora del algoritmo de aprendizaje lineal basado en sensibilidad estadística para redes de neuronas de dos capas, conocido como SBLLM, que incluye la técnica de regularización del decaimiento de pesos para manejar situaciones en las que es posible la aparición del fenómeno de sobreajuste a los datos. Este problema puede llegar a ser crítico, especialmente cuando se dispone de un número reducido de muestras de entrenamiento o si se manejan datos distorsionados por ruido. Además, la modificación realizada incluye la estimación automática del parámetro de regularización y permite conservar las principales características del algoritmo de aprendizaje original, que se resumen brevemente a continuación,

- Los pesos de cada capa de la red se calculan resolviendo un sistema de ecuaciones lineales.
- Muestra una alta velocidad de convergencia.
- Obtiene un buen rendimiento en términos de error cometido.

2. Se han desarrollado algoritmos de aprendizaje *online* tanto para redes de una capa como de dos, que permiten su aplicación en entornos no estacionarios en los que el proceso a modelar no permanezca inalterable. Sus principales características son,

   - La capacidad de aprendizaje *online* permite su uso en entornos que tienen que trabajar en tiempo real.
   - Se adaptan a entornos no estacionarios gracias a la inclusión de un término de olvido en sus funciones de coste. Los métodos consiguen un comportamiento flexible para afrontar los diferentes tipos de cambios que pueden aparecer a lo largo del aprendizaje.
   - Son capaces de aprender de la información más reciente pero al mismo tiempo retienen el conocimiento previo relevante. Estas características permite hablar de un equilibrio estabilidad-plasticidad adecuado.
   - Presentan requisitos de memoria reducidos debido a que únicamente necesitan almacenar el patrón actual y actualizar, para cada salida de la red, las matrices $\mathbf{A}$ (de tamaño $(I+1) \times (I+1)$, donde I es la dimensión de la entrada) y los vectores $\mathbf{b}$ (de tamaño $(I+1) \times 1$) de coeficientes que contienen el conocimiento previo.
   - La complejidad del método viene determinada por la complejidad para resolver los sistemas de ecuaciones lineales, uno para cada una de las salidas de cada red o subred. De este modo se puede decir que la complejidad del método viene dada por $O(N^2)$ donde N viene dado por el número de parámetros de la red.
   - Estas características permiten un uso adecuado de los recursos disponibles. Por tanto, los métodos se podrían aplicar en situaciones *batch* con grandes

volúmenes de datos para las que el uso de otros algoritmos (Levenberg-Marquardt, Gradiente Conjugado, etc.) se hace impracticable debido a los requisitos espaciales (memoria).

3. Se ha diseñado e implementado una adaptación del algoritmo de aprendizaje *on-line* para incrementar la topología de red de manera automática en función de las necesidades del aprendizaje. Este hecho permite que la estructura de la red comience con el mínimo número de neuronas ocultas y vaya añadiendo unidades siempre y cuando la topología actual no sea suficiente para satisfacer las necesidades del proceso. A mayores, permite ahorrar tiempo y recursos espaciales, una característica importante en aquellas situaciones en las que se hace necesario manejar un gran número de datos de entrenamiento o incluso cuando el problema es complejo y requiere una red con un elevado número de nodos para la resolución del mismo.

En cuanto al posible trabajo futuro, se proponen las siguientes líneas de investigación:

- Para trabajar de manera eficiente con problemas de clasificación se hace necesario encontrar una medida de distancia más adecuada que el ECM y que permita resolver igualmente el cálculo de los pesos de forma lineal.
- Con respecto al método SBLLM, y para paliar el problema del estancamiento de las curvas de error en pocas iteraciones del aprendizaje, se plantea la investigación del uso de otras opciones tales como las sensibilidades de segundo orden.
- En cuanto a la aplicación de los métodos de aprendizaje *online* a entornos reales, se prevé su utilización en sistemas de control en los que está trabajando actualmente el laboratorio LIDIA, tales como el diagnóstico predictivo de aerogeneradores en que se trabaja con la empresa INDRA.
- Para el método con topología incremental automática, se realizarán más estudios y propuestas con el fin de mejorar la adición de unidades ocultas empleando

para ello ciertas medidas, como por ejemplo, la tendencia creciente de los errores cometidos. Además también podría plantearse una modificación del método de manera que incluyese alguna técnica de podado que permitiese la eliminación de unidades innecesarias en caso de que el aprendizaje así lo requiera.

