UNIVERSIDAD DE GRANADA

Departamento de Arquitectura y Tecnología de Computadores

ESTUDIO CUANTITATIVO DE LA TOLERANCIA A FALLOS, LA CAPACIDAD DE GENERALIZACIÓN Y NUEVOS ALGORITMOS DE APRENDIZAJE PARA EL PERCEPTRÓN MULTICAPA

TESIS DOCTORAL

José Luis Bernier Villamor

1998

Editor: Editorial de la Universidad de Granada
Autor: José Luis Bernier Villamor
D.L.: Gr. 1166 - 2007
ISBN: 978-84-338-4338-8

ESTUDIO CUANTITATIVO DE LA TOLERANCIA A FALLOS, LA CAPACIDAD DE GENERALIZACIÓN Y NUEVOS ALGORITMOS DE APRENDIZAJE PARA EL PERCEPTRÓN MULTICAPA

José Luis Bernier Villamor

TESIS DOCTORAL

DIRECTORES:

Julio Ortega Lopera y Alberto Prieto Espinosa

1998

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES.
UNIVERSIDAD DE GRANADA

D. Julio Ortega Lopera, Profesor Titular de Universidad, **y D. Alberto Prieto Espinosa**, Catedrático de Universidad, del Departamento de Arquitectura y Tecnología de Computadores

CERTIFICAN

Que la memoria titulada: "Estudio cuantitativo de la tolerancia a fallos, capacidad de generalización y nuevos algoritmos de aprendizaje para el perceptrón multicapa" ha sido realizada por **D. José Luis Bernier Villamor** bajo nuestra dirección, en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor en Ciencias Físicas.

Granada, a 27 de Noviembre de 1998

Fdo. Julio Ortega Lopera
Director de la Tesis

Fdo. Alberto Prieto Espinosa
Director de la Tesis

ESTUDIO CUANTITATIVO DE LA TOLERANCIA A FALLOS, LA CAPACIDAD DE GENERALIZACIÓN Y NUEVOS ALGORITMOS DE APRENDIZAJE PARA EL PERCEPTRÓN MULTICAPA

José Luis Bernier Villamor

Fdo. José Luis Bernier Villamor

A todos los que quiero

Abreviaturas y símbolos utilizados en la memoria

ARC	Algoritmo de retropropagación clásico
ARR	Algoritmo de retropropagación robusto
ARRN	Algoritmo de retropropagación robusto frente a ruido en las entradas
ARRU	Algoritmo de retropropagación con ruido en los pesos
ARRW	Algoritmo de retropropagación robusto frente a desviaciones en los pesos
EC	Error de clasificación
ECM	Error cuadrático medio
MLP	Perceptrón multicapa
RNA	Red neuronal artificial
SCM	Sensibilidad cuadrática media
SCMN	Sensibilidad cuadrática media a desviaciones en las entradas
SCMW	Sensibilidad cuadrática media a desviaciones en los pesos
SSM	Sensibilidad estadística media
ε'	Error cuadrático medio sujeto a perturbaciones
$\varepsilon(p)$	Error cuadrático proporcionado para el patrón de entrada p
ε_0	Error cuadrático medio nominal
σ	Desviación típica de las perturbaciones
$E[\cdot]$	Valor esperado de $[\cdot]$
f_i^m	Función de activación (salida) de la neurona i perteneciente a la capa m
m	Indice del número de capa
M	Número de capas
N_0	Número de entradas a la red
N_m	Número de neuronas de la capa m
N_M	Número de neuronas de la capa de salida
N_p	Número de patrones de entrada
p	Patrón de entrada
S_c	Sensibilidad cuadrática
S_i^m	Sensibilidad estadística de la neurona i perteneciente a la capa m
S_i^M	Sensibilidad estadística de la neurona de salida i
$var(\cdot)$	Varianza de $(\cdot)$
w_{ij}^m	Peso que conecta a la neurona i de la capa m con la neurona j de la capa m-1
y_i^m	Salida proporcionada por la neurona i de la capa m
z_i^m	Suma ponderada de las entradas a la neurona i de la capa m
∂f_i^m	Derivada de f_i^m respecto de z_i^m
α	Factor de aprendizaje
β	Factor de momento
γ	Factor de robustez
K	Constante del factor de robustez
∇	Gradiente
$(\nabla\varepsilon)_{ij}^m$	Gradiente del error cuadrático respecto del peso w_{ij}^m
$(\nabla S)_{ij}^m$	Gradiente de la sensibilidad estadística media respecto del peso w_{ij}^m
$(\nabla S_c)_{ij}^m$	Gradiente de la sensibilidad cuadrática respecto del peso w_{ij}^m

Prólogo

Las redes neuronales artificiales (RNAs) constituyen una alternativa a la computación convencional basada en el uso de algoritmos específicos programados para resolver un problema en cuestión. Existen tareas que, a pesar de ser resueltas de forma relativamente sencilla por el cerebro humano, son difíciles o imposibles de programar en un ordenador convencional. De esta manera, la computación neuronal inspirada en modelos simplificados de la biología (neuronas y sistemas neuronales naturales) trata de realizar dichas tareas.

Las RNAs constan de un elevado número de procesadores simples (neuronas artificiales) con un alto nivel de conexión entre ellos. Por sí misma, cada neurona artificial realiza una tarea bastante sencilla a partir de las entradas proporcionadas por las neuronas a las que está conectada, así como de las entradas externas que recibe. Sin embargo, con relaciones adecuadas o "pesos" en las conexiones neuronales, la red en conjunto es capaz de resolver problemas complicados.

Lo que caracteriza a las RNAs frente a otros paradigmas computacionales es el aprendizaje. Las RNAs no necesitan ser programadas en sentido estricto, sino que son sometidas a un proceso de entrenamiento mediante el cual modifican de manera automática sus parámetros (normalmente sus pesos) de forma que "aprenden" la función que deben implementar. Al no ser necesaria la programación de las mismas, permiten dar solución a problemas para los que no se conoce un algoritmo eficaz. Hay un paso directo desde el nivel de conocimiento al nivel físico, sin ser necesario un nivel simbólico o de programación intermedio.

Los tipos de problemas donde las RNAs se han mostrado eficaces son variados. Frecuentemente se usan para el reconocimiento de patrones, reconocimiento del habla, extracción de características, control, predicción, visión artificial, diagnóstico, etc.

Así pues, el proceso de aprendizaje tiene por objeto calcular los valores de los pesos que permiten resolver una determinada tarea, una vez fijada una estructura para la red neuronal. Para ello se utiliza un conjunto de vectores de entrada llamados "patrones de entrenamiento". Una vez entrenada, la red puede efectuar la función para la que ha sido adiestrada, aunque los vectores de entrada sean en principio desconocidos para ella. En este caso se dice que la red "generaliza", es decir, no sólo aprende el comportamiento para los vectores de entrenamiento sino también para otros nuevos.

Es frecuente que el entrenamiento sea simulado por medio de un ordenador convencional, debiéndose "cargar" posteriormente los valores obtenidos para los pesos en una implementación física concreta. Así se consigue una reducción en la complejidad del circuito ya que se prescinde de la lógica necesaria para el aprendizaje. Sin embargo, las diferencias entre la precisión usada para la implementación y la usada por el algoritmo de entrenamiento pueden provocar una degradación importante de las prestaciones de la red ya que la red de la implementación física no presentaría los mismos valores de los pesos que la red simulada.

A menudo, por semejanza con las redes neuronales biológicas, a las RNAs se les atribuye el ser inherentemente tolerantes a fallos. No obstante, se ha probado que este hecho no es totalmente cierto. Es decir, las RNAs no son estrictamente tolerantes a fallos en sí mismas, pero sí pueden presentar un cierto nivel de tolerancia distinto según sea la estructura elegida y los valores de los pesos. Pueden tenerse dos configuraciones distintas de pesos para una estructura dada con idénticas prestaciones en cuanto a error de salida, pero con distinto grado de robustez frente a perturbaciones en los mismos. Este mismo hecho se puede observar respecto de la capacidad de generalización.

Uno de los modelos más ampliamente conocidos es el llamado perceptrón multicapa (MLP). Como su nombre indica, un MLP consta de varias capas, donde cada una de ellas está constituida por un número determinado de neuronas. La salida de las neuronas de cada capa se encuentra conectada a las entradas de las neuronas de la siguiente capa. El perceptrón multicapa se suele entrenar mediante aprendizaje supervisado, utilizando un conjunto de entradas para las que se conoce la salida deseada, y buscando minimizar la diferencia entre la salida deseada y la proporcionada por el MLP (error de salida).

El trabajo presentado en esta memoria[1] constituye una contribución al estudio de la tolerancia a desviaciones y de la capacidad de generalización en las redes neuronales artificiales, en concreto de los perceptrones multicapa. Partiendo de una medida propuesta en la bibliografía para evaluar de manera cuantitativa la robustez proporcionada por un conjunto de pesos frente a desviaciones en sus valores y a ruido en las entradas de la red, particularizamos ésta para cada nodo de la red. A partir de estas medidas hemos obtenido otra medida más adecuada que no sólo permite seleccionar entre varias configuraciones de pesos posibles, sino también predecir cuantitativamente la degradación de las prestaciones de aprendizaje para una desviación en los pesos (por ejemplo, para un margen de tolerancia analógica dado en una implementación física) o un determinado nivel de ruido en las entradas de la red. Con ayuda de las medidas obtenidas hemos desarrollado distintos algoritmos de aprendizaje que optimizan conjuntamente el error de

[1]parcialmente financiado por el proyecto TIC97-1149.

salida y la tolerancia a fallos, inmunidad al ruido o la capacidad de generalización de la red. Así, es posible obtener configuraciones de pesos estables frente a desviaciones en sus valores y a perturbaciones en las entradas, manteniendo las mismas prestaciones en cuanto a aprendizaje que proporcionan los algoritmos usados normalmente.

Este trabajo se distribuye en los capítulos que a continuación se indican:

Capítulo 1: se realiza una introducción general sobre las características de las redes neuronales artificiales (modelos, tipos de aprendizaje, algoritmos de entrenamiento, etc) y se revisa el modelo del MLP para introducir cuestiones de notación y revisar la bibliografía al respecto, particularmente sobre el tema de la tolerancia a fallos y las técnicas de regularización.

Capítulo 2: se utiliza la sensibilidad estadística como medida cuantitativa para medir el grado de robustez de un perceptrón multicapa. La sensibilidad estadística mide cuánto varía la salida del MLP al variar los valores de sus pesos. Se obtiene la expresión de esta sensibilidad estadística para cualquier nodo del MLP usando los dos modelos usuales de desviación: el aditivo y el multiplicativo. También se muestran resultados de simulación que corroboran que a menor sensibilidad estadística mayor es la estabilidad del MLP frente a perturbaciones en los valores de los pesos que lo constituyen.

Capítulo 3: utilizando la expresión obtenida para la sensibilidad estadística, se incluye ésta como nuevo término en la regla usada por el algoritmo de aprendizaje. De esta forma durante el entrenamiento del MLP, los pesos se ajustan para minimizar tanto el error de salida como la sensibilidad estadística a desviaciones en los pesos, obteniendo así configuraciones para los mismos que, manteniendo las prestaciones en cuanto a aprendizaje, son sin embargo más tolerantes a fallos. A este nuevo algoritmo se le ha denominado "Algoritmo de Retropropagación Robusto" o ARR. Se incluyen resultados que demuestran la eficacia del algoritmo.

Capítulo 4: se encuentra una relación explícita entre las perturbaciones del error cuadrático medio cuando los pesos sufren desviaciones y la sensibilidad estadística de las neuronas de salida. De esta forma se obtiene una nueva medida, la "Sensibilidad Cuadrática Media" (SCM). La SCM se calcula a partir de la sensibilidad estadística de las neuronas de salida y constituye una medida más apropiada para evaluar el grado de robustez de un MLP por estar relacionada directamente con el error de salida. Así, para una perturbación en los pesos de magnitud conocida es posible predecir de manera cuantitativa la degradación del error cuadrático medio sufrida por el MLP. Se diseña un

algoritmo similar a ARR, llamado ARRW, en el que se utiliza la SCM como magnitud a minimizar conjuntamente con el error cuadrático medio durante el entrenamiento del MLP. Se realizan diversos experimentos que muestran la validez de la expresión obtenida para la SCM, así como la eficacia del algoritmo ARRW.

Capítulo 5: se realiza un estudio de la influencia del ruido en las entradas del MLP. Se particularizan las expresiones de la sensibilidad estadística así como de la sensibilidad cuadrática media para perturbaciones en las entradas de la red. Para un cierto nivel de ruido es posible conocer de antemano cuánto se degradan las prestaciones de aprendizaje del MLP. También se muestra la relación existente entre la SCM a ruido en las entradas (SCMN) y la capacidad de generalización. Se desarrolla un nuevo algoritmo de entrenamiento llamado ARRN, el cual trata de minimizar la SCMN, usando la misma filosofía que los algoritmos previamente desarrollados (ARR y ARRW). De los resultados obtenidos se concluye que con ARRN se obtienen configuraciones de pesos con las que el MLP presenta mayor inmunidad a perturbaciones en sus entradas y mayor capacidad de generalización.

Capítulo 6: se muestra una relación explícita entre la SCM a desviaciones en los pesos (SCMW) y otra medida de la tolerancia a fallos llamada "saliencia" propuesta en la bibliografía, y la conveniencia de usar la primera. Se compara el procedimiento ARRW con otro procedimiento propuesto en la literatura para aumentar la tolerancia a fallos y ampliamente conocido. Los resultados obtenidos prueban que ARRW ofrece prestaciones claramente superiores. También se analizan problemas reales, pertenecientes al conjunto de pruebas estándar PROBEN1.

Capítulo 7: recoge las conclusiones y resume las aportaciones principales del presente trabajo, así como las líneas a seguir en investigaciones futuras.

Índice

Capítulo 1

Introducción: Notación y Conceptos Previos

Las redes neuronales artificiales surgen como alternativa al procedimiento de cómputo usual en las arquitecturas de computador de Von Neumann en las que un programa almacenado determina el flujo de instrucciones y datos en el ordenador, y la resolución de un problema implica disponer de un algoritmo para el mismo. Existen tareas relativamente sencillas para el cerebro humano que, sin embargo, resultan difíciles de describir usando las técnicas clásicas de programación de ordenadores. De esta manera, las redes neuronales artificiales tratan de modelar la estructura y funcionamiento de las redes neuronales biológicas con objeto de dar solución a este tipo de problemas sustituyendo la programación propiamente dicha por un proceso de aprendizaje. Una de las características de las redes neuronales biológicas es su tolerancia a fallos. Si una red neuronal artificial tuviese también esta propiedad, resultaría altamente interesante desde el punto de vista de su implementación física. En este primer capítulo se realiza una revisión de los principales conceptos relacionados con objeto de poner en contexto el trabajo presentado, así como introducir la notación básica seguida en el resto de los capítulos.

1.1. INTRODUCCIÓN

El desarrollo de los computadores ha estado motivado por la búsqueda de sistemas para procesamiento automático de la información. Según Allen Newell [MIR95], en el paradigma computacional se pueden considerar tres niveles conceptuales: *nivel de conocimiento*, *nivel simbólico* y *nivel físico*. La transformación del nivel de conocimiento al nivel simbólico la realiza el programador utilizando un espacio simbólico en el que define un conjunto de procesos (algoritmos o programas) que operan con entradas pertenecientes al espacio simbólico de representaciones de entrada. Los procesos generan un conjunto de salidas pertenecientes a un espacio simbólico de representaciones de salida. El nivel físico contiene los procesadores que implementan los procesos. El paso del nivel simbólico al nivel físico es realizado por un programa traductor adecuado. "*El obstáculo básico es la programación en detalle, en vez de la capacidad biológica de madurar y aprender a representar la información presentada de forma poco ordenada en estructuras internas autoorganizativas y robustas, capaces de ser aplicadas a una amplia variedad de circunstancias*" [MIR95].

Los procesadores usuales están basados en las ideas desarrolladas por John von Neumann en 1945 [NEU45] ya que desde entonces no se ha concebido otra filosofía ampliamente aceptada para realizar este tipo de sistemas; aunque resulta evidente que se han producido mejoras sustanciales en muchos aspectos de la máquina que inicialmente propuso. Estas mejoras han sido debidas tanto a avances tecnológicos (en microelectrónica, óptica y mecánica) como arquitecturales y estructurales (optimizaciones de los repertorios de instrucciones máquina, mejoras en el diseño de las distintas unidades y sus interconexiones, e incluso introducción de elementos que permiten el procesamiento dirigido por datos en lugar de ser dirigido por instrucciones).