Apéndice I

Notación y abreviaturas empleadas

Nomenclatura empleada

Notación	Significado
$\mathbf{x}$	vector de entrada a la red
$\mathbf{y}$	vector de salida de la red
$\mathbf{z}$	vector de salida de la capa oculta
$\mathbf{w}$	vector pesos de la red
$\mathbf{d}$	vector salidas deseadas
i	subíndice variables de entrada
j	subíndice variables de salida
k	subíndice variables capa oculta
s	subíndice patrones entrenamiento
I	número de entradas
J	número de salidas
K	número de unidades de la capa oculta
S	número de patrones de entrenamiento

Nomenclatura empleada

Notación	Significado
N	número de pesos y sesgos de la red neuronal
x_0	sesgos capa de entrada
z_0	sesgos capa de oculta
μ	parámetro control crecimiento función olvido
α	parámetro de regularización
ρ	paso de aprendizaje
ε	error
Superíndice $^{(1)}$	primera capa de la red neuronal
Superíndice $^{(2)}$	segunda capa de la red neuronal
Superíndice ^{T}	Traspuesto
Superíndice $^{-1}$	Inversa
Superíndice $^{'}$	Derivada Primera
Superíndice $^{''}$	Derivada Segunda
Letra minúscula normal	Variable escalar
Letra minúscula negrilla	Vector columna
Letra mayúscula negrilla	Matriz
∇	Gradiente
Δ	Incremento
∂	Derivada parcial
$f(\cdot)$	función activación no lineal neuronas ocultas
$g(\cdot)$	función activación no lineal neuronas salida
$P(\cdot)$	probabilidad
$p(\cdot)$	función de densidad de probabilidad
d_{VC}	dimensión Vapnik-Chervonenkis

Abreviaturas

RNA	-	Red de neuronas artificiales
ECM	-	Error cuadrático medio
SBLLM	-	Método de aprendizaje lineal basado en sensibilidad
GD	-	Gradiente descendente
GDX	-	Gradiente descendente con momento y paso de aprendizaje adaptativo
SGD	-	Gradiente descendente estocástico
SCG	-	Gradiente conjugado escalado
LM	-	Levenberg-Marquardt
NLMS	-	Normalized least mean square
LMS	-	Least mean square
RMS	-	Raíz cuadrada del error cuadrático medio
RPM	-	Revoluciones por minuto
KL	-	Kullback-Leiber
MDL	-	Minimum description length
VC	-	Validación cruzada
LOO	-	Leave-one-out
SVR	-	Support Vector Regresssion
OS-ELM	-	Online Sequential Extreme Machine Learning

Apéndice II

Publicaciones

- Beatriz Pérez-Sánchez, Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas. *Optimización del proceso de aprendizaje en redes de neuronas artificiales*. Actas de la XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2007). Volumen II. Doctoral Consortium. pp. 347 – 350. Salamanca, 12-16 de Noviembre de 2007. *Premio José Cuena de la Asociación Española para la Inteligencia Artificial (AEPIA) al Mejor Artículo de Doctoral Consortium.*

- Bertha Guijarro-Berdiñas, Oscar Fontenla-Romero, Beatriz Pérez-Sánchez, Amparo Alonso-Betanzos. *A Regularized Learning Method for Neural Networks Based on Sensitivity Analysis*. Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN 2008), pp. 289 –294. Bruges (Belgium), 23-25 de Abril de 2008.

- Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas. *A Supervised Learning Method for Neural Networks Based on Sensitivity Analysis with Automatic Regularization*. Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN 2009). Lecture Notes in Computer Science 5517, Part I, pp. 157 – 164. Salamanca, 10-12 de Junio de 2009.

- Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas. *An*

Incremental Learning Method for Neural Networks Based on Sensitivity Analysis. Actas de la XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2009), pp. 529–538. Sevilla, 9 - 13 de Noviembre de 2009.

- Beatriz Pérez-Sánchez, Oscar Fontenla-Romero and Bertha Guijarro-Berdiñas *An Incremental Learning Method for Neural Networks in Adaptive Environments*. Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010), pp. 815–822. Barcelona, 18-23 de Julio de 2010.

- David Martínez-Rego, Oscar Fontenla-Romero, Beatriz Pérez-Sánchez, Amparo Alonso-Betanzos. *Fault Prognosis of Mechanical Components Using On-Line Learning Neural Networks*. Proceedings of the 20th International Conference on Artificial Neural Networks (ICANN 2010). Lecture Notes in Computer Science 6352, Part I, pp. 60 – 66. Thessaloniki, Greece, 15-18 de Septiembre de 2010.

- Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, Beatriz Pérez-Sánchez, Amparo Alonso-Betanzos. *A new convex objective function for the supervised learning of single-layer neural networks*. Pattern Recognition 43 (2010) 1984-1992.

- David Martínez-Rego, Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, Amparo Alonso-Betanzos. *A robust incremental learning method for non-stationary environments*. NeuroComputing: Special Issue on Incremental Learning. Aceptado para su publicación. En prensa.

Bibliografía

[1] Federal Reserve Economic Data FRED, U.S. Department of Labor: Bureau of Labor Statistics. `http://research.stlouisfed.org/fred2`. Último acceso en 2008.

[2] Laboratorio de investigación y desarrollo en Inteligencia Artificial, Dept. Computación, Universidade da Coruña. `http://aliana.dc.fi.udc.es/www-lidia/`.

[3] StatLib. StatLib datasets archive. Carnegie Mellon University, Department of Statistics. `http://lib.stat.cmu.edu/datasets` (1999).

[4] Allen J. "Natural Language Understanding". Addison-Wesley: Reading, MA, 2nd edición (1995).

[5] Almeida L. B., Langlois T., Amaral J. D. y Plakhov A. Parameter adaptation in stochastic optimization. En Saad D., editor, "On-line Learning in Neural Networks", capítulo 6, páginas 111–134. Cambridge University Press, New York, NY, USA (1998).

[6] Alpaydin E. "Introduction to Machine Learning (Adaptive Computation and Machine Learning)". The MIT Press (2004).

[7] Anderson J. A. A memory storage model utilizing spatial correlation functions. *Biological Cybernetics* **5**(3), 113–119 (1968).

[8] Anderson J. A. A theory for the recognition of items from short memorized lists. *Psychological Review* **80**(6), 417–438 (1973).

[9] Anderson J. A. y Murphy G. L. Psychological concepts in a parallel system. *Physica* **22D**, 318–336 (1986).

[10] Ash T. Dynamic node creation in backpropagation networks. *Connection Science* **1**(4), 365–375 (1989).

[11] Auer P., Herbster M. y Warmuth M. K. Exponentially many local minima for single neurons. En Touretzky D. S., Mozer M. C. y Hasselmo M. E., editores, "Advances in Neural Information Processing Systems", tomo 8, páginas 316–322. MIT Press (1996).

[12] Aylward S. y Anderson R. An algorithm for neural network architecture generation. En "AIAA Computing in Aerospace Conference VIII" (1991).

[13] Azoff E. M. "Neural Network Time Series, Forecasting of Finnancial Markets". John Wiley & Sons, New York, NY, USA (1994).

[14] Barron A., Rissanen J. y Yu B. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory* **44**(6), 2743–2760 (1998).

[15] Basseville M. y Nikiforov I. "Detection of Abrupt Changes: Theory and Applications". Prentice-Hall Inc. (1993).

[16] Battiti R. First and second order methods for learning: Between steepest descent and Newton's method. *Neural Computation* **4**(2), 141–166 (1992).

[17] Baum E. B. y Haussler D. What size net gives valid generalization? *Neural Computation* **1**(1), 151–160 (1989).

[18] Beale E. M. L. A derivation of conjugate gradients. En Lootsma F. A., editor, "Numerical methods for nonlinear optimization", páginas 39–43. Academic Press, London (1972).

[19] Biegler-König F. y Bärmann F. A Learning Algorithm for Multilayered Neural Networks Based on Linear Least-Squares Problems. *Neural Networks* **6**(1), 127–131 (1993).

[20] Bilbro G. L., Snyder W. E., Gamier S. J. y Gault J. Mean field annealing: A formalism for constructing GNC-like algorithms. *IEEE Transactions on Neural Networks* **3**(1), 131–138 (1992).

[21] Bishop C. M. "Neural Networks for Pattern Recognition". Oxford University Press, New York (1995).

[22] Bishop C. M. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation* **7**(1), 108–116 (1995).

[23] Bojanczyk A. Complexity of solving linear systems in different models of computation. *SIAM Journal on Numerical Analysis* **21**(3), 591–603 (1984).

[24] Brady M., Raghavan R. y Slawny J. Back propagation fails to separate where perceptrons succeed. *IEEE Transactions on Circuits and Systems* **36**(5), 665–674 (1989).

[25] Budinich M. y Milotti E. Geometrical interpretation of the backpropagation algorithm for the perceptron. *Physica A: Statistical Mechanics and its Applications* **185**(1–4), 369–377 (1992).