Un computador von Neumann dispone de una unidad central de control que capta instrucciones de una memoria central, y las ejecuta secuencialmente. El conjunto de tales instrucciones es llamado programa. Una evolución notable de estas máquinas von Neumann, o computadores SISD (una secuencia de instrucciones y una secuencia de datos), se debe al desarrollo de procesadores con segmentación de cauce y superescalares, así como al de computadores paralelos en sus distintas versiones: SIMD (una secuencia de instrucciones, múltiples secuencias de datos), y MIMD (múltiples secuencias de instrucciones operando simultáneamente con múltiples secuencias de datos), ésta última versión con dos variantes: de memoria compartida (multiprocesadores) y de memoria distribuida (multicomputadores). Con estos esquemas se consiguen mejorar sustancialmente las prestaciones de los computadores monoprocesador en cuanto a velocidad y fiabilidad de funcionamiento.

El esquema de computación descrito en los párrafos anteriores presenta limitaciones importantes,

la mayoría de las cuales se pueden considerar desde los siguientes puntos de vista:

> 1) Para implementar una tarea determinada es necesario pasar del dominio del conocimiento al dominio simbólico, y existen numerosos problemas cuyas soluciones son muy difíciles, o imposibles, de algoritmizar, por ser soluciones muy complejas o porque no se dispone del conocimiento imprescindible para plasmarlo en el espacio simbólico. Ejemplos de este tipo de tareas son el reconocimiento del habla, reconocimiento de imágenes, planificación del movimiento (obtención de la trayectoria para aparcar un coche), y percepción compleja (reconocer a una persona dentro de una multitud de un simple vistazo).
>
> 2) Los computadores actuales son extremadamente eficaces en los dominios de la computación numérica y tratamiento de símbolos (en general); pero hay muchas aplicaciones del mundo real en donde el procesamiento de la información en tiempos razonables requiere explotar la tolerancia a la imprecisión y la incertidumbre. Las máquinas de von Neumann no se adaptan bien a este tipo de problemas.

Dentro de las tareas indicadas pueden incluirse la emulación del procesamiento de información que los humanos realizamos rutinariamente. Muchos de los aspectos que se plantean caen dentro de lo que a veces se conoce como *soft computing* [ZAD95] o sistemas inteligentes.

El *soft computing* puede considerarse como un conjunto de metodologías de computación que tratan de aprovechar la tolerancia de la imprecisión, la incertidumbre y la verdad parcial para conseguir versatilidad, robustez, bajo coste de la solución y mejor aproximación con la realidad. En la actualidad se incluyen dentro de este concepto clases de metodologías tales como la lógica difusa, la neurocomputación, la computación genética y el razonamiento probabilístico . Aunque cada uno de estos paradigmas puede usarse para resolver cierto tipo de aplicaciones, en realidad no son competitivos sino complementarios, y en muchos casos se obtiene una gran sinergia si se utilizan en combinación (sistemas híbridos), en lugar de aisladamente. Como indica Zadeh [ZAD95], dentro del *soft computing* cada una de las clases de metodologías proporciona unas capacidades. Así la lógica difusa dispone de un cuerpo de conceptos y técnicas para tratar con la imprecisión, la granularidad de la información, el razonamiento aproximado, y, sobre todo, la computación con palabras. En el caso de la computación genética se presenta la posibilidad de la búsqueda aleatoria sistematizada con la consecución de un rendimiento óptimo. El razonamiento probabilístico proporciona un conjunto de técnicas para la gestión de la incertidumbre y el razonamiento abductivo.

La *neurocomputación*, en cambio, ofrece la capacidad de aprender, generalizar, adaptar e identificar. Se fundamenta en que el cerebro humano (y en general las redes neuronales

biológicas) presenta un conjunto de características de las que adolecen los computadores von Neumann (incluyendo los modernos computadores paralelos). Entre estas características se encuentran [PRI96]:

- paralelismo masivo
- representación y computación altamente distribuidos
- capacidad de aprendizaje
- capacidad de generalización
- adaptabilidad
- procesamiento de información inherentemente contextual
- tolerancia a fallos
- bajo consumo de energía

También hay que recordar que a los sistemas biológicos superiores, bajo el control de redes neuronales, se les atribuyen funciones tales como:

- conocimiento ("cognición)
- percepción (visión, oído, olfato, tacto, gusto, equilibrio)
- acción motora
- emoción
- instinto
- memoria
- aprendizaje
- lenguaje, etc

Ante el interés de realizar sistemas artificiales con las características y propiedades indicadas, y la evidencia de la calidad del tejido nervioso para procesar información y almacenar conocimiento, desde la década de los 40 se ha buscado la inspiración en dicho tejido para implementar aquellos sistemas. La emulación del sistema nervioso ha tratado de ser bien funcional o bien estructural, y con distintos grados de fidelidad.

En la historia de la neurocomputación cabe distinguir tres etapas. La primera etapa, se inicia a partir del modelo de McCulloch y Pitts en 1943, y se prolonga hasta 1960 aproximadamente; la segunda etapa, caracterizada por un cierto estancamiento durante los años 60 y 70; y por último, la tercera etapa, donde se produce un resurgimiento de la computación neuronal, y dura desde 1980 hasta nuestros días.
En 1943, McCulloch y Pitts introducen el primer modelo formal de neurona artificial, formado por una función lógica seguida de un retardo y en el que la programación se sustituye por el aprendizaje [MCC43]. De esta manera se inicia una primera etapa en la historia de las RNAs,

donde las ideas básicas (agrupadas bajo el nombre de *neurocibernética*) se basan en considerar que los seres vivos y las máquinas pueden ser comprendidos usando los mismos principios organizacionales y las mismas herramientas formales (la lógica y las matemáticas). Se busca la inspiración en la biología para obtener modelos de neuronas biológicas y redes de neuronas. Así, las redes neuronales artificiales se intentan utilizar para el reconocimiento de caracteres, recordar, inferir de forma distribuida, cooperar, aprender por refuerzo o autoorganizarse. Da comienzo a la teoría modular de autómatas y se utiliza la lógica (determinística y probabilística) como forma de representar el conocimiento que queda distribuido en las conexiones de la red.

Es curioso señalar que el diseño lógico (arquitectura) propuesto por von Neumann usa neuronas formales del tipo de McCulloch-Pitts, de forma que demuestra la equivalencia entre máquina de Turing y redes neuronales sobre una arquitectura programable y universal [MIR95]. De hecho, von Neumann en 1945 en su primer borrador sobre el computador EDVAC [NEU45], después de describir la unidad aritmética central (CA), la unidad central de control (CC) y la memoria (M) decía:

"...

> *2.6 The three specific parts CA, CC (together C), and M correspond to the associative neurons in the human nervous system. It remains to discuss the equivalents of the sensory or afferent and the motor or efferent neurons. These are the input and output organs of the device ...*"

Es decir, la estructura del primer computador de programa almacenado estaba, a grandes rasgos, inspirada en el sistema nervioso.

Sin embargo, a finales de los 60 las limitaciones de las redes de neuronas formales tipo perceptrón fueron puestas de manifiesto por Minsky y Papert [MIN69] en un trabajo sobre configuraciones que no son separables (reconocibles) mediante funciones de discriminación lineales, de forma que se frenó esta primera etapa conexionista.

Durante los años 70 y 80 se sigue trabajando en conexionismo aunque de forma más moderada, desarrollándose las redes probabilísticas [MOR68, MOR72], las memorias asociativas [AND89], el reconocimiento de caracteres [FUK80], el aprendizaje por refuerzo [GRO72, MIR71], la visión artificial a nivel de procesadores y a nivel de procesos [MAR76, MAR82], la decisión cooperativa [KIL68] y los modelos biofísicos e inferenciales que reclaman una extensión de la computación neuronal hacia el simbolismo de las formulaciones analíticas.

Al comienzo de los años 80s, las propuestas de Rumelhart [RUM86] renuevan el interés hacia las RNAs. Todavía permanece el concepto inicial de red neuronal como cálculo realizado por una arquitectura modular con gran número de elementos con alto grado de conectividad y realizando

una función analógica no lineal. La diferencia esencial está en la sustitución de la función umbral por una función derivable que permita la retropropagación de la función de error que guía el aprendizaje supervisado por un método de descenso de gradiente. El algoritmo de retropropagación ya había sido propuesto por Werbos en 1974 [WID90], no obstante pasó desapercibido hasta que fue redescubierto por Rumelhart en [RUM85].

También en los ochenta Hopfield introduce nuevas ideas [HOP82], basadas en una serie de trabajos de Hebb [HEB49], que permiten entrenar una clase de redes recurrentes. Los modelos de Hopfield propiciaron el desarrollo de otros tales como la máquina de Boltzmann [BEN95], en el que sobre un modelo de Hopfield se utilizan procesos de enfriamento simulado (Simulated Annealing [BER95A, BER96, PAR94]). Otras importantes aportaciones al conexionismo en esta época se deben a Kohonen con el desarrollo de los mapas autoorganizativos [KOH90].

Hoy día el conexionismo aborda diversos temas que van desde la neurociencia, buscando inspiración en las redes biológicas, hasta la implementación de arquitecturas paralelas de propósito específico tales como preprocesadores, coprocesadores o neurocircuitos completos, pasando por multitud de aplicaciones en problemas de percepción, control y clasificación. Las redes neuronales artificiales son sistemas que tratan de emular más o menos fidelignamente a las redes neuronales naturales, y existen distintas definiciones de ellas, dependiendo de que se haga más o menos hincapié en el símil biológico. Dos de estas definiciones son:

> a) Las redes neuronales artificiales son redes de elementos sencillos (usualmente adaptativos) interconectados paralela y masivamente, y sus organizaciones jerárquicas que tienen por misión interactuar con los objetos del mundo real de la misma forma en que lo hace el sistema nervioso biológico [KOH87, KOH88].
>
> b) Una red neuronal artificial es un sistema dinámico formado por unidades de procesamiento sencillas (neuronas artificiales, nudos, o primitivas) en el que la información se transfiere a través de los nudos por medio de interconexiones ponderadas (sinapsis o pesos) que pueden realizar procesamientos con los estímulos aplicados a un conjunto de entradas.

En el sentido más amplio una red neuronal es un sistema masivamente paralelo con un gran número de sencillos procesadores interconectados y que puede resolver problemas computacionales. Las redes neuronales artificiales a veces se denominan redes o sistemas conexionistas. La neurocomputación o computación neuronal puede definirse como la disciplina que versa sobre los sistemas de procesamiento de la información (redes neuronales) que autónomamente desarrollan capacidades operacionales en respuesta adaptativa a un entorno de información [HEC90].

El cerebro es un sistema masivamente paralelo con el orden de 10 a 100 mil millones (10^{11}) de elementos de procesamiento (neuronas), y cada neurona se conecta con hasta otras 10.000. Parece ser que las neuronas biológicas hacen unas operaciones muy sencillas con la información. El cerebro es capaz de resolver problemas difíciles de visión o habla en aproximadamente medio segundo. Este es un hecho sorprendente pensando que una neurona opera, sin considerar los tiempos de transmisión a través de ellas, en el rango de los milisegundos. Esta circunstancia implica que esas tareas complejas pueden efectuarse en tan sólo 100 "pasos" de procesamiento, mientras que en un computador convencional se necesitarían miles de millones de pasos.

La arquitectura del sistema neuronal biológico es completamente diferente a la arquitectura von Neumann. Estas diferencias significativas afectan al tipo de funciones que cada modelo computacional puede realizar. La Tabla 1.1 trata de resumir algunas de ellas.

Tabla 1.1. Comparación entre sistemas neuronales biológicos y computadores convencionales

	Sistema neuronal biológico	Computadores convencionales
Procesador	Gran cantidad Baja velocidad	Uno o relativamente pocos Alta velocidad
Computación	Flujo de datos Aprendizaje	Flujo de instrucciones Programas almacenados
Fragilidad	Robusto	Muy vulnerable
Interconexión entre procesadores	Irregular y masiva Cambia dinámicamente	Regular Estática / Dinámica limitada
Tipo de problemas a los que se adapta	Problemas de percepción, cómputo aproximado y cualitativo.	Manipulaciones numéricas y de símbolos
Entornos de funcionamiento	Pobremente definidos Imprecisos	Bien definidos Precisos

Los modelos artificiales de redes neuronales presentan o tratan de presentar algunas de las características de las redes biológicas, entre ellas:

a) La programación se sustituye por el aprendizaje. No es necesario conocer el algoritmo correspondiente a la tarea a realizar. La red neuronal puede descubrir patrones y sus

relaciones, y organizarse a si misma para efectuar asociaciones. De esta forma se realiza directamente el paso del nivel de conocimiento al nivel físico. Esta capacidad tiene dos implicaciones de suma importancia: la habilidad de resolver problemas cuyas reglas son difíciles de formular, y la habilidad de extraer de grandes conjuntos de datos con numerosas variables tanto modelos estadísticos como reglas basadas en el conocimiento. En ciertas aplicaciones clásicas es necesario conocer e introducir en la máquina las reglas para la toma de decisiones; en una red neuronal las reglas no existen de forma explícita y, en cierto sentido, puede decirse que se establecen automáticamente (mapas autoorganizativos).

b) Se busca en las redes neuronales artificiales que, al igual que ocurre en las biológicas, exista tolerancia a fallos físicos. Debe existir una "degradación elegante" lo que significa que la eliminación de un pequeño número de elementos de la red dé lugar a una mínima disfunción computacional [KAN95].

c) Una característica importante de los sistemas neuronales biológicos es la plasticidad, según la cual el número de conexiones se incrementa en las zonas de gran actividad y se reduce en las zonas sobreabundantes, donde la actividad no sobrepasa un determinado umbral. Utilizando un símil computacional se puede decir que el "soft" modifica al "hard" y viceversa. Esta facultad permite la realización de reconfiguraciones tras las lesiones. Este comportamiento trata de imitarse dentro del campo de arquitecturas evolutivas, en el que se pretende realizar sistemas electrónicos que se reconfiguren automáticamente ante una avería, manteniendo su función aunque sea de forma degradada [SAN96, SAN97].

d) Las aplicaciones usuales de las RNAs se basan en sus facultades de aprender, generalizar, adaptar e identificar, y entre ellas se encuentra la asociación de patrones, clasificación, detección de las regularidades y optimización. El reconocimiento de patrones se hace en paralelo y pueden recomponerse componentes incompletos o deformados en la entrada. Las RNAs pueden reconocer patrones espaciales y/o temporales, en presencia de ruido y con distorsión.

Como conclusión se puede decir que las RNAs, emulando modelos naturales, constituyen una herramienta valiosa y complementaria a los esquemas de cómputo convencionales para la resolución de problemas difícilmente algoritmizables o imprecisos. Además, al funcionar en paralelo todos los elementos de la red, pueden obtenerse grandes velocidades operativas (tiempo real) y tolerancia a fallos.

1.2. MODELOS ARTIFICIALES DE REDES NEURONALES

En las neuronas biológicas se distinguen tres partes: axón, soma y dendritas. Por medio del axón una neurona se puede conectar a las dendritas de otras. Estos contactos se realizan por medio de unos pequeños bulbos denominados sinapsis. El sentido de la transmisión es post-sinapsis, dendrita o soma, axón, pre-sinapsis. En el cerebro humano existen alrededor de 10^{11} neuronas y unas 10^{14} conexiones.

Las neuronas procesan señales eléctricas de origen químico. Cada neurona presenta un cierto potencial de membrana de unos -70 milivoltios en reposo y cuando recibe un estímulo procedente de alguna sinapsis se produce una variación en ese potencial cambiando su permeabilidad a ciertos iones, de forma que si se supera un cierto umbral, la neurona se "dispara" y se propaga una señal a través del axón y hacia sus sinapsis. La señal propagada tiene forma de tren de pulsos y este proceso puede repetirse a su vez en las neuronas vecinas.

En el modelo clásico de neurona artificial se representan las sinapsis por medio de un valor o peso (w) que pondera la entrada procedente de otra neurona vecina. De esta forma una neurona con n entradas presenta un potencial de membrana z igual a:

$$z \quad \sum_{j\ 1}^{n} w_j\ y_j \quad I$$

donde w_j es el peso ("sinapsis") asociado a la entrada y_j procedente de la salida de otra neurona o de una de las entradas a la red, e I es una entrada de control.

Si el potencial de membrana z es mayor que un determinado umbral t, la neurona se disparará. Matemáticamente se puede representar este hecho diciendo que la salida de la neurona es:

$$y \quad f(z \quad t)$$

donde f es una función no lineal denominada función de activación. La Figura 1.1 muestra un esquema del modelo de neurona descrito.

Habitualmente como función de activación, dependiendo del modelo o arquitectura de la red, se considera una función umbral, una función rampa, una función sigmoidal, o una función de base radial. En la Figura 1.2 se representan algunos ejemplos de funciones de activación ampliamente utilizadas en la literatura.

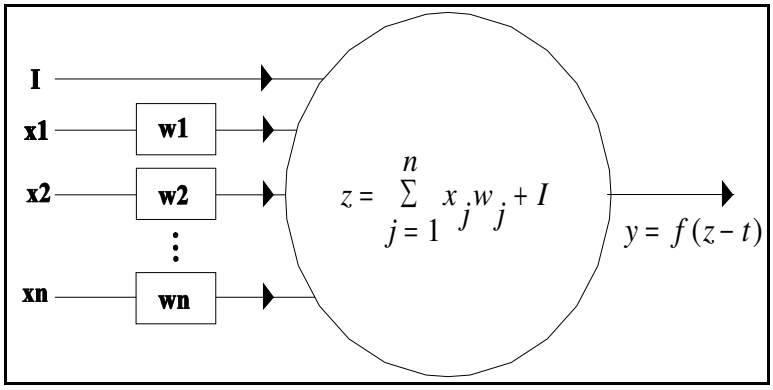

Figura 1.1. Modelo básico de neurona.

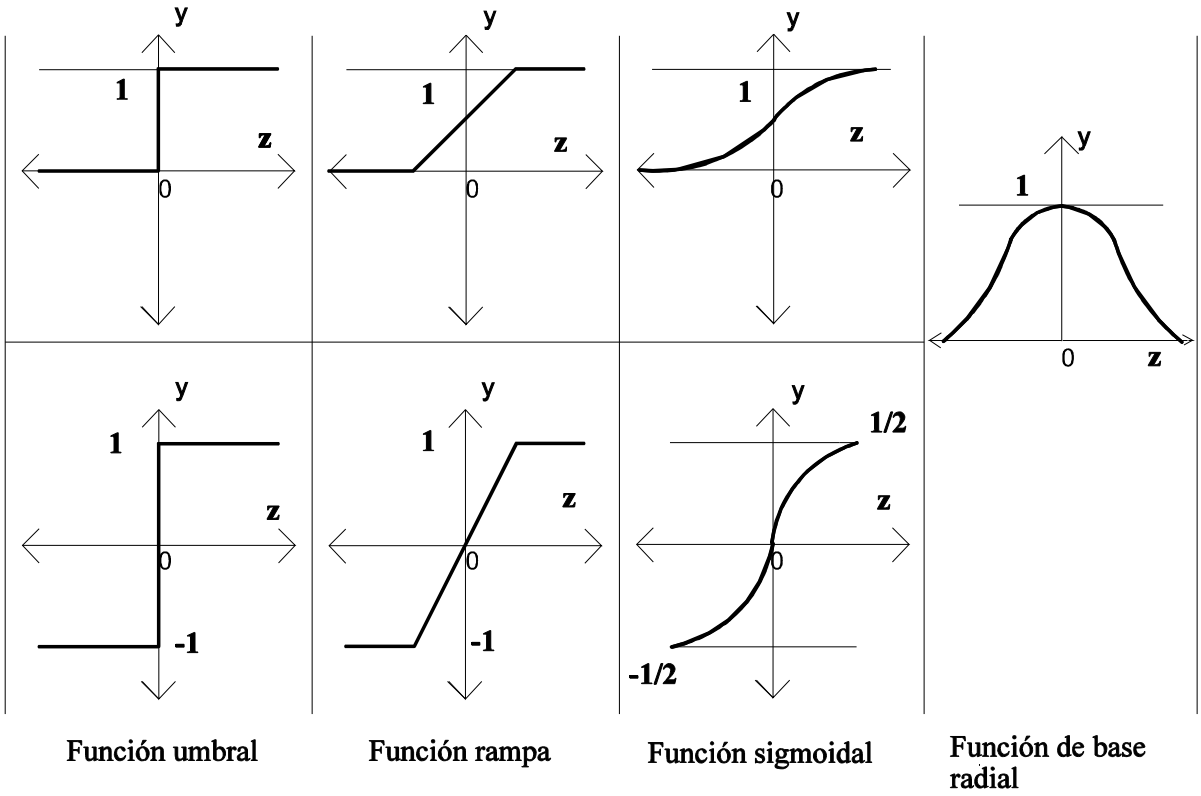

Figura 1.2. Distintas funciones de activación frecuentemente utilizadas.

Existen otros modelos donde se tiene más en cuenta la base biológica, y el valor de salida y_i representa un tren de impulsos ("spikes") que genera el circuito, de forma que el tiempo es una variable continua.

Las redes neuronales artificiales suelen estructurarse en capas, donde cada capa está constituida por cierto número de neuronas. Las entradas de las neuronas de una determinada capa provienen

de las salidas de las neuronas de la capa anterior, y sus salidas son las entradas a las neuronas de la capa posterior. No obstante, existen múltiples variantes. Hay modelos que sólo consideran conexiones hacia adelante (redes sin realimentación o "feed-forward") mientras que en otras se permite que las neuronas se conecten también a neuronas de capas anteriores (redes con realimentación, recurrentes o "feed-backward"). En otras, incluso se permite la conexión con neuronas pertenecientes a la misma capa. Un esquema de algunas de las topologías utilizadas en distintos modelos de RNAs se muestra en la Figura 1.3..

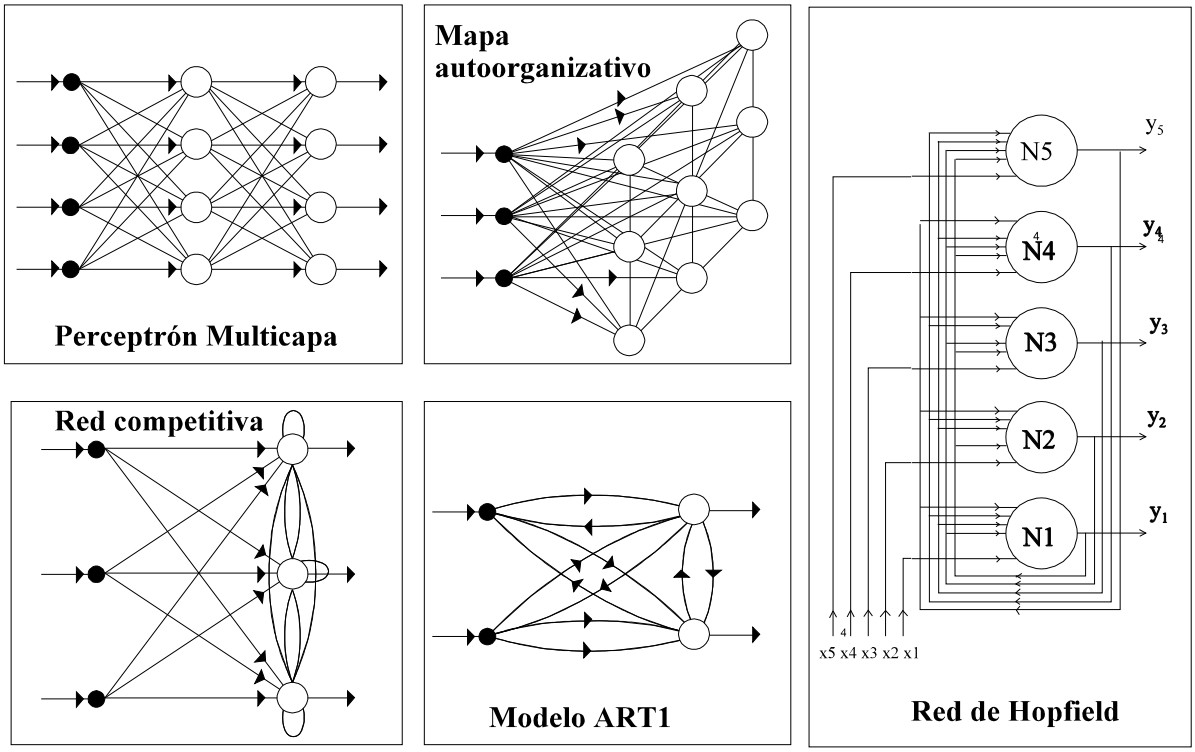

Figura 1.3. Topologías usuales de distintos modelos de RNAs.

El aprendizaje o autoprogramación es una de las principales características de las RNAs que las diferencia claramente de las computadoras convencionales. En el contexto de las RNAs, el aprendizaje es un problema de ajuste de los valores de los pesos y estructura de la red con objeto de que se realice eficientemente una tarea determinada. El aprendizaje se lleva a cabo a través de ejemplos (muestras) y pretende o bien formar asociaciones sencillas entre representaciones que se especifican externamente, o bien construir representaciones internas del entorno. El objetivo es que las redes *generalicen* (o *abstraigan*) correctamente casos de un determinado dominio, adquiriendo esa facultad a partir de las muestras presentadas durante el aprendizaje. La estructura de la red suele fijarse [WID90] de forma que el problema del aprendizaje se reduce a obtener los valores de los pesos que permiten la síntesis de la función deseada en la estructura escogida. La red neuronal se entrena para realizar una tarea concreta con un conjunto de muestras

representativas L, y para cada uno de los elementos de L se efectúa una modificación en los pesos encaminada a conseguir una mejora en las prestaciones de la red.

Las tres estrategias de aprendizaje que suelen usarse son:

a) *Aprendizaje supervisado*, durante la fase de aprendizaje se suministra a la red, junto a las entradas ejemplo, la salida esperada (o deseada) para cada una de las muestras del conjunto L, o una indicación del grado de acuerdo o error de salida ("refuerzo") obtenida con la entrada de entrenamiento (aprendizaje con refuerzo).

b) *Aprendizaje no supervisado*, no se requiere conocer la salida deseada para cada muestra usada durante el aprendizaje. La red construye modelos internos a través de la detección de regularidades o correlaciones en los vectores de aprendizaje, que la red categorza en grupos disjuntos, sin recibir ninguna información adicional [PRI90]. Esta estrategia parece ser la que está en mayor consonancia con la realidad biológica [HAY94].

c) *Aprendizaje híbrido*, es una combinación de las estrategias anteriores; así en ocasiones, partiendo de un proceso no supervisado para el aprendizaje de la red, sus prestaciones pueden mejorarse mediante una fase adicional de entrenamiento supervisado realizando así un *ajuste fino* [PRI96].

Hay gran cantidad de modelos de redes neuronales para multitud de aplicaciones. Entre los más populares se encuentran [PRI96]:

- Perceptrón multicapa (MLP) [WID90,CAÑ91,FER93]

- Red de funciones de base radial (RBF) [MUS92, BAP94].

- Mapa autoorganizativo de Kohonen (SOM) [KOH90,MER91,MER94]

- Modelos de la teoría de adaptación resonante (ART) [GRO72].

- Red de Hopfield [HOP82,ORT93A, BER95B].

- Redes neuronales celulares (CNN) [CHU88, ANG91, ANG95,ANG96].

En la Figura 1.4. se indican los campos de aplicación de los modelos descritos [MAR90, PAT96,

JAI96].

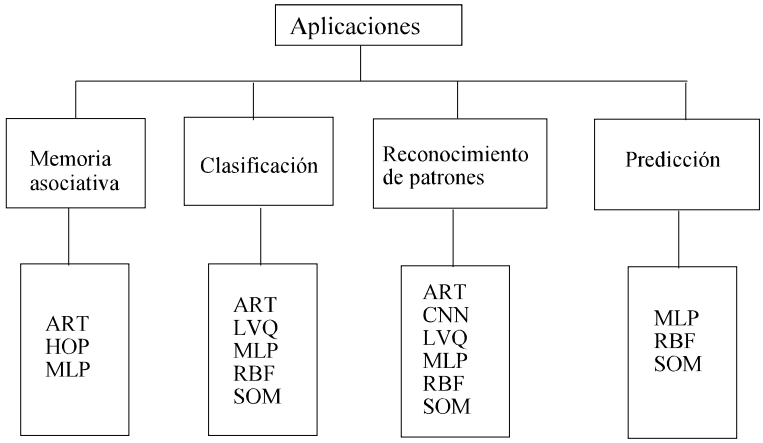

Figura 1.4. Campos de aplicación más relevante de distintos modelos de RNAs.

1.3. IMPLEMENTACIONES DE LAS REDES NEURONALES ARTIFICIALES

Hoy en día se han desarrollado multitud de posibilidades para implementar RNAs. Por un lado existen neurocircuitos, que son circuitos específicos que implementan una determinada estructura de red. Por otro lado se dispone de coprocesadores neuronales, es decir, de circuitos de propósito general que optimizan el cálculo de las operaciones elementales de los elementos neuronales.

En cualquier caso existen implementaciones tanto analógicas como digitales. También existe la posibilidad de simular redes neuronales artificiales mediante un computador convencional y, de hecho, es habitual que el proceso de entrenamiento se realice de esta manera, trasladando los pesos obtenidos a la implementación escogida [WID90]. Así, se reduce la complejidad de los circuitos puesto que prescinden de la lógica necesaria para el aprendizaje.

Un aspecto importante es que los circuitos neuronales, como cualquier otro circuito, puede sufrir daños, dejando de funcionar correctamente. A menudo se piensa que las RNAs, por analogía con las redes biológicas son robustas frente a fallos, pero eso no es cierto en general [SEG94, PAT95]. Téngase en cuenta el enorme número de neuronas y conexiones que existen en el cerebro humano (10^{11} y 10^{14}, respectivamente) y compárese con los correspondientes en un

circuito neuronal artificial (del orden de decenas). Si a eso se añade que las redes biológicas disponen de mecanismos que permiten redistribuir la estructura de conexión para recuperarse del daño y que este mecanismo sólo se puede emular añadiendo la circuitería que permita realizar un nuevo aprendizaje de la implementación dañada, entonces se ve claro que las RNAs no son en absoluto inherentemente tolerantes a fallos.

De esta manera, las implementaciones de RNAs precisan de algún tipo de test que permita detectar posibles fallos. Existen diversas técnicas de test [BER93A] que pueden aplicarse a los circuitos digitales [ORT91, ORT93A, ORT93B, GIL98] y analógicos [BER93B,BER95B].

Sin embargo, aún cuando las RNAs no son estrictamente tolerantes a fallos, algunas configuraciones de pesos se muestran más robustas que otras. Es decir, si el aprendizaje se ha distribuido de manera uniforme entre todas las neuronas, un daño en alguna de ellas provoca una mínima disfunción de la red [EDW98C], que posiblemente sea preferible a tener que sustituir el circuito. El proceso de aprendizaje no garantiza que esta distribución se consiga [EDW97], ya que, por ejemplo, en el caso del algoritmo de retropropagación sólo se busca que la configuración de pesos obtenida mediante el aprendizaje minimice el error de salida, pero eso no implica que todos los pesos contribuyan equitativamente a reconocer cada patrón de entrada.

Incluso, a veces podría pensarse que por el hecho de añadir más elementos neuronales, entonces se obtendrían mayores prestaciones en cuanto a robustez, debido a que se dispone de elementos redundantes. Esta suposición, también se ha probado no ser cierta [STE90], a menos que esta redundancia sea acompañada de una estrategia apropiada que reparta el aprendizaje entre todos los elementos. De hecho, incrementar el número de neuronas lo que aumenta es la *tolerancia potencial* de la red [EDW97], pero sin que ello implique un aumento efectivo de dicha tolerancia.

También se debe observar que, tal como antes se ha indicado, frecuentemente el algoritmo de entrenamiento se realiza en un ordenador convencional, trasladando los valores de los pesos resultantes a una implementación física concreta. Si la implementación es analógica, los valores cargados pueden diferir de los calculados debido a los efectos de la tolerancia de los elementos analógicos, mientras que si es digital se producen diferencias debido a los efectos de la cuantización o variaciones entre la precisión utilizada con el ordenador y la disponible en la implementación. En el contexto del test de circuitos este tipo de "defecto" se conoce como faltas paramétricas o de desviación [BER93A].

Por todas estas causas, resulta interesante disponer de:

a) Medidas para evaluar la robustez de una red neuronal artificial.

b) Medidas que permitan predecir la disfunción de una configuración cuando la precisión con que se van a cargar los pesos varía respecto de la obtenida tras el entrenamiento.

c) Algoritmos de aprendizaje en los que la distribución de los pesos favoreciera un incremento en dicha robustez.

Este ha sido el objetivo principal perseguido en este trabajo. Nos hemos centrado en el perceptrón multicapa (MLP), por ser un modelo ampliamente conocido y frecuentemente utilizado en aplicaciones muy diversas, tal como se desprende de la Figura 1.4, además de haberse demostrado que el MLP es un aproximador universal [HOR89]. A continuación se realiza una revisión de este modelo con objeto de introducir la notación y terminología seguida en esta memoria.

1.4. ESTRUCTURA GENERAL DE UN PERCEPTRÓN MULTICAPA

Un perceptrón multicapa o MLP consiste en una red compuesta por M capas, donde cada capa m (m=1...M) se compone de N_m neuronas. La neurona i de la capa m está conectada a las N_{m-1} neuronas de la capa previa mediante un conjunto de pesos w_{ij}^m ($j=1..N_{m-1}$). En la Figura 1.5 puede apreciarse un ejemplo de una estructura de red neuronal artificial con cuatro capas de neuronas (una capa de entrada, dos capas ocultas y una capa de salida). La función básica de la capa de entrada (capa 0) es propagar las entradas a la red a las neuronas de la primera capa oculta, por lo que realmente se trata de una capa virtual que no posee neuronas.