[26] Buntine W. L. y Weigend A. S. Computing Second Derivatives in Feed-Forward Networks: A review. *IEEE Transactions on Neural Networks* **5**(3), 480–488 (1993).

[27] Carayannis G., Kalouptsidis N. y Manolakis D. Fast recursive algorithms for a class of linear equations. *IEEE Transactions on acoustics, speech, and signal processing* **30**(2), 227–239 (1982).

[28] Castillo E., Cobo A., Gutierrez J. M. y Pruneda R. E. Working with differential, functional and difference equations using functional networks. *Applied Mathematical Modelling* **23**(2), 89–107 (1999).

[29] Castillo E., Conejo A., Pedregal P., García R. y Alguacil N. Functional networks. A new neural network based methodology. *Computer-Aided Civil and Infrastructure Engineering* **15**(2), 90–106 (2000).

[30] Castillo E., Fontenla-Romero O., Alonso-Betanzos A. y Guijarro-Berdiñas B. A Global Optimum Approach for One-Layer Neural Networks. *Neural Computation* **14**(6), 1429–1449 (2002).

[31] Castillo E., Guijarro-Berdiñas B., Fontenla-Romero O. y Alonso-Bentanzos A. A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis. *Journal of Machine Learning Research* **7**, 1159–1182 (2006).

[32] Castillo E., Gutiérrez J. M. y Pruneda R. E. Sensitivity analysis in discrete bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics* **26**(7), 412–423 (1997).

[33] Cherkassky V. y Mulier F. "Learning from Data: Concepts, Theory, and Methods". Wiley-IEEE Press, New York (1998).

[34] Cichocki A. y Unbehauen R. "Neural Networks for Optimization and Signal Processing". Jonh Willey & Sons, New York, NY, USA (1993).

[35] Coetzee F. M. y Stonick V. L. On a natural homotopy between linear and nonlinear single-layer networks. *IEEE Transactions on Neural Networks* **7**(2), 307–317 (1996).

[36] Coetzee F. M. y Stonick V. L. On the uniqueness of weights in single layer perceptrons. *IEEE Transactions on Neural Networks* **7**(2), 318–325 (1996).

[37] Cooper L. A possible organization of animal memory and learning. En "Proceedings of the Nobel Symposium on Collective Properties of Physical Systems", tomo 24 (1973).

[38] Cooper L., Lieberman F. y Oja E. A theory for the acquisition and loss of neuron specificity in visual cortex. *Biological Cybernetics* **3**, 9–28 (1979).

[39] Drago G. P. y Ridella S. Statistically controlled activation weight initialization. *IEEE Transactions on Neural Networks* **3**, 899–905 (1992).

[40] Dries A. y Rückert U. Adaptive concept drift detection. *Statistical Analysis and Data Mining* **2**, 311–317 (2009).

[41] Duda R. O. y abd D. G. Stork P. E. H. "Pattern Classification". Wiley-Interscience (2001).

[42] Elwell R. y Polikar R. Incremental learning in nonstationary environments with controlled forgetting. En "Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN 2009)", páginas 771–778, Piscataway, NJ, USA (2009). IEEE Press.

[43] Erdogmus D., Fontenla-Romero O., Principe J. C., Alonso-Betanzos A. y Castillo E. Linear-Least-Squares Initialization of Multilayer Perceptrons Through Backpropagation of the Desired Response. *IEEE Transactions on Neural Networks* **16**(2), 325–337 (2005).

[44] Esposito F., Ferilli S., Fanizzi N., Basile T. M. A. y Mauro M. D. Incremental learning and concept drift in INTHELEX. *Intelligent Data Analysis* **8**(3), 213–237 (2004).

[45] Fiesler E. Comparative Bibliography of Ontogenic Neural Networks. En "Proccedings of the International Conference on Artificial Neural Networks (ICANN 1994)", páginas 793–796 (1994).

[46] Fisher R. "Statistical methods and scientific inference". Edinburgh: Oliver & Boyd (1959).

[47] Fontenla-Romero O., Erdogmus D., Principe J. C., Alonso-Betanzos A. y Castillo E. Linear Least-Squares Based Methods for Neural Networks Learning. En "Proceedings of the International Conference on Artificial Neural Networks (ICANN 2003), Lecture Notes in Computer Science", tomo 2714, páginas 84–91 (2003).

[48] Fontenla-Romero O., Guijarro-Berdiñas B., Pérez-Sánchez B. y Alonso-Betanzos A. A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition* **43**(5), 1984–1992 (2010).

[49] Fukuoka Y., Matsuki H., Minamitani H. y Ishida A. A modified backpropagation method to avoid false local minima. *Neural Networks* **11**(6), 1059–1072 (1998).

[50] Geman S. y Geman D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (6), 721–741 (November 1984).

[51] Gibbons J. D. "Nonparametric Statistical Inference, 2nd Edition". New York: Marcel Dekker (1985).

[52] Girosi F., Jones M. y Poggio T. Regularization theory and neural networks architectures. *Neural Computation* **7**, 219–269 (1995).

[53] Goodwin G. C. y Sin K. S. "Adaptive Filtering, Prediction, and Control". Prentice-Hall, Englewood Cliffs, NJ (1984).

[54] Gori M. y Tesi A. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(1), 76–85 (1992).

[55] Guijarro-Berdiñas B., Fontenla-Romero O., Pérez-Sánchez B. y Alonso-Betanzos A. A Regularized Learning Method for Neural Networks Based on Sensitivity Analysis. En "Proceedings of 16th European Symposium on Artificial Neural Networks (ESANN 2008)", páginas 289–294 (2008).

[56] Guo P., Lyu M. R. y Chen C. L. P. Regularization Parameter Estimation for Feedforward Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **33**(1), 35–44 (2003).

[57] Hagan M. T. y Menhaj M. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks* **5**(6), 989–993 (1994).

[58] Hansen L. K., Rasmussen C. E., Svarer C. y Larsen J. Adaptive Regularization. En J. Vlontzos J.-N. H. y Wilson E., editores, "Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV", páginas 78–87. IEEE Press (1994).

[59] Haykin S. "Neural Networks: A Comprehensive Foundation". Prentice-Hall, 2 edición (1999).

[60] Haykin S. y Widrow B. "Least-Mean-Square Adaptive Filters". Wiley-Interscience, Hoboken, NJ (2003).

[61] Hebb D. "The Organization of Behavior: A neuropsychological theory". Willey (1949).

[62] Hinton G., Ackley D. y Sejnowski T. Boltzmann machines: Constraint satisfaction networks that learn. Informe técnico, CMU-CS-84-119, Carnegie-Mellon University, Department of Computer Science (1984).

[63] Hirose Y., Yamashita K. y Hijiya S. Backpropagation algorithm which varies the number of hidden units. *Neural Networks* **4**(1), 61–66 (1991).

[64] Hollander M. y Wolfe D. A. "Nonparametric Statistical Methods". John Wiley & Sons, New York (1973).

[65] Honovar V. y Uhr L. Generative learning structures for generalized connectionist networks. *Information Sciences* **70**(1-2), 75–108 (1993).

[66] Hopfield J. Neural networks and physical systems with emergent collective computational abilities. En "Proceedings of the National Academy of Sciences of the United States of America", tomo 79, páginas 2554–2558 (1982).

[67] Hopfield J. Neurons with graded response have collective computational properties like those of two-state neurons. En "Proceedings of the National Academy of Sciences", tomo 81, páginas 3088–3092 (1984).

[68] Huang G.-B., Zhu Q.-Y. y Siew C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **70**(1-3), 489–501 (2006).

[69] Hush D. R. y Horne B. G. Progress in supervised neural networks; what's new since lippmann? *IEEE Signal Processing Magazine* **10**, 8–39 (1993).

[70] Hush D. R. y Salas J. M. Improving the learning rate of back-propagation with the gradient reuse algorithm. *Proceedings of the IEEE Conference of Neural Networks* **1**, 441–447 (1988).

[71] Hyndman R. n. Time series data library. `http://www.robhyndman.info/TSDL`. Último acceso en 2008.

[72] Ihm B. C. y Park D. J. Acceleration of learning speed in neural networks by reducing weight oscillations. *Proceedings of the International Joint Conference on Neural Networks* **3**, 1729–1732 (1999).

[73] Jacobs R. A. Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks* **1**(4), 295–308 (1988).

[74] Kirkpatrick S., Gelatt C. D. y Vecchi M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).

[75] Kohonen T. Correlation Matrix Memories. *IEEE Transactions on Computers* **C-21**(4), 353–359 (1972).

[76] Kohonen T. An adaptative associative memory principle. *IEEE Transactions on Computers* **23**(4) (1974).

[77] Kolen J. F. y Pollack J. B. Back propagation is sensitive to initial conditions. En Lippmann R. P., Moody J. E. y Touretzky D. S., editores, "Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS 1991)", tomo 3, páginas 860–867, San Francisco, CA, USA (1991). Morgan Kaufmann Publishers Inc.