La salida de la neurona i de la capa m, y_i^m, está determinada por la activación z_i^m, definida como la suma ponderada de las salidas de las neuronas de capa anterior, y una función f_i^m llamada función de activación, usualmente continua, derivable y no lineal con respecto a z_i^m, tal como una función sigmoidal o una tangente hiperbólica:

$$y_i^m \quad f_i^m(z_i^m) \quad f_i^m\left(\sum_{j\ 1}^{N_{m\ 1}} w_{ij}^m\ y_j^{m\ 1}\right) \quad f_i^m \qquad i\ 1,...,N_m \tag{1.1}$$

donde los términos y_j^{m-1} son las salidas de las N_{m-1} neuronas de la capa previa. En particular, y_j^0 ($j=1...N_0$) son las entradas a la red.

En la Figura 1.6 puede apreciarse en detalle cómo se realiza la interconexión entre las neuronas pertenecientes a una capa m-1 y una neurona i de la capa siguiente m. Se especifican los pesos correspondientes a dichas conexiones, así como las salidas de cada una de las neuronas. En la figura se esquematiza la función realizada por la neurona i: calcular la suma ponderada de las entradas con los pesos correspondientes en un primer paso, y aplicar una función logística a dicha suma.

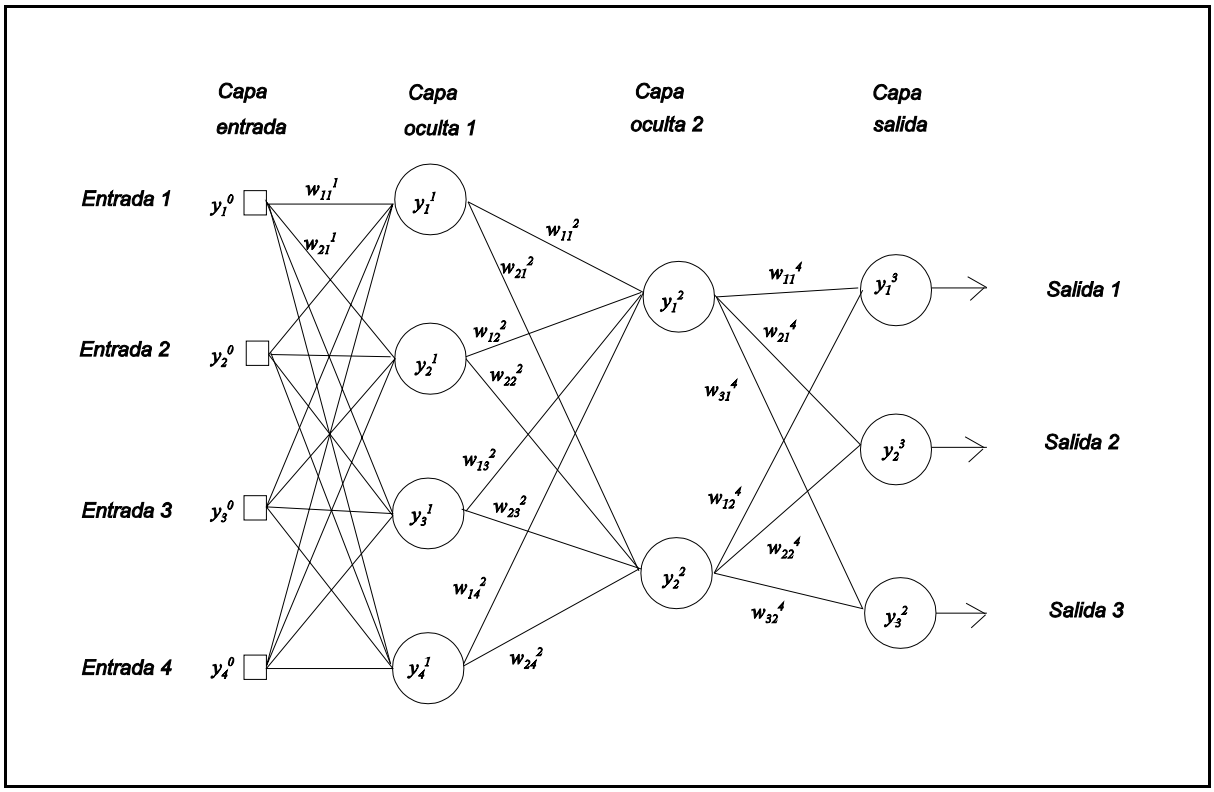

Figura 1.5. Estructura de un perceptrón con 4 capas.

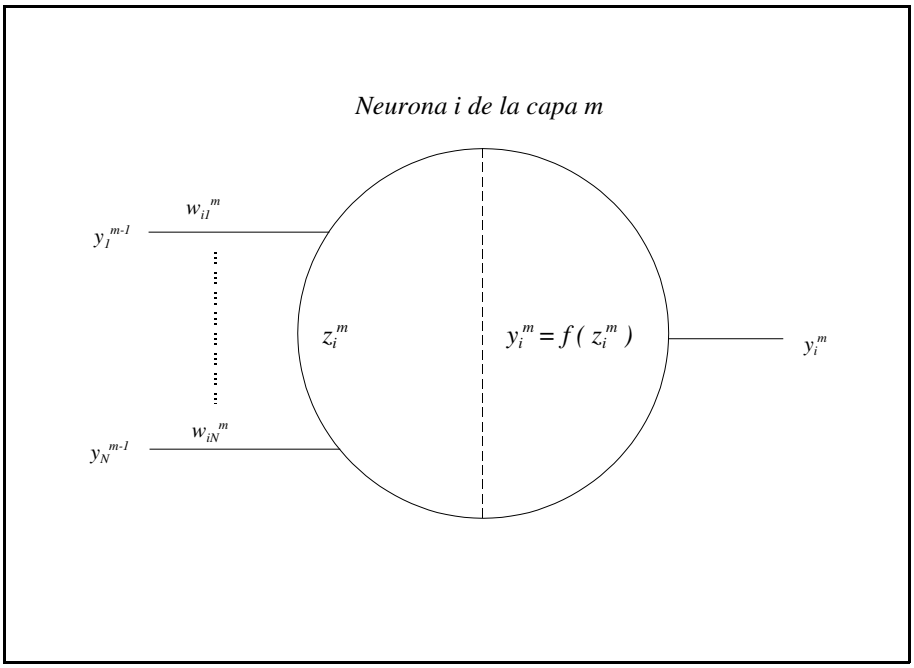

Figura 1.6. Detalle de la interconexión entre neuronas de capas adyacentes.

A menudo se añade un peso modificable a cada neurona llamado sesgo (“offset” o “bias”). Este peso, que suele notarse con el subíndice 0 (w_{i0}^{m}), se considera conectado a una entrada ficticia fijada al valor 1 y tiene como objetivo variar el origen de la función de activación permitiendo una convergencia más rápida del proceso de entrenamiento.

Fijada la estructura deseada para un perceptrón multicapa (MLP), es decir, el número de capas y de neuronas de cada capa, el ajuste de los pesos de las neuronas constituyentes se realiza mediante un algoritmo de aprendizaje. El algoritmo de aprendizaje usual es el conocido con el nombre de "algoritmo de retropropagación" [WID90, ANZ89] y precisa de un conjunto de patrones de entrada para los que se conoce la salida deseada. A tal conjunto se le denomina conjunto de entrenamiento.

Así, con ayuda de este conjunto de entrenamiento, se lleva a cabo un proceso de aprendizaje supervisado. Durante este proceso, los patrones de entrenamiento se presentan a la red uno a uno, y la salida que proporciona el perceptrón se compara con la salida deseada para el patrón presentado obteniéndose un error de salida. Los pesos de cada neurona se ajustan en función del error de salida usando un algoritmo de descenso de gradiente. Este algoritmo propaga el error de salida desde la capa de salida hacia la de entrada y varía los pesos en sentido opuesto al del gradiente del error. El proceso de aprendizaje se lleva a cabo durante un cierto número de iteraciones llamadas épocas, donde cada época se caracteriza por la presentación del conjunto de entrenamiento completo a la red.

Los pasos de que consta este algoritmo son básicamente los siguientes [WID90]:

1. Inicializar los pesos del MLP con valores aleatorios pequeños.

2. Para cada vector del conjunto de entrenamiento se repiten los siguientes pasos hasta que el error de salida para el conjunto completo se considere aceptablemente bajo, en cuyo caso la red se dice que está entrenada y puede ser usada para reconocimiento.

    2.a. Presentar entrada al MLP y calcular la salida de la red

    2.b. Calcular el error de salida respecto de la salida deseada.

    2.c. Ajustar los pesos usando un algoritmo de gradiente que se presenta a continuación.

El ajuste de los pesos tiene por objeto minimizar el error cuadrático en la salida (1.2); éste se

calcula comparando la salida deseada para el patrón de entrada con la salida que proporciona la red. Por este motivo, comenzando por los elementos de la última capa, se usa un algoritmo que modifica los valores de los pesos en sentido opuesto al del gradiente de dicho error. Este proceso se repite en la capa anterior, y así sucesivamente hasta llegar a la primera capa. Es por esto que recibe el nombre de algoritmo de retropropagación. La expresión del error cuadrático para un determinado patrón de entrada p, ε(p), viene dada por la expresión:

$$\varepsilon(p) \quad \frac{1}{2} \sum_{i\ 1}^{N_M} (d_i(p) \quad y_i(p))^2 \tag{1.2}$$

donde $d_i(p)$ e $y_i(p)$ representan la salida deseada y calculada de la neurona i perteneciente a la última capa para el patrón de entrada p, respectivamente, y N_M es el número de neuronas de la capa de salida.

El cálculo exacto del gradiente implica realizar el ajuste de los pesos al final de cada época, es decir tras haber presentado al MLP todo el conjunto de entrenamiento [ANZ94] de forma que se minimice el error cuadrático medio (ECM) para el conjunto de patrones de entrenamiento, sin embargo, se suele realizar la siguiente aproximación : los pesos se modifican tras presentar cada patrón de entrenamiento usando la regla siguiente [LIP87, ANZ94]:

$$(w_{ij}^{m})^{t\ 1} \quad (w_{ij}^{m})^{t} \quad \alpha((\ \varepsilon)_{ij}^{m})^{t} \quad \beta((w_{ij}^{m})^{t} \quad (w_{ij}^{m})^{t\ 1}) \tag{1.3}$$

En esta expresión t indica el número de iteración, α es el llamado "factor de aprendizaje", β es conocido como "momento" y $(\varepsilon)_{ij}^{m}$ es el gradiente del error cuadrático de salida respecto al peso w_{ij}^{m}. El factor de aprendizaje α es el responsable de la minimización del error cuadrático mientras que el factor β o momento tiene por misión evitar cambios bruscos suavizando dichos cambios durante el aprendizaje [ANZ94].

En los capítulos restantes, al algoritmo de aprendizaje que usa la regla (1.3) durante el entrenamiento le llamaremos “Algoritmo de Retropropagación Clásico” y lo denotaremos con las siglas “ARC”.

Usando esta regla, los pesos se van adaptando para cada patrón de entrenamiento presentado a la entrada. Si se varían las condiciones iniciales (es decir, se comienza con distintos valores iniciales de los pesos) o se cambian los parámetros α ó β de la regla (1.3) se obtienen distintas configuraciones para los pesos que, aunque puedan proporcionar similares prestaciones en cuanto a error cuadrático medio de salida, puede variar la tolerancia a fallos obtenida para la red. Esto mismo se puede decir respecto de la capacidad de generalización o de la inmunidad a perturbaciones en las entradas del MLP. Así, puesto que es posible encontrar soluciones con

similares prestaciones en cuanto a aprendizaje (ECM similar) pero mejores prestaciones en cuanto a las características anteriormente mencionadas (tolerancia a fallos, capacidad de generalización, inmunidad al ruido), resulta interesante disponer de algún mecanismo que permitiese obtener de forma automática dichas soluciones.

1.5. LAS TÉCNICAS DE REGULARIZACIÓN

Una forma que se ha probado efectiva para obtener RNAs más robustas se basa en la teoría de la regularización [MAO93]. Las técnicas basadas en la teoría de la regularizción inicialmente persiguen incrementar la capacidad de generalización de las RNAs [BOS97], no obstante, como efecto adicional se obtiene un reparto más equitativo del aprendizaje entre los elementos neuronales y, de aquí, configuraciones más tolerantes a desviaciones [MAO93].

Las técnicas de regularización consisten básicamente en penalizar la función a minimizar durante el aprendizaje mediante la adición de un nuevo término. Por ejemplo, en el caso del algoritmo de retropropagación se trata de minimizar simplemente el error cuadrático medio (ECM). Utilizando las técnicas de regularización, se trata de incluir en la minimización un término adicional:

$$T \quad ECM \quad \lambda E^{p} \tag{1.4}$$

donde E^p es un término de penalización conocido con el nombre de *estabilizador*. El estabilizador puede adoptar distintas formas, pero en cualquier caso, contiene términos relacionados con la salida obtenida con la red y(w,x). Así, habitualmente se utilizan regularizaciones de orden cero, primer o segundo orden, según si E^p se basa en la salida, primera o segunda derivada de la salida, respectivamente. Es decir, E^p tiene la forma genérica siguiente:

$$E^{p} \qquad \int_{D} \left| \frac{d^{p} y(w,x)}{dx^{p}} \right| \, p(x) \, dx \tag{1.5}$$

donde D se refiere al espacio de entradas (el conjunto de vectores de entrenamiento) y p(x) es la función de densidad de la probabilidad para la variable de entrada x, siendo w el vector de pesos de la red.

Las técnicas de regularización se pueden clasificar en tres tipos [MAO93]:

Tipo I: Regularización *inherente a la arquitectura*. Cuando se escoge una arquitectura

concreta para una implementación dada, se está limitando el espacio de soluciones. Consecuentemente, la especificación de una arquitectura de red introduce un cierto grado de regularización [GIR95].

Tipo II: Regularización *inherente al algoritmo*. Este tipo de regularización se obtiene por ejemplo, debido al criterio seguido para parar el entrenamiento de la red. De esta manera, se puede conseguir que la red generalice mejor o peor según el criterio utilizado. También se ha probado que la adición de ruido en los patrones de entrada implica un incremento en la generalización obtenida por la red ya que el efecto es equivalente a la adición de términos de las derivadas de primer y segundo orden de las salidas respecto a los pesos [EDW96].

Tipo III: Regularización *explícitamente especificada*. En este caso se añade un estabilizador E^p a la función a minimizar. Es habitual condicionar la suavidad de las soluciones, es decir, buscar mínimos anchos, en los cuales si se produce una cierta perturbación la solución sigue estando recluida en su entorno; en cambio, los mínimos locales estrechos tienen el peligro que con una mínima perturbación, la solución escapa de dicho mínimo pudiendo degradar drásticamente las prestaciones. Con estabilizadores de segundo orden se obtienen curvas de salida más suaves, de forma que se obtiene mejores prestaciones en cuanto a generalización. Escogiendo un estabilizador adecuado se puede conseguir mejorar las prestaciones de generalización e incluso de tolerancia a fallos uniforme [MAO93].

1.6. CONCLUSIONES

Las redes neuronales artificiales están en principio inspiradas en las redes biológicas, y por ello tratan de modelarlas con objeto de heredar algunas de sus características. Una de las principales características que presentan las RNAs es el aprendizaje, lo que las diferencia claramente de otros paradigmas computacionales, de forma que prescinden de la programación en sentido estricto.

Otra característica deseable, es que las RNAs sean tolerantes a fallos. Esta característica a menudo es asumida, pero lo cierto es que algunas configuraciones de pesos presentan mayor robustez que otras, ya que usualmente no se obtiene un reparto uniforme del aprendizaje entre los distintos elementos de cómputo. Los modelos de RNAs no son inherentemente tolerantes a fallos y existe una serie de conjeturas acerca de cómo se puede conseguir dicha tolerancia. De manera análoga, es deseable incrementar la capacidad de generalización de la red, puesto que ésta también es una propiedad que varía para distintas configuraciones de pesos. Existen algunas medidas propuestas para estas propiedades, basadas principalmente sólo en conjeturas, de forma

que a menudo no son adecuadas ni ampliamente aceptadas.

Uno de los modelos de RNA más utilizados es el perceptrón multicapa, cuyas principales características se han resumido con la intención de introducir la notación utilizada en el resto de capítulos, en los que se van a obtener medidas adecuadas para evaluar la tolerancia a faltas paramétricas, la inmunidad al ruido y la capacidad de generalización. Así, se proponen modelos cuantitativos y se justifica su validez.

Los modelos propuestos se utilizan para generar configuraciones de pesos que presentan ventajas en cuanto a tolerancia a fallos, inmunidad al ruido o capacidad de generalización, manteniendo además el resto de las prestaciones que proporciona un algoritmo de entrenamiento habitual. Por tanto, se presentan algoritmos de aprendizaje para incrementar las propiedades mencionadas.

Es interesante hacer notar que, a partir del estudio realizado, se obtienen expresiones matemáticas que permiten justificar muchas de las conjeturas acerca de la tolerancia a fallos y capacidad de generalización realizadas previamente por otros autores.