[78] Krzyzak A., Dai W. y Suen C. Classification of large set of handwritten characters using modified back propagation model. En "Proceedings of the International Joint Conference on Neural Networks (IJCNN 1990)", tomo 3, páginas 225–232 (1990).

[79] Kubat M., Gamma J. y Utgoff P. Incremental learning and concept drift, Editor's introduction. *Intelligent Data Analysis* **8**(3), 211–212 (2004).

[80] Kullback S. "Information Theory and Statistics". New York: Willey (1959).

[81] Kwok T.-Y. y Yeung D.-Y. Constructive Feedforward Neural Networks for Regression Problems: A Survey. Informe técnico HKUST-CS95-43, Department of Computer Science. The Hong Kong University of Science & Technology (1995).

[82] Kwok T.-Y. y Yeung D.-Y. Constructive Algorithms for Structure Learning in FeedForward Neural Networks for Regression Problems. *IEEE Transactions on Neural Networks* **8**(3), 630–645 (1997).

[83] Lai C.-A. NLMS algorithm with decreasing step size for adaptive IIR filters. *Signal Processing* **82**(10), 1305–1316 (2002).

[84] LeCun Y., Bottou L., Orr G. y Müller K.-R. Efficient BackProp. En Orr G. B. y Müller K.-R., editores, "Neural Networks: Tricks of the trade", número 1524, páginas 9–50. Springer-Verlag (1998).

[85] LeCun Y., Kanter I. y Solla S. A. Second order properties of error surfaces: Learning time and generalization. En Lippmann R., Moody J. y Touretzky D., editores, "Proceedings of the 1990 conference on Advances in neural information processing systems (NIPS)", tomo 3, páginas 918–924, San Francisco, CA, USA (1990). Morgan Kaufmann Publishers Inc.

[86] Lee B. W. y Sheu B. J. Paralleled hardware annealing for optimal solutions on electronic neural networks. *IEEE Transactions on Neural Networks* **4**(4), 588–599 (1993).

[87] Lee J., Qiu H., Yu G., Lin J. y Services R. T. Bearing Data Set, NASA Ames Prognostics Data Repository, http://ti.arc.nasa.gov/project/prognostic-data-repository. Informe técnico, IMS, Univ. of Cincinnati (2007).

[88] Lee Y., Oh S. H. y Kim M. W. An analysis of premature saturation in back propagation learning. *Neural Networks* **6**(5), 719–728 (1993).

[89] Levenberg K. A method for the solution of certain non-linear problems in least squares. *Quaterly Journal of Applied Mathematics* **2**(2), 164–168 (1944).

[90] Liang N.-Y. y Huang G.-B. A fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Transactions on Neural Networks* **17**(6), 1411–1423 (2006).

[91] Lorenz E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* **20**, 130–141 (1963).

[92] Ma J., Theiler J. y Perkins S. Accurate On-line Support Vector Regression. *Neural Computation* **15**(11), 2686–2703 (2003).

[93] Marquardt D. W. An algorithm for least-squares estimation of non-linear parameters. *SIAM Journal of the Society of Industrial and Applied Mathematics* **11**(2), 431–441 (1963).

[94] Martínez-Rego D., Pérez-Sánchez B., Fontenla-Romero O. y Alonso-Betanzos A. A robust incremental learning method for non-stationary environments. *Aceptado para su publicación en Special Issue: Incremental Learning de Neurocomputing* (2010). En prensa.

[95] Massey F. J. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association* **46**(253), 68–78 (1951).

[96] Masters T. "Practical Neural Networks recipies in C++". Academic Press Professional, Inc., San Diego, CA, USA (1993).

[97] McCulloch J. y Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **7**, 115–133 (1943).

[98] Miller W. T., Sutton R. S. y Werbos P. J. "Neural Networks for Control". Cambridge, MA, MIT Press (1990).

[99] Minsky M. y Papert S. "Perceptrons: An introduction to computational geometry". The MIT Press (1969).

[100] Moller M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks* **6**, 525–533 (1993).

[101] Murata N. Network Information Criterion-Determining the number of hidden units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks* **5**(6), 865–872 (1994).

[102] Myers C. E. "Delay Learning in Artifical Neural Networks". Chapman & Hall (1992).

[103] Nagumo J. y Noda A. A learning method for system identification. *IEEE Transactions on Automatic Control* **12**(3), 282–287 (1967).

[104] Nguyen D. y Widrow B. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. *Proceedings of the International Joint Conference on Neural Networks* **3**, 21–26 (1990).