El siguiente capítulo presenta una medida cuantitativa que permite evaluar la robustez de una configuración de pesos en un MLP. Se obtiene la expresión de esta medida, llamada "sensibilidad estadística", para dos modelos de desviación distintos, y se presentan resultados que muestran la validez de estas expresiones. En los capítulos posteriores, esta medida es utilizada como estabilizador en el desarrollo de nuevos algoritmos de aprendizaje y es relacionada explícitamente con la degradación del error de salida cuando el MLP está sujeto a perturbaciones.

También se realiza la comparación de los procedimientos propuestos con otros algoritmos de aprendizaje usados habitualmente, utilizando varios ejemplos presentes en la bibliografía y problemas reales usados como *benchmarks*. A partir de los resultados obtenidos se deduce que se consigue un aumento en la robustez e inmunidad al ruido, así como una mayor capacidad de generalización.

Capítulo 2

Estudio de la Sensibilidad Estadística en Perceptrones Multicapa

---

En este capítulo se hace un estudio de la sensibilidad de los perceptrones multicapa frente a desviaciones en sus pesos, con objeto de estudiar su tolerancia a fallos. Esta medida permitirá desarrollar un algoritmo para aumentar la tolerancia de un perceptrón multicapa a modificaciones en los pesos de las unidades sinápticas, que se presentará en el capítulo siguiente. En el presente capítulo, en primer lugar se hace una discusión de la literatura existente sobre el tema. En segundo lugar, haciendo uso de la notación presentada en el Apartado 1.4 se introduce el concepto de sensibilidad estadística, una magnitud que ha sido propuesta para evaluar la tolerancia a faltas paramétricas de un MLP, y se presentan dos modelos de desviación, uno aditivo y otro multiplicativo, a partir de los cuales se obtienen las expresiones que permiten calcular dicha magnitud. Por último, se muestran resultados experimentales que permiten validar tales modelos y concluir que una menor sensibilidad estadística implica una mayor tolerancia del perceptrón a perturbaciones en los pesos de sus conexiones, terminando con una discusión sobre las conclusiones derivadas de tales resultados.

2.1. INTRODUCCIÓN

Como cualquier otro circuito electrónico, durante el proceso de fabricación o incluso durante la vida activa de una implementación electrónica de un modelo neuronal (circuito neuronal), éste puede verse afectado por distintos tipos de defectos físicos que pueden degradar sus prestaciones [ELS94]. Los defectos producidos durante el proceso de fabricación pueden detectarse aplicando diferentes procedimientos de test al final del proceso. No obstante, para detectar los defectos producidos durante la vida activa del circuito es preciso o bien realizar periódicamente tests al mismo, o bien hacer el circuito tolerante a tales defectos.

Como en principio las redes neuronales artificiales (RNAs) se inspiraron en las redes neuronales naturales, a menudo se asume que las RNAs son tolerantes a fallos. Aún más, como los elementos de una RNA no son lineales, los cambios en alguno de sus parámetros pueden producir modificaciones no proporcionales en sus salidas para algunas configuraciones de entrada. De este modo, algunas aplicaciones pueden tolerar desviaciones de cierta magnitud en los parámetros de la red. Parece ser que las redes neuronales naturales muestran una gran tolerancia a faltas debido principalmente a su plasticidad o capacidad para cambiar las interconexiones de sus neuronas con el objetivo de sustituir a las neuronas dañadas, y a la elevada conectividad de dichas neuronas [NIJ89, MIR95]. Así, en principio, sería posible incrementar la tolerancia a faltas de una RNA si se incluye la capacidad de aprendizaje adaptativo en su implementación [FRY91] o bien haciendo que la RNA sea suficientemente redundante [BEL89, PAT95]. Ambas soluciones implican un incremento en el hardware y las tecnologías actuales limitan el número de neuronas que se pueden incluir en un mismo circuito integrado.

Algunos autores [SEG94, PAT95, EDW98B] han aportado resultados que demuestran que las RNAs no son inherentemente tolerantes a faltas por lo que se hace necesario definir medidas cuantitativas para determinar la robustez de RNAs. En [SEG94] se propone una medida de la tolerancia basada en la simulación de faltas. En [PAT95], se considera el incremento de la redundancia de la red como una forma de obtener redes tolerantes a faltas, y se proporcionan métricas para cuantificar la tolerancia a faltas en función de dicha redundancia. Otros autores [STE90, ALI94] han estudiado el impacto de las faltas en la computación neuronal de estructuras de tipo Madaline [WID62, HOF62] con funciones de activación escalón y múltiple escalón, usando un modelo geométrico para determinar la influencia de las desviaciones de los parámetros en el error de clasificación de estas redes. Por otro lado, los efectos de la precisión con que se almacenan los valores de los parámetros de la red en las implementaciones físicas han sido estudiados en diversos trabajos [CAS91, REY91, XIE92] probando que, a consecuencia de la cuantización, las prestaciones de la red pueden verse fuertemente afectadas, aspecto especialmente relevante en el caso de que el entrenamiento se haya realizado “off-line” mediante simulación en un computador y los pesos obtenidos deban ser trasladados a una implementación

determinada [STE97].

Algunos trabajos indican la posibilidad de hacer la red redundante, con objeto de aumentar su tolerancia a faltas a costa de un incremento del número de neuronas [BEL89, PHA95]; mientras, en otras publicaciones, se considera la modificación de las técnicas de entrenamiento. De esta manera, en [MUR93, MUR94] se propone la inyección de ruido sináptico y, en [SEQ90,CHI94, LIN94, SEG94] se introducen faltas intermitentes durante el proceso de aprendizaje como medio para aumentar la tolerancia a fallos del perceptrón entrenado. En [CHI94] también se describe un procedimiento basado en forzar a que los pesos obtenidos tomen valores absolutos pequeños aunque ello suponga una degradación en las prestaciones de la red, que se resuelve incrementando el tamaño de la misma.

Los procedimientos propuestos para calcular la tolerancia a faltas o para modificar el proceso de aprendizaje basados en la simulación de faltas presentan algunos inconvenientes. Por un lado, necesitan usar un modelo de defecto que sea suficientemente preciso para describir correctamente los efectos de una falta, y suficientemente simple como para permitir la simulación de un alto número de faltas en un tiempo de cómputo razonable. Por otra parte, la bondad de un modelo de defecto dado depende de la tecnología usada para implementar el circuito. Así, por ejemplo, las faltas que modelan los defectos físicos que presenta un circuito digital son diferentes de las de uno analógico, y diferentes para un circuito de tecnología TTL y CMOS. Usualmente se diseñan procedimientos de test, se realizan análisis de testeabilidad, y se evalúa la tolerancia a fallos considerando que los circuitos sólo se ven afectados por una única falta, usualmente de tipo anclaje (“stuck-at”)[ORT93B]. Sin embargo, es importante tener en cuenta que la hipótesis de considerar sólo faltas simples se basa en la baja probabilidad de que un circuito se vea afectado simultáneamente por varios defectos, pero para ciertas tecnologías, es posible encontrar defectos que deben ser modelados mediante varias faltas que actúan simultáneamente.

Por otra parte, el número de faltas simples también puede ser demasiado elevado, o incluso no ser finito, en el caso de circuitos analógicos cuyas faltas dependan de parámetros que puedan tomar cualquier valor real. De esta manera, la simulación de todas las posibles faltas es imposible y requiere una selección aleatoria de las mismas, tal como se hace en [PHA95]. Además, tal como se indica en [EDW97, EDW98C] las faltas de anclaje no son realistas para la mayoría de implementaciones de RNAs, siendo más adecuado considerar faltas de desviación o paramétricas.

La tolerancia a fallos parece estar relacionada con la capacidad de generalización. De esta manera, en [EDW95] se propone medir la robustez a partir de la capacidad de generalización usando la diagonal de la matriz hessiana del error de salida respecto del valor de los pesos. Sin

embargo, tal como indican los propios autores, la relación generalización-tolerancia no está demasiado clara.

Otra aproximación diferente al problema de la tolerancia a faltas consiste en proponer una medida para evaluar la sensibilidad de la red frente a posibles defectos en su implementación. En esta línea, en [CHO92] se propone una medida para la tolerancia a faltas paramétricas de perceptrones multicapa, que recibe el nombre de *sensibilidad estadística.* Usando esta medida se puede establecer un criterio de selección entre diferentes conjuntos de pesos con similares prestaciones en cuanto a error de salida. Los autores muestran implícitamente que a menor sensibilidad estadística, mayor es la tolerancia a desviaciones en los valores de los pesos.

En esta memoria se propone un algoritmo de retropropagación modificado que, basándose en la medida de sensibilidad estadística anteriormente indicada, permite obtener directamente configuraciones de pesos que aún manteniendo las prestaciones del algoritmo de retropropagación clásico, minimizan la sensibilidad del perceptrón. A diferencia de otras medidas de tolerancia previamente propuestas, que dan información sobre la tolerancia global de la red [ALI94, SEG94, PHA95], las expresiones que aquí se obtienen permiten evaluar la tolerancia de cada nodo, e incluso de cada peso, por lo que puede ser incluida de una manera relativamente directa como un factor más dentro de la regla de aprendizaje.

2.2. SENSIBILIDAD ESTADÍSTICA DE UN PERCEPTRÓN MULTICAPA

En esta sección se desarrollan las expresiones que permiten calcular la sensibilidad estadística de un perceptrón multicapa. La sensibilidad estadística puede tomarse como una medida cuantitativa de la tolerancia de la red, puesto que permite estimar cuanto varía la salida de la red cuando sus pesos se modifican respecto de un valor nominal en una cierta magnitud. Primero se define la sensibilidad estadística de cada neurona, lo que hará posible obtener una medida local en cada nodo de la red, y en segundo lugar se propone la sensibilidad estadística media de la red como figura de mérito susceptible de ser utilizada para evaluar la tolerancia a fallos. La sensibilidad estadística media se calcula en función de la sensibilidad estadística de las neuronas de salida. Ello se debe, a que como se mostrará, la sensibilidad estadística de las neuronas de una determinada capa influye en la correspondiente a neuronas de la capa siguiente.

Tal como se vio en el Apartado 1.4 la salida de cada neurona i perteneciente a la capa m, y_i^m, viene dada por la siguiente expresión

$$y_i^m = f_i^m(z_i^m) = f_i^m\left(\sum_{j=1}^{N_{m-1}} w_{ij}^m\, y_j^{m-1}\right) \equiv f_i^m \qquad i=1,...,N_m \tag{2.1}$$

Si hay alguna desviación en los pesos de dicha neurona, y dicha desviación se asume suficientemente pequeña, la variación en la salida de la neurona para un patrón de entrada puede aproximarse como:

$$\Delta y_i^m \approx \sum_{j=1}^{N_{m-1}} \left(\frac{\partial y_i^m}{\partial w_{ij}^m} \Delta w_{ij}^m + \frac{\partial y_i^m}{\partial y_j^{m-1}} \Delta y_j^{m-1} \right) \tag{2.2}$$

donde se han despreciado los términos de orden superior a 1. Si se desarrolla esta expresión explícitamente teniendo en cuenta la dependencia de y_i^m con z_i^m se obtiene que:

$$\begin{aligned} \Delta y_i^m &= \sum_{j=1}^{N_{m-1}} \left(\frac{\partial y_i^m}{\partial z_i^m} \frac{\partial z_i^m}{\partial w_{ij}^m} \Delta w_{ij}^m + \frac{\partial y_i^m}{\partial z_i^m} \frac{\partial z_i^m}{\partial y_j^{m-1}} \Delta y_j^{m-1} \right) \\ &= \frac{\partial y_i^m}{\partial z_i^m} \sum_{j=1}^{N_{m-1}} \left(\frac{\partial z_i^m}{\partial w_{ij}^m} \Delta w_{ij}^m + \frac{\partial z_i^m}{\partial y_j^{m-1}} \Delta y_j^{m-1} \right) \end{aligned} \tag{2.3}$$

Tras sustituir los valores de las derivadas en la expresión anterior se obtiene finalmente:

$$\Delta y_i^m = \partial f_i^m \sum_{j=1}^{N_{m-1}} (y_j^{m-1} \Delta w_{ij}^m + w_{ij}^m \Delta y_j^{m-1}) \tag{2.4}$$

donde, por simplicidad de notación, se utilizará ∂f_i^m para indicar la derivada de y_i^m respecto de z_i^m.

Nótese que en las expresiones no se han incluido los pesos de sesgo a cada neurona. En caso de querer incluirlos no hay más que añadir una entrada ficticia para cada neurona, y_0^{m-1}, con valor prefijado a 1, por lo que el índice j de la suma comenzaría en el valor 0, incluyendo el peso correspondiente a dicha entrada tal como se indicó en el Apartado 1.4.

Definición 2.1.: La sensibilidad estadística de una neurona i perteneciente a la capa m se define como la siguiente expresión [CHO92]:

$$S_i^m = \lim_{\sigma \to 0} \frac{\sqrt{var\ (\Delta y_i^m)}}{\sigma} \qquad i=1,...,N_m \ ; \quad m=1,...,M \tag{2.5}$$

donde σ representa la desviación estándar de las desviaciones en los pesos y $var(\Delta y_i^m)$ es la varianza de Δy_i^m, la cual puede calcularse como:

$$var\ (\Delta y_i^{\ m}) = E\ [(\Delta y_i^{\ m})^2\ ] - (E\ [\Delta y_i^{\ m}\ ])^2 \qquad (2.6)$$

siendo E[◦] el valor esperado de [◦].

Las expresiones anteriores, (2.5) y (2.6), corresponden a la sensibilidad estadística de cada neurona a un patrón de entrada determinado. Nótese que la sensibilidad estadística evalúa cuánto varía la salida de la neurona al producirse perturbaciones en los pesos. Para proporcionar una medida cuantitativa de la sensibilidad global de la red se define la sensibilidad estadística media de la red de la siguiente manera:

Definición 2.2.: La sensibilidad estadística media de la red (SSM) se define como el promedio de las sensibilidades estadísticas de las neuronas de la capa de salida sobre el conjunto de patrones de entrada [CHO92], tal y como indica la siguiente expresión:

$$SSM = \frac{1}{N_p} \sum_{p=1}^{N_p} S^M = \frac{1}{N_p N_M} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} S_i^{\ M} \qquad (2.7)$$

En esta expresión N_p es el número de patrones de entrada, N_M es el número de neuronas de la capa de salida y S^M es la sensibilidad estadística media de las neuronas de salida para un patrón de entrada. En [CHO92] se proponen otras posibles definiciones para la sensibilidad estadística de la red que, dependiendo del tipo de función implementada por el MLP, pueden ser más indicadas como medida de la tolerancia a fallos. No obstante, en el caso presente, la expresión (2.7) constituye una figura de mérito conveniente para comparar los resultados que se presentarán en el siguiente capítulo. A continuación se obtienen las expresiones de la sensibilidad estadística para desviaciones de tipo aditivo (Sección 2.4) y de tipo multiplicativo (Sección 2.5).

2.3. CÁLCULO DE LA SENSIBILIDAD ESTADÍSTICA USANDO UN MODELO DE DESVIACIÓN ADITIVO

Para realizar el cálculo de la sensibilidad es necesario calcular la varianza de $\Delta y_i^{\ m}$, $var(\Delta y_i^{\ m})$, tal como se observa en la expresión (2.5). En primer lugar se considerará un modelo de desviación en los pesos de tipo aditivo. El modelo aditivo asume que cada peso $w_{ij}^{\ m}$ es modificado de forma que el nuevo valor viene dado por $(w_{ij}^{\ m})^* = w_{ij}^{\ m} \pm \delta$, donde δ es una variable aleatoria con desviación típica igual a σ.

Este modelo de desviación puede ser utilizado, por ejemplo, para estudiar los efectos del sesgo ("offset") presentado por los transistores en una implementación electrónica de una red neuronal,

ya que este tipo de defecto supone una perturbación de tipo aditiva a la intensidad de salida de dichos transistores.

Se va a asumir que las perturbaciones en los pesos son procesos aditivos de ruido blanco con valor medio igual a cero que satisfacen:

$$E\ [\Delta w_{ij}^{m}\ \Delta w_{i'j'}^{m'}] = \begin{cases} \sigma^2 & si\quad i=i'\quad y\quad j=j'\quad y\quad m=m' \\ 0 & en\quad otro\quad caso \end{cases} \tag{2.8}$$

es decir, se asume que las perturbaciones entre pesos distintos no están correlacionadas entre sí.

Como se aprecia en la expresión (2.6), es preciso evaluar dos términos para obtener la varianza, $E[\Delta y_i^m]$ y $E[(\Delta y_i^m)^2]$. A continuación, en la Sección 2.4.1 se procede a la evaluación de $E[\Delta y_i^m]$ y , en la Sección 2.4.2, al cálculo de la sensibilidad estadística a un patrón de entrada.

En el cálculo de las expresiones anteriormente citadas se hará uso de las leyes de la aritmética de valores esperados, es decir:

A1) El valor esperado de una suma de variables es igual a la suma de los valores esperados de dichas variables, es decir, $E[\sum X_i] = \sum E[X_i]$.