[105] Ooyen A. V. y Nienhuis B. Improving the convergence of the backpropagation algorithm. *Neural Networks* **5**(3), 465–471 (1992).

[106] Orr G. B. y Leen T. K. Using Curvature Information for Fast Stochastic Search. En Jordan M., Mozer M. y Petsche T., editores, "In Advances in Neural Information Processing Systems (NIPS 1996)", tomo 9, páginas 606–612, Cambridge (1996). MIT Press.

[107] Pao Y. H. "Adaptive Pattern Recognition and Neural Networks". Addison-Wesley Publishing Company, Inc., Reading, MA, (1989).

[108] Parker D. B. Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning. *Proceedings of the IEEE Conference on Neural Networks* **2**, 593–600 (1987).

[109] Parrella F. "Online Support Vector Regression". Tesis Doctoral, Department of Information Science, University of Genova (June 2007).

[110] Parthiban A., Madhavan J., Savitha D., V.Lakshmi y Kumar L. Performance analysis of SIGN and NLMS algorithms for CPI radar in stationary and nonstationary environment. En "9th National Conference on Communications (NCC2003)", páginas 110–114 (2003).

[111] Penrose R. A generalized inverse for matrices. En "Proceedings of the Cambridge Philosophical Society", tomo 51, páginas 406–413 (1955).

[112] Peterson C. y Anderson J. R. A mean field theory learning algorithm for neural networks. *Complex Systems* **1**, 995–1019 (1987).

[113] Pethel S., Bowden C. y Scalora M. Characterization of Optical Instabilities and Chaos Using MLP Training Algorithms. *SPIE Chaos Opt.* **2039**, 129–140 (1993).

[114] Plaut D., Nowlan S. y Hinton G. E. Experiments on learning by back propagation. Informe técnico CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University (1986).

[115] Pérez-Sánchez B., Fontenla-Romero O. y Guijarro-Berdiñas B. An Incremental Learning Method for Neural Networks Based on Sensitivity Analysis. En "Actas de la XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2009)", páginas 529–538 (2009).

[116] Pérez-Sánchez B., Fontenla-Romero O. y Guijarro-Berdiñas B. A Supervised Learning Method for Neural Networks Based on Sensitivity Analysis with Automatic Regularization. En Cabestany J., Sandoval F. y Corchado A. P. J. M., editores, "Lecture Notes in Computer Science. Bio-Inspired Systems: Computational and Ambient Intelligence", tomo 5517, páginas 157–164. Springer (2009).

[117] Pérez-Sánchez B., Fontenla-Romero O. y Guijarro-Berdiñas B. An Incremental Learning Method for Neural Networks in Adaptive Environments. En "Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)", páginas 815–822 (2010).

[118] Reed R. Pruning Algorithms: A Survey. *IEEE Transactions on Neural Networks* **4**, 740–747 (1993).

[119] Rigler A. K., Irvine J. M. y Vogl T. P. Rescaling of variables in back propagation learning. *Neural Networks* **4**(2), 225–229 (1991).

[120] Ripley B. D. "Pattern Recognition and Neural Networks". Cambridge University Press: Cambridge (1996).

[121] Rissanen J. Modeling By Shortest Data Description. *Automatica* **14**, 465–471 (1978).

[122] Rosenblatt F. The perceptron: A perceiving and recognicing automation. Informe técnico, 85-460-1, Cornell Aeronautical Laboratory (1957).

[123] Rosenblatt F. The perceptron: A theory of statistical separability in cognitive systems. Informe técnico, VG-1196-G-1, Cornell Aeronautical Laboratory (1958).

[124] Rumelhart D. E., Hinton G. E. y Williams R. J. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", tomo 1, capítulo Learning internal representations by error propagation, páginas 318–362. The MIT Press, Cambridge, MA (1986).

[125] Rumelhart D. E., Hinton G. E. y Willian R. J. Learning Representations of Back-Propagation Errors. *Nature* **323**, 533–536 (1986).

[126] Schraudolph N. N. Fast Curvature Matrix-Vector Products. En Dorffner G., Bischof H. y Hornik K., editores, "Proceedings of the International Conference on Artificial Neural Networks (ICANN 2001) - Lecture Notes in Computer Science", London, UK (2001). Springer-Verlag.

[127] Sejnowski T. Higher-order boltzmann machines. En "Conference Proccedings, Neural Networks for Computing", páginas 398–403 (1986).