A2) El valor esperado del producto de una constante por una variable es igual al producto de la constante por el valor esperado de dicha variable, o sea, $E[kX] = kE[X]$.

A3) El valor esperado del producto de dos variables independientes X e Y es igual al producto de los valores esperados de dichas variables: $E[X \cdot Y] = E[X] \cdot E[Y]$.

2.3.1. Cálculo de $E[\Delta y_i^m]$

Para realizar el cálculo de $E[\Delta y_i^m]$ se hará uso de la siguiente proposición:

Proposición 2.1: Si $E[\Delta w_{ij}^m]=0\ \forall$ i,j,m entonces $E[\Delta y_i^m]=0\ \forall$ i,m.

Demostración 2.1: Se va a realizar por inducción sobre m.

- Para la capa 1:

Aplicando A1) y A2), en las neuronas pertenecientes a la primera capa (m=1) se tiene que:

$$\begin{aligned} E\ [\Delta y_i^{\,1}] &= E\ [\partial f_i^{\,1} \sum_{j=1}^{N_0} (y_j^{\,0} \Delta w_{ij}^{\,1})] \\ &= \partial f_i^{\,1} \sum_{j=1}^{N_0} y_j^{\,0}\ E\ [\Delta w_{ij}^{\,1}] \\ &= 0 \end{aligned} \tag{2.9}$$

puesto que $E[\Delta w_{ij}^{\,l}]=0\ \forall$ i,j,l.

- Para la capa m:

Asumiendo que $E[\Delta y_j^{\,m-1}]=0$ se tiene que, para las neuronas de la capa m:

$$\begin{aligned} E\ [\Delta y_i^{\,m}] &= E\ [\partial f_i^{\,m} \sum_{j=1}^{N_{m-1}} (y_j^{\,m-1} \Delta w_{ij}^{\,m} + w_{ij}^{\,m} \Delta y_j^{\,m-1})] \\ &= \partial f_i^{\,m} \sum_{j=1}^{N_{m-1}} (y_j^{\,m-1}\ E\ [\Delta w_{ij}^{\,m}] + w_{ij}^{\,m}\ E\ [\Delta y_j^{\,m-1}]) \\ &= 0 \end{aligned} \tag{2.10}$$

□

Así, finalmente queda demostrado que $E[\Delta y_i^{\,m}]=0$, para cualquier i y cualquier m, y por tanto $var(\Delta y_i^{\,m}) = E[(\Delta y_i^{\,m})^2]$. De esta forma, según (2.5), la sensibilidad se puede expresar como:

$$S_i^{\,m} = \lim_{\sigma \to 0} \frac{\sqrt{E\ [(\Delta y_i^{\,m})^2\ ]}}{\sigma} \tag{2.11}$$

2.3.2. Cálculo de la sensibilidad estadística de una neurona a un patrón de entrada

El objetivo en este apartado es calcular la expresión (2.11) usando el modelo aditivo que se ha introducido anteriormente. Para el cálculo de esta expresión, es necesario básicamente obtener la expresión de $E[(\Delta y_i^{\,m})^2]$, que no es tan trivial como en el caso de $E[\Delta y_i^{\,m}]$. Como se verá, aparecen términos cruzados no nulos (los términos relacionados con la covarianza) de forma que,

en general, los $E[\Delta y_j^m \Delta y_k^m]$ con $j \neq k$, no son cero. Por esta razón, a continuación se procederá al cálculo del valor de $E[\Delta y_j^m \Delta y_k^m]$ para cualquier valor de j,k,m con objeto de obtener la expresión de la sensibilidad.

Proposición 2.2: Para los elementos de la capa de entrada (m=0), se cumple que $E[\Delta y_j^0 \Delta y_k^0]=0 \ \forall j,k$.

Demostración 2.2:

La demostración de la Proposición 2.2 es trivial , puesto que las entradas a la red se suponen libres de error, es decir, $\Delta y_i^0 = 0 \ \forall i$.

□

Proposición 2.3: Definiendo C_{jk}^m como

$$C_{jk}^m \equiv \frac{E\ [\Delta y_j^m\ \Delta y_k^m\ ]}{\sigma^2} \tag{2.12}$$

el valor de este término $\forall$ m>0 se puede expresar como:

$$C_{jk}^m = \begin{cases} (\partial f_j^m)^2 \sum_{r=1}^{N_{m-1}} ((y_r^{m-1})^2 + w_{jr}^m \sum_{s=1}^{N_{m-1}} w_{js}^m\ C_{rs}^{m-1}) & si\ j=k \\ \partial f_j^m\ \partial f_k^m \sum_{r=1}^{N_{m-1}} w_{jr}^m \sum_{s=1}^{N_{m-1}} w_{ks}^m\ C_{rs}^{m-1} & en\ otro\ caso \end{cases} \tag{2.13}$$

Demostración 2.3:

Para demostrar esta proposición se va a proceder al cálculo de $E[\Delta y_j^m \Delta y_k^m] \ \forall j,k$ siendo m>0.

Al desarrollar $E[\Delta y_j^m \Delta y_k^m]$ se obtiene:

$$
\begin{aligned}
E[\Delta y_j^{\,m}\ \Delta y_k^{\,m}] &= E\,[\sum_{r=1}^{N_{m-1}} \partial f_j^{\,m}\ (y_r^{\,m-1}\ \Delta w_{jr}^{\,m} + w_{jr}^{\,m}\ \Delta y_r^{\,m-1})\cdot \\
&\quad \cdot \sum_{s=1}^{N_{m-1}} \partial f_k^{\,m}\ (y_s^{\,m-1}\ \Delta w_{ks}^{\,m} + w_{ks}^{\,m}\ \Delta y_s^{\,m-1})] \\
&= E\,[\sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} \partial f_j^{\,m}\ \partial f_k^{\,m}\ (y_r^{\,m-1}\ \Delta w_{jr}^{\,m} + w_{jr}^{\,m}\ \Delta y_r^{\,m-1})\cdot \\
&\quad \cdot (y_s^{\,m-1}\ \Delta w_{ks}^{\,m} + w_{ks}^{\,m}\ \Delta y_s^{\,m-1})] \\
&= \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} \partial f_j^{\,m}\ \partial f_k^{\,m}\ (y_r^{\,m-1} y_s^{\,m-1} E[\Delta w_{jr}^{\,m}\ \Delta w_{ks}^{\,m}] + \\
&\quad + w_{jr}^{\,m} w_{ks}^{\,m}\ E[\Delta y_r^{\,m-1}\ \Delta y_s^{\,m-1}] + y_r^{\,m-1} w_{ks}^{\,m}\ E[\Delta w_{jr}^{\,m}\ \Delta y_s^{\,m-1}] + \\
&\quad + y_s^{\,m-1} w_{jr}^{\,m}\ E[\Delta w_{ks}^{\,m}\ \Delta y_r^{\,m-1}]) \\
&= \partial f_j^{\,m}\ \partial f_k^{\,m} \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} (y_r^{\,m-1} y_s^{\,m-1} E[\Delta w_{jr}^{\,m}\ \Delta w_{ks}^{\,m}] + \\
&\quad + w_{jr}^{\,m} w_{ks}^{\,m}\ E[\Delta y_r^{\,m-1}\ \Delta y_s^{\,m-1}] + 0 + 0)
\end{aligned}
\tag{2.14}
$$

Los términos $E[\Delta w_{jr}^{\,m}\,\Delta y_s^{\,m-1}]$ y $E[\Delta w_{ks}^{\,m}\,\Delta y_r^{\,m-1}]$ son nulos ya que se puede aplicar la regla del producto de valores esperados A3) y la Proposición 2.1, al tratarse de variables independientes entre las que no existe correlación alguna, según se deduce aplicando la expresión (2.4) para $\Delta y_r^{\,m-1}$ y $\Delta y_s^{\,m-1}$.

En este punto, si se hace el cambio de variable indicado en (2.12) junto con la aplicación del modelo de desviación aditivo (2.8), se obtiene que, cuando j=k:

$$
E[(\Delta y_j^{\,m})^2] = (\partial f_j^{\,m})^2 \sum_{r=1}^{N_{m-1}} (\sigma^2 (y_r^{\,m-1})^2 + w_{jr}^{\,m} \sum_{s=1}^{N_{m-1}} w_{ks}^{\,m}\ \sigma^2 C_{rs}^{\,m-1}) \tag{2.15}
$$

y, en el caso de que $j \neq k$:

$$
E[\Delta y_j^{\,m}\ \Delta y_k^{\,m}] = \partial f_j^{\,m}\ \partial f_k^{\,m} \sum_{r=1}^{N_{m-1}} w_{jr}^{\,l} \sum_{s=1}^{N_{m-1}} w_{ks}^{\,m}\ \sigma^2 C_{rs}^{\,m-1} \tag{2.16}
$$

Así, dividiendo por σ^2 en ambos lados de las expresiones (2.15) y (2.16) se obtiene la expresión (2.13), tal como se quería demostrar. Nótese que esta expresión es recursiva y que de la Proposición 2.2 se obtiene el caso inicial, ya que de ahí se deduce que $C_{jk}^{\,0} = 0$

$\forall$j,k.

□

Corolario 2.4: La sensibilidad estadística a un patrón de entrada de una neurona i perteneciente a la capa m se puede expresar como:

$$S_i^m = \partial f_i^m \sqrt{\sum_{j=1}^{N_{m-1}} ((y_j^{m-1})^2 + w_{ij}^m \sum_{k=1}^{N_{m-1}} w_{ik}^m C_{jk}^{m-1})} \tag{2.17}$$

Demostración 2.4: Teniendo en cuenta la definición de los términos C_{jk}^{m} se tiene que $E[(\Delta y_i^m)^2]= \sigma^2 C_{ii}^m$, por lo que haciendo uso de la expresión (2.11) se obtiene que:

$$S_i^m = \lim_{\sigma \to 0} \frac{\sqrt{\sigma^2 C_{ii}^m}}{\sigma} = \sqrt{C_{ii}^m} \tag{2.18}$$

Aquí, al sustituir el valor de C_{ii}^1 obtenido a partir de (2.13) se deduce la expresión de la sensibilidad estadística cuando se usa un modelo aditivo de perturbaciones en los pesos.

□

Corolario 2.5: La sensibilidad estadística a un patrón de entrada de las neuronas de la primera capa oculta (m=1) para el modelo de desviación aditivo se puede expresar como:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{j=1}^{N_0} (y_j^0)^2} \tag{2.19}$$

Demostración 2.5: Si la expresión (2.17) se particulariza para m=1 y teniendo en cuenta que, por la Proposición 2.2, $C_{jk}^0 = 0$ $\forall$j,k, se obtiene la expresión (2.19).

□

Corolario 2.6: Los términos C_{jk}^{1} con $j \neq k$ en las neuronas de la primera capa oculta son nulos.

Demostración 2.6: Teniendo en cuenta la expresión (2.13) para el caso en que $j \neq k$ y $m=1$, y que $C_{jk}^{0} = 0 \ \forall j,k$ se deduce que $C_{jk}^{1} = 0$ si $j \neq k$.

□

Corolario 2.8: La sensibilidad estadística a un patrón de entrada de las neuronas de la segunda capa oculta (m=2) cuando se usa un modelo de desviación aditivo se puede expresar como:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{ij}^2 \ S_j^1)^2)} \qquad (2.20)$$

Demostración 2.5: Si se particulariza (2.17) para $m=2$ teniendo en cuenta que $C_{jk}^{1} = 0$ si $j \neq k$, tal como se desprende del Corolario 2.7, se obtiene la expresión (2.20).

□

Las expresiones (2.19) y (2.20) se cumplen siempre en cualquier perceptrón. La sensibilidad estadística en el caso de perceptrones con una única capa oculta se puede calcular de forma sencilla ya que, al no aparecer términos cruzados del tipo $C_{jk}^{m} \ \forall m=0,1$ siendo $j \neq k$, el valor de la sensibilidad estadística queda expresado en función de las salidas de la capa anterior y de las sensibilidades estadísticas de dichas neuronas. De esta manera, disminuye la memoria necesaria para almacenar los términos C_{jk}^{m} y se simplifica el cálculo de la sensibilidad estadística.

En la siguiente sección se van a obtener expresiones similares a las obtenidas en este apartado, pero considerando otro modelo de desviación, el modelo multiplicativo.

2.4. CÁLCULO DE LA SENSIBILIDAD ESTADÍSTICA USANDO UN MODELO DE DESVIACIÓN MULTIPLICATIVO

En el modelo multiplicativo se asume que las desviaciones en los pesos son proporcionales al valor de los mismos, es decir, cada peso w_{ij}^{m} es modificado tomando valores $(w_{ij}^{m})^{*} = w_{ij}^{m}(1 \pm \delta)$, presentando δ una desviación típica igual a σ. En este caso, se considera que las desviaciones tienen valor medio igual a cero y que satisfacen la siguiente condición:

$$E[\Delta w_{ij}^{m}\ \Delta w_{i'j'}^{m'}]=\begin{cases}(\sigma\ w_{ij}^{m})^2 & si\ \ i=i'\ \ y\ \ j=j'\ \ y\ \ m=m' \\ 0 & en\ \ otro\ \ caso\end{cases} \tag{2.21}$$

Las desviaciones de tipo multiplicativo suelen usarse para modelar la tolerancia analógica en los circuitos electrónicos. De esta forma, el parámetro σ representaría precisamente la tolerancia de los componentes analógicos.

De forma idéntica a como en el Apartado 2.4.1 se ha demostrado la Proposición 2.1, es inmediato demostrar para este modelo que se cumple que $E[\Delta y_i^m]=0$ para cualquier neurona i y cualquier capa m. También, usando los mismos argumentos que en la demostración de la Proposición 2.2 se prueba que para las neuronas de la capa de entrada se cumple que $E[\Delta y_j^0\ \Delta y_k^0]=0\ \forall j,k$. A continuación se va a obtener la expresión de la sensibilidad a un patrón de entrada de las neuronas de un MLP cuando se asumen desviaciones de tipo multiplicativo, para lo que previamente se probará la siguiente proposición.

Proposición 2.9: Definiendo C_{jk}^{m} como

$$C_{jk}^{m} \equiv \frac{E\ [\Delta y_j^{m}\ \Delta y_k^{m}\ ]}{\sigma^2} \tag{2.22}$$

el valor de este término, cuando se asume un modelo de desviación multiplicativo, se puede expresar para todo m mayor que 0 como:

$$C_{jk}^{m} = \begin{cases}(\partial f_j^{m})^2 \sum_{r=1}^{N_{m-1}} ((w_{jr}^{m} y_r^{m-1})^2 + w_{jr}^{m} \sum_{s=1}^{N_{m-1}} w_{js}^{m}\ C_{rs}^{m-1}) & si\ \ j=k \\ \partial f_j^{m}\ \partial f_k^{m} \sum_{r=1}^{N_{m-1}} w_{jr}^{m} \sum_{s=1}^{N_{m-1}} w_{ks}^{m}\ C_{rs}^{m-1} & en\ \ otro\ \ caso\end{cases} \tag{2.23}$$

Demostración 2.9: Los términos $E[\Delta y_j^m\ \Delta y_k^m]\ \forall j,k$ para una capa m>0 se calculan de forma idéntica a como se hizo al probar la Proposición 2.3. Si en la expresión (2.14) se sustituye el modelo de desviación multiplicativo se obtiene (2.23).

□

Corolario 2.10: Si se considera un modelo de desviación multiplicativo, la sensibilidad

a un determinado patrón de entrada de una neurona i perteneciente a la capa m de un perceptrón multicapa puede expresarse como:

$$S_i^m = \partial f_i^m \sqrt{\sum_{j=1}^{N_{m-1}} ((y_j^{m-1} w_{ij}^m)^2 + w_{ij}^m \sum_{k=1}^{N_{m-1}} w_{ik}^m \; C_{jk}^{m-1})} \qquad (2.24)$$

Demostración 2.10: Basta sustituir el valor correspondiente de C_{ii}^m, obtenido a partir de la Proposición 2.9, en la expresión de la sensibilidad (2.11) para deducir (2.24).

□

Corolario 2.11: La sensibilidad de las neuronas pertenecientes a la primera capa oculta, cuando se considera el modelo multiplicativo, puede calcularse según la siguiente expresión:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{j=1}^{N_0} (y_j^0 w_{ij}^1)^2} \qquad (2.25)$$

Demostración 2.11: Teniendo en cuenta que, por la Proposición 2.2, los términos C_{jk}^0 son nulos, y particularizando la expresión (2.24) para m=1, se obtiene (2.25).

□

Corolario 2.12: Los términos C_{jk}^1 con $j \neq k$ en las neuronas de la primera capa oculta son nulos.

Demostración 2.12: Teniendo en cuenta la expresión (2.23) para el caso en que $j \neq k$ y m=1, y que $C_{jk}^0 = 0 \;\; \forall j,k$ se deduce que $C_{jk}^1 = 0$ cuando $j \neq k$.

□

Corolario 2.13: La sensibilidad a un patrón de entrada de las neuronas de la segunda capa (m=2) se puede expresar como:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} (w_{ij}^2)^2 ((y_j^1)^2 + (S_j^1)^2)} \qquad (2.26)$$

Demostración 2.13: Si la expresión general de la sensibilidad para el modelo

multiplicativo (2.24) se particulariza para m=2 teniendo en cuenta que $C_{jk}^{1} = 0$ si $j \neq k$, tal como se dedujo en el Corolario 2.12, se obtiene la expresión (2.26).

□

2.5. SENSIBILIDAD ESTADÍSTICA EN MLPs CON UNA CAPA OCULTA

Las expresiones (2.17) y (2.24) permiten expresar la sensibilidad estadística de cualquier neurona dada, i, de cualquier capa, m, para un patrón de entrada determinado, usando los modelos de desviación aditivo o multiplicativo, respectivamente. En esta sección se obtienen de forma compacta las expresiones para perceptrones con una única capa oculta. Tales MLPs se caracterizan por estar constituidos por una capa de entrada, una capa oculta y una capa de salida, y se ha demostrado que son aproximadores universales [HOR89], por lo que centrar el estudio en tales perceptrones no supone restricción alguna desde el punto de vista de su aplicabilidad práctica aunque sí desde el punto de vista de sus prestaciones en cuanto a rapidez de entrenamiento, etc. En los casos donde los requisitos hagan necesario utilizar perceptrones de más capas habría que hacer uso de las expresiones generales que hemos deducido anteriormente. Aquí se han considerado estas expresiones en aras de una mayor legibilidad de la memoria. De esta forma, a partir de ahora sólo consideraremos MLPs con una capa oculta ya que, tal como se obtuvo en los apartados anteriores, el cálculo de la sensibilidad se hace sencillo porque los términos cruzados, C_{jk}^{m} con $j \neq k$, son nulos para las capas m=0 (capa de entrada) y m=1 (capa oculta), tal como se dedujo en la Proposición 2.2 y en los Corolarios 2.6 y 2.11. Por tanto, para evaluar la sensibilidad de las neuronas en un perceptrón de tres capas sólo es preciso almacenar el valor de las sensibilidades de las neuronas de la capa anterior, con el consecuente ahorro de tiempo de cómputo y capacidad de almacenamiento necesaria. Esto es importante a la hora de desarrollar los algoritmos propuestos en el siguiente capítulo, además de simplificar las expresiones correspondientes.

Asumiendo que la sensibilidad estadística de las neuronas de la capa de entrada es cero (recordemos que se trata de una capa cuya única misión es transmitir las entradas a la red y que no tiene pesos asociados), esto es, $S_i^0 = 0 \ \forall i$, podemos expresar la sensibilidad de las neuronas de un perceptrón de tres capas de una forma compacta:

$$S_i^m = \partial f_i^m \sqrt{\sum_{j=1}^{N_{m-1}} ((y_j^{m-1})^2 + (w_{ij}^m \ S_j^{m-1})^2)} \qquad m=1,2 \qquad (2.27)$$

en el caso de usar un modelo aditivo de desviación y,

$$S_i^m = \partial f_i^m \sqrt{\sum_{j=1}^{N_{m-1}} (w_{ij}^m)^2((y_j^{m-1})^2 + (S_j^{m-1})^2)} \qquad m=1,2 \tag{2.28}$$

en el caso de que el modelo de desviación considerado sea el multiplicativo.

A diferencia de [CHO92] se han obtenido expresiones explícitas para el cálculo de la sensibilidad estadística de cada nodo de la red. En [CHO92] se derivan expresiones para la sensibilidad estadística en notación matricial por lo que el estudio de las mismas resulta sumamente difícil, así como inferir propiedades a partir de dichas expresiones por no mostrar claramente la dependencia entre la sensibilidad de unas neuronas y aquellas a las que se encuentra conectada.

En la siguiente sección se analizará experimentalmente la validez de las expresiones anteriores, para terminar con las conclusiones del capítulo.

2.6. VALIDACIÓN DE LAS EXPRESIONES DE LA SENSIBILIDAD ESTADÍSTICA

Para validar las expresiones de sensibilidad obtenidas con cada uno de los modelos de desviación considerados, se han considerado una serie de MLPs entrenados, con distintas configuraciones de neuronas por capa y tipos de neuronas (con o sin sesgo y con función de activación sigmoidal o hiperbólica). Estos MLPs corresponden a dos aplicaciones diferentes: la aproximación funcional, y la predicción de series temporales.

En el caso de la aproximación funcional, se ha escogido un MLP que aproxima la función seno. Esta estructura recibe como entrada la variable independiente y proporciona el valor del seno en su salida. El predictor de la serie temporal en cambio, recibe como entradas los valores de una función, correspondientes a instantes consecutivos de tiempo, y proporciona el valor de dicha función para el instante siguiente. La serie temporal utilizada en este caso es la serie estocástica de Mackey-Glass. Ambos ejemplos son ampliamente conocidos y frecuentemente utilizados en distintas publicaciones [SUD94, WAN94].

Como de la expresión (2.5) se puede derivar que:

$$\sqrt{var(\Delta y_i^m)} = \sigma S_i^m \tag{2.29}$$

los pesos de las neuronas de los MLPs han sido perturbados fijando distintos valores de σ, y se ha calculado el valor de la parte izquierda de la expresión (2.29) que después se compara con el

valor obtenido al multiplicar σ por la sensibilidad estadística media, la cual se puede determinar a partir de la expresión (2.7) considerando las expresiones (2.27) o (2.28) según se trate de un modelo aditivo o multiplicativo de desviación, respectivamente.

2.6.1. Validación del modelo aditivo

Como primer ejemplo se ha tomado una estructura que aproxima la función seno. Este MLP se compone de 1 neurona de entrada, 11 neuronas en la capa oculta y 1 neurona de salida. En las Figuras 2.3, 2.4, 2.5 y 2.6 se muestran las curvas obtenidas para la desviación típica de salida calculadas teóricamente (σS) y mediante simulación. Los pesos de la estructura han sido modificados respecto de sus valores nominales siguiendo un modelo aditivo de desviación con una distribución normal de media cero y fijando σ a distintos valores. Para cada valor de σ considerado se han realizado 100 análisis, y se ha representado la desviación típica media del error de salida.

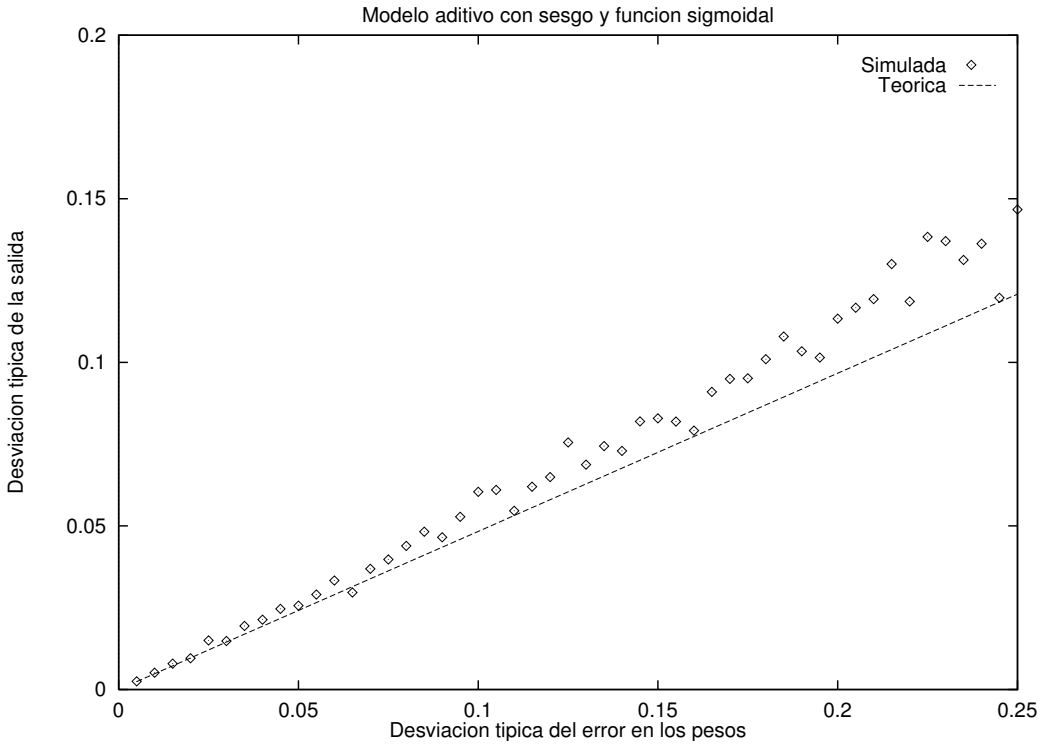

Figura 2.3. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

En la Figura 2.3 se ha tomado una estructura cuyas neuronas incorporan una entrada de sesgo y cuya función de activación es de tipo sigmoidal. En la Figura 2.4 las neuronas no tienen sesgo, mientras que en las Figuras 2.5 y 2.6 se muestran los resultados para estructuras con función de

activación de tipo hiperbólico con y sin sesgo, respectivamente. En el eje de las abscisas se muestra el valor de σ, mientras que las ordenadas representan el valor medio de $\sqrt{var(\Delta y)}$, constituyendo Δy la desviación de la salida respecto de la salida proporcionada por el perceptrón entrenado y con sus pesos no desviados.

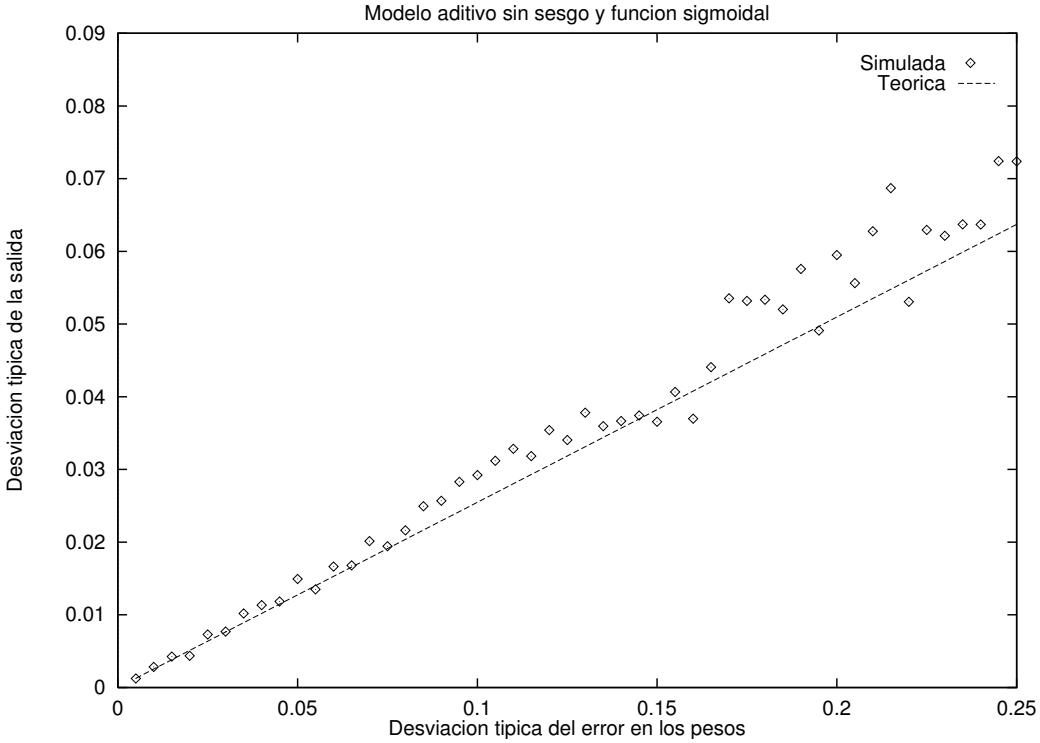

Figura 2.4. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

Puede observarse que la recta obtenida a partir de la medida propuesta se puede predecir de forma bastante aproximada el valor de la desviación típica de la salida obtenido como resultado de la simulación. Esta aproximación es tanto más precisa cuanto menor sea la perturbación considerada. Téngase en cuenta que se hicieron aproximaciones de primer orden para el cálculo de la sensibilidad estadística. Como la sensibilidad estadística está relacionada con la desviación típica de la salida del MLP perturbado (respecto de la salida que se obtendría sin perturbaciones), es obvio que un menor valor de ésta implica que dicha desviación es menor y, por tanto, más tolerante a faltas paramétricas.

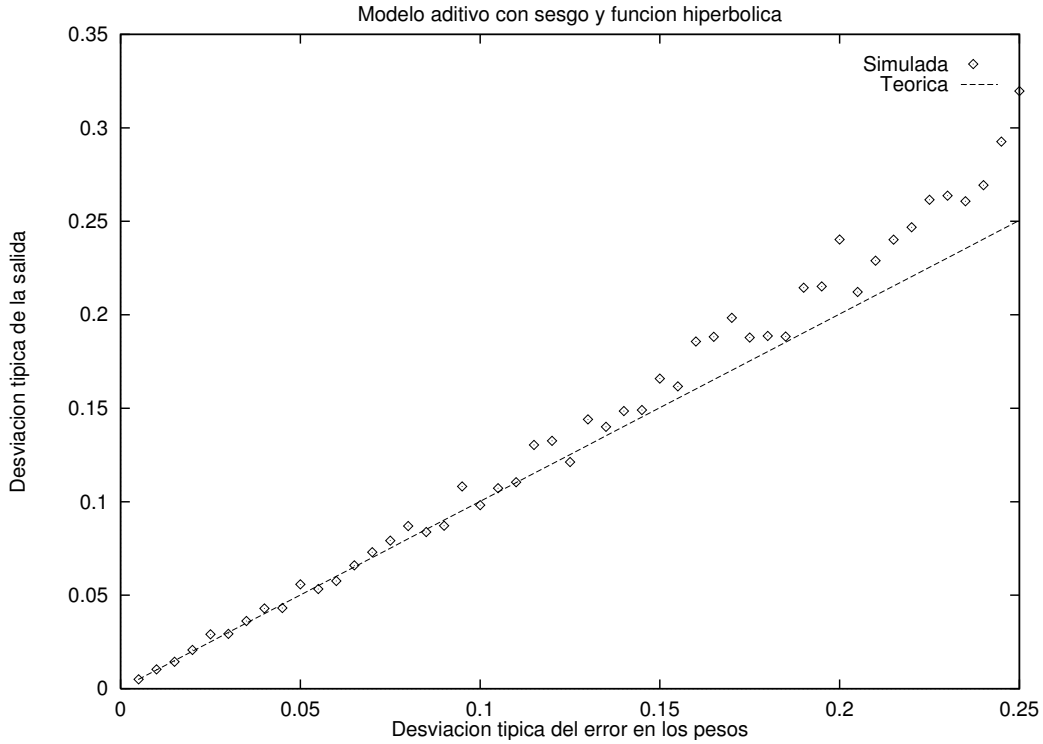

Figura 2.5. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

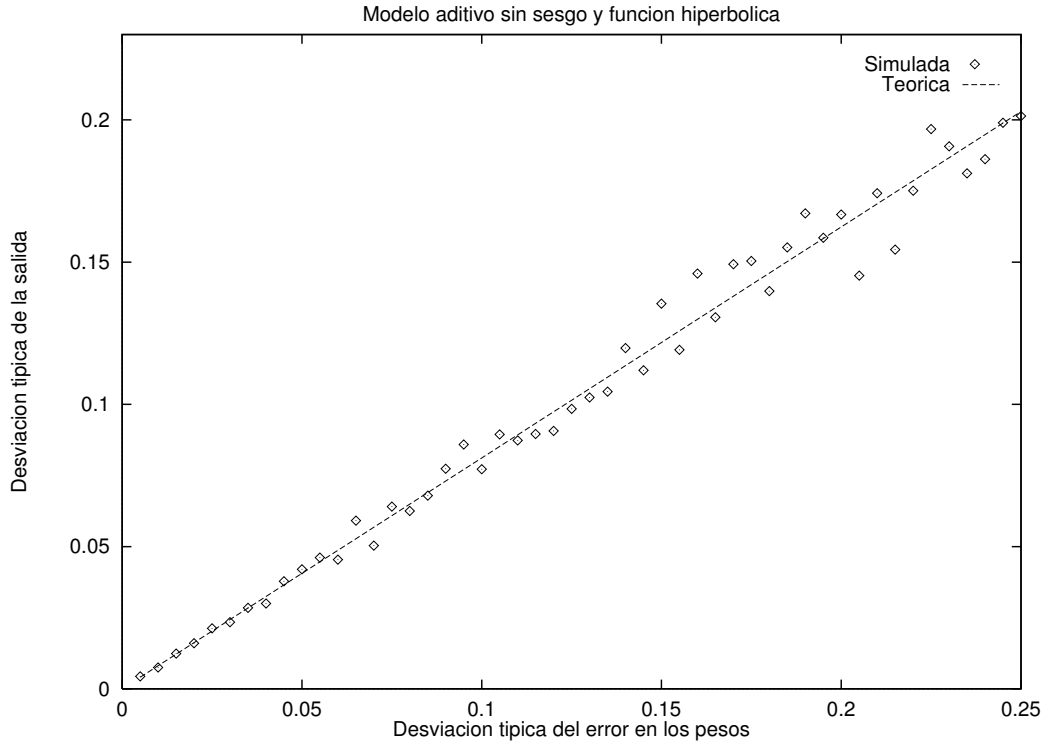

Figura 2.6. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

La siguiente figura (Figura 2.7) representa también la variación de la desviación típica media del error de salida respecto de las prestaciones nominales, pero en este caso se ha tomado un MLP que implementa un predictor de la serie temporal de Mackey_Glass [WAN94]. La estructura del perceptrón cuenta con 3 neuronas de entrada, 30 en la capa oculta y 1 neurona de salida. Las neuronas presentan entrada de sesgo y se ha considerado la función sigmoidal como función de activación.