[128] Skrzypek J. y Karplus W. Special issue-neural networks in vision and pattern recognition. *Pattern Recognition and Artificial Intelligence* **6**(1), 1–208 (1996).

[129] Sontag E. y Sussmann H. J. Backpropagation can five rise to spurious local minima even for networks without hidden layers. *Complex Systems* **3**(1), 91–106 (1989).

[130] Street N. W. y Kim Y. A streaming ensemble algorithm (SEA) for large-scale classification. En "Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", páginas 377–382 (2001).

[131] Styblinski M. A. y Tang T. S. Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. *Neural Networks* **3**(4), 467–483 (1990).

[132] Suykens J. A. K. y Vandewalle J. "Nonlinear Modeling: advanced black-box techniques". Kluwer Academic Publishers Boston (1998).

[133] Tikhonov A. N. y Arsenin V. Y. "Solution of Ill-Possed Problems". V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York (1977).

[134] Tollenaere T. SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties. *Neural Networks* **3**(5), 561–573 (1990).

[135] Tsymbal A. The problem of concept drift: definitions and related work. Informe técnico TCD-CS-2004-15, Computer Science Department, Trintity College Dublin, Ireland (2004).

[136] Utans J. y Moody J. E. Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction. En "Proceedings on the First International Conference on Artificial Intelligence Applications on Wall Street", páginas 35–41. IEEE Computer Society Press (1991).

[137] Vapnik V. "Statistical Learning Theory". John Wiley & Sons, Inc. New York (1998).

[138] Viñuela P. I. y León I. M. G. "Redes de Neuronas Artificiales. Un Enfoque Práctico." Pearson Prentice Hall (2004).

[139] Vogl T. P., Mangis J. K., Rigler A. K., Zink W. T. y Alkon D. L. Accelerating the Convergence of Back-Propagation Method. *Biological Cybernetics* **59**(4), 257–263 (1988).

[140] Wahba G., Gu C. y Wang Y. Soft Classification, a.k.a. Risk Estimation, via Penalized Log Likelihood and Smoothing Spline Analysis of Variance. En "The Mathematics of Generalization - The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning", páginas 331–359. Addison-Wesley (1993).

[141] Wahba G. y Wold S. A completely automatic french curve: Fitting splines by cross validation. *Communications in Statistical. Theory and methods* **4**(1), 1–17 (1975).

[142] Wan E. A. Time series data. Department of Computer Science and Electrical Engineering. Oregon Health & Science University. `http://www.cse.ogi.edu/`. Último acceso en 2008.

[143] Wang H., Fan W. y Han J. Mining concept-drifting data streams using ensemble classifiers. En "Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)", páginas 226–235, New York, NY, USA (2003).

[144] Webb G. I., Pazzani M. J. y Billsus D. Machine learning for user modelling. *User Modelling and User Adapted Interaction* **11**, 19–29 (2001).

[145] Weigend A., Rumelhart D. y Huberman B. Back-propagation, weight-elimination and time series prediction. En T. Sejnowski G. H. y D.Touretzky, editores, "Proceedings of the 1990 Connectionist Models Summer School". Morgan Kaufmann (1990).

[146] Weir M. K. A method for self-determination of adaptive learning rates in back propagation. *Neural Networks* **4**(3), 371–379 (1991).

[147] Widmer G. Learning in the presence of concept drift and hidden contexts. *Machine Learning* **23**(1), 69–101 (1996).

[148] Widrow B. Adaptive sampled-data systems — a statistical theory of adaptation. En "IRE WESCON Convention Record", tomo 4, páginas 74–85 (1959).

[149] Widrow B. An adaptive adaline neuron using chemical memistors. Informe técnico, 1553-2, Stanford Electronics Laboratory (1960).

[150] Widrow B. y Hoff M. E. Adaptive Switching Circuits. En "IRE WESCON Convention Record", tomo 4, páginas 96–104 (1960).

[151] Widrow B. y Lehr M. A. Thirty years of adaptive neural networks. En "Proceedings of the IEEE", tomo 78, páginas 1415–1442 (1990).

[152] Xu L., Klasa S. y Yuille A. Recent advances on techniques of static feedforward networks with supervised learning. *International Journal of Neural Systems* **3**(3), 253–290 (1992).

[153] Yam J. Y. F., Chow T. W. S. y Leung C. T. A new method in determining the initial weights of feedforward neural networks for training enhancement. *Neurocomputing* **16**(1), 23–32 (1997).

[154] Yao X. Evolving Artificial Neural Networks. En "Proceedings of the IEEE", tomo 87, páginas 1423–1447 (1999).