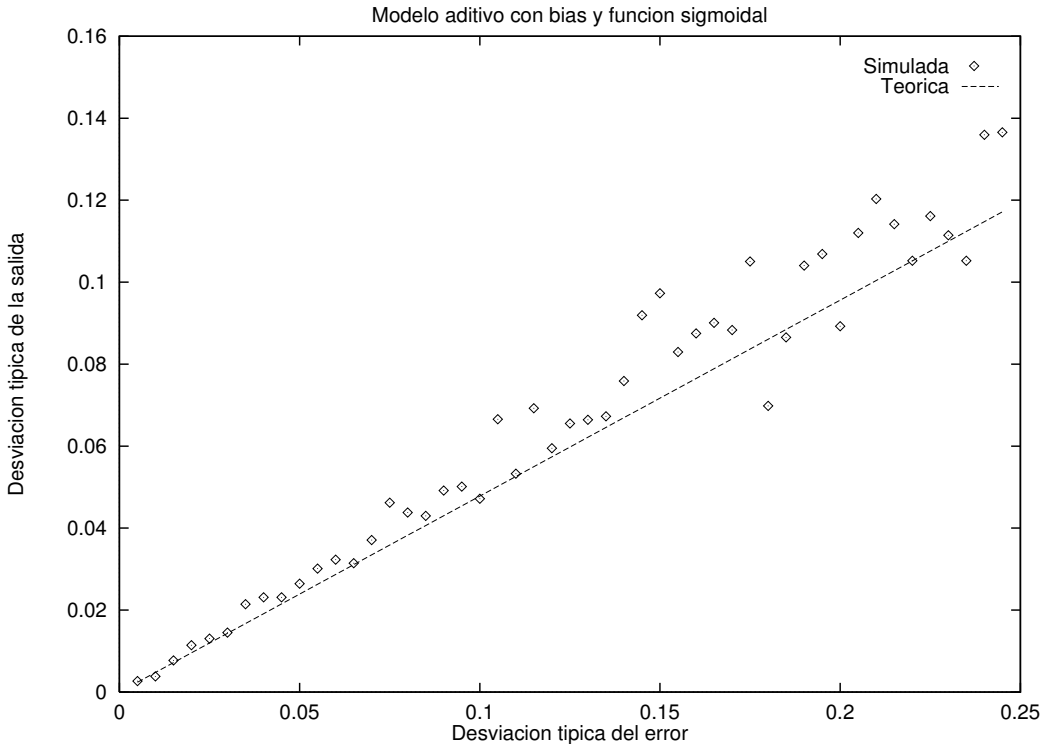

Figura 2.7. Validación de la expresión de la sensibilidad estadísticausando un predictor de la serie temporal de Mackey-Glass..

Se deduce a partir de las Figuras 2.3 a 2.7 que la expresión (2.27) constituye una medida fiable de la degradación de la red cuando los pesos se ven sometidos a perturbaciones de tipo aditivo. Como a partir de la expresión (2.29) se puede concluir que la sensibilidad estadística es directamente proporcional a la desviación típica del error de salida, queda probado que una menor sensibilidad estadística implica una mayor robustez de la red frente a cambios de tipo aditivo en los valores nominales de los pesos y, en consecuencia, una menor degradación de las prestaciones de la misma. Entendiendo que por prestaciones nos referimos al error cuadrático medio de salida (respecto de la salida deseada) obtenido tras el entrenamiento del perceptrón o, en su caso, al error de clasificación.

Para ilustrar que los MLPs con menor sensibilidad estadística son más robustos se han sometido

a desviaciones de tipo aditivo diversas configuraciones de pesos de un MLP implementando la función seno y se ha calculado el error cuadrático medio de salida (entre la salida deseada y la salida obtenida) para distintos valores de σ.

La Figura 2.8 muestra el error cuadrático medio de salida para 4 conjuntos de pesos que presentaban diferentes valores de sensibilidad estadística tras el proceso de entrenamiento. Puede apreciarse que aquellos conjuntos de pesos que presentan una mayor sensibilidad estadística se degradan en mayor cuantía que los que presentan una sensibilidad estadística menor. Incluso, aunque inicialmente presenten un error menor (por ejemplo, la curva con sensibilidad estadística igual a 1.00), a medida que crece σ sus prestaciones disminuyen de tal forma que otros con sensibilidad estadística menor (curva de sensibilidad igual a 0.25) puede responder mejor a las desviaciones, a pesar de que en un principio sus prestaciones fueran peores.

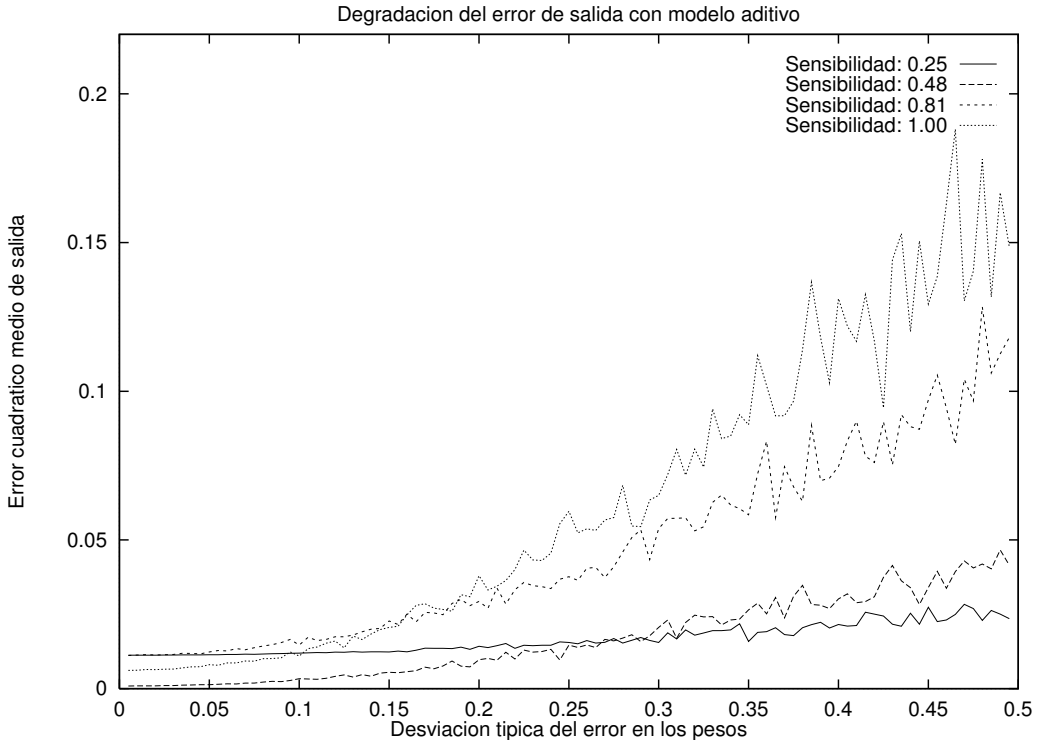

Figura 2.8. Degradación de las prestaciones frente a perturbaciones aditivas en los pesos usando un aproximador de la función seno.

Este hecho prueba que no todos los MLPs son igualmente tolerantes a fallos y que, a igualdad de prestaciones en cuanto a aprendizaje, interesa que la configuración de pesos obtenida tras el entrenamiento presente una sensibilidad estadística media baja, puesto que de esta manera mantendrá mejor su funcionalidad cuando los pesos sufran perturbaciones.

2.6.2. Validación del modelo multiplicativo

Hemos realizado pruebas similares a las del apartado anterior pero considerando ahora el modelo de desviación multiplicativo para modificar los valores de los pesos, y la expresión de la sensibilidad estadística para dicho modelo de desviación (2.28).

De nuevo, partiendo del MLP ya entrenado, sus pesos se han variado aleatoriamente según un modelo multiplicativo para un valor de σ prefijado. Las Figuras 2.9 a 2.12 muestran los valores teóricos y simulados de la desviación típica del error de salida para distintos valores de σ, para un MLP que aproxima la función seno y que presenta distintas configuraciones (usando la función sigmoidal o la hiperbólica, con entrada de sesgo o sin ella). En la Figura 2.13 se muestra una gráfica similar pero, en este caso, se refiere a un MLP que predice la serie temporal de Mackey-Glass con idéntica estructura a la indicada en el Apartado 2.6.1.

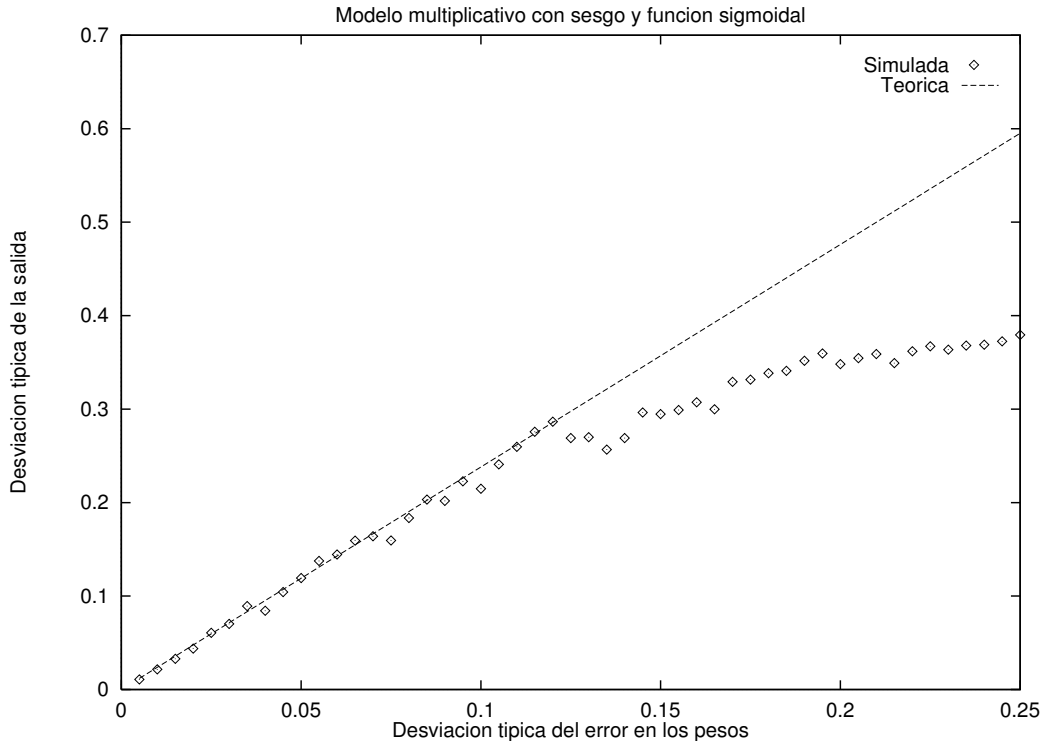

Figura 2.9. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

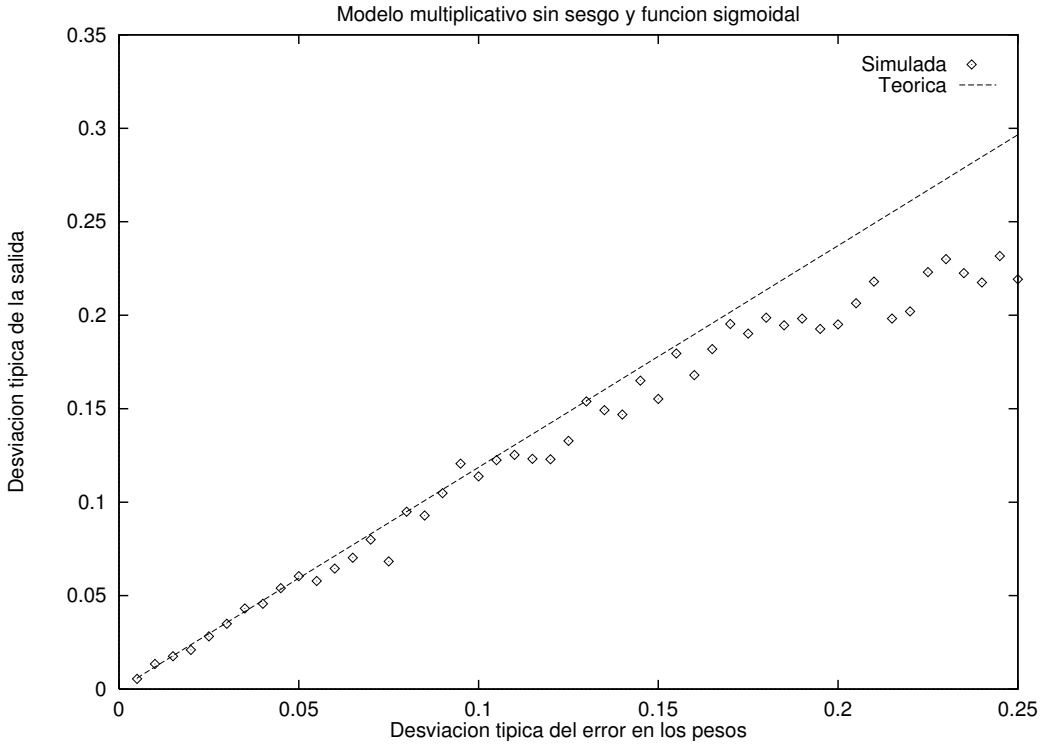

Figura 2.10. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

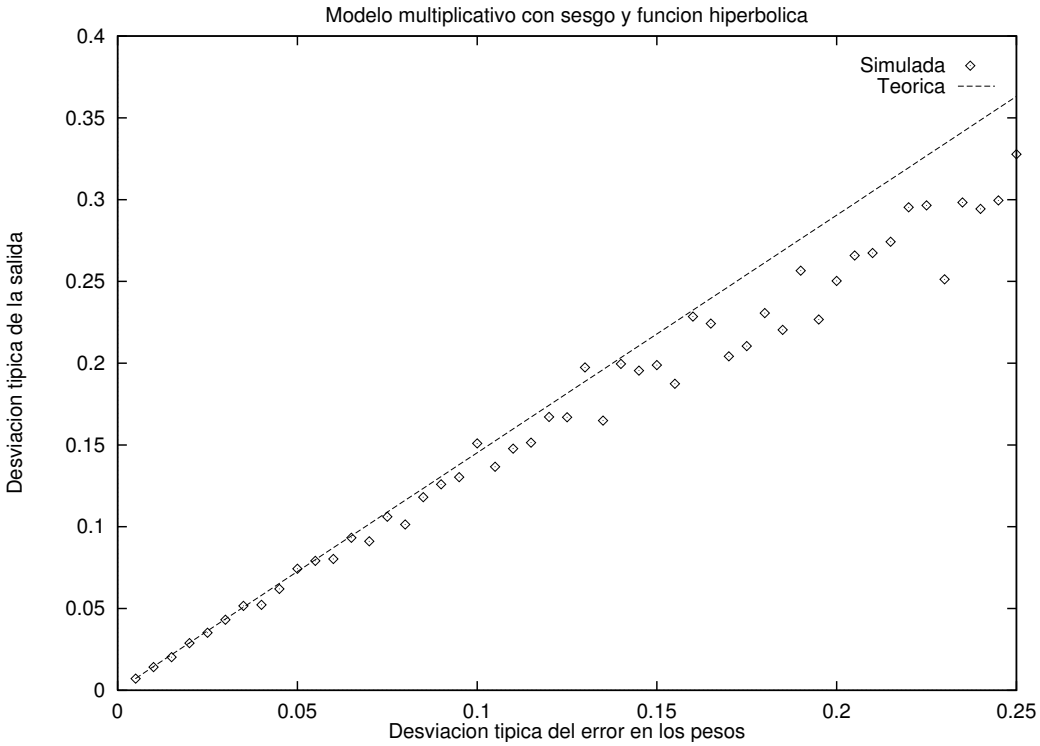

Figura 2.11. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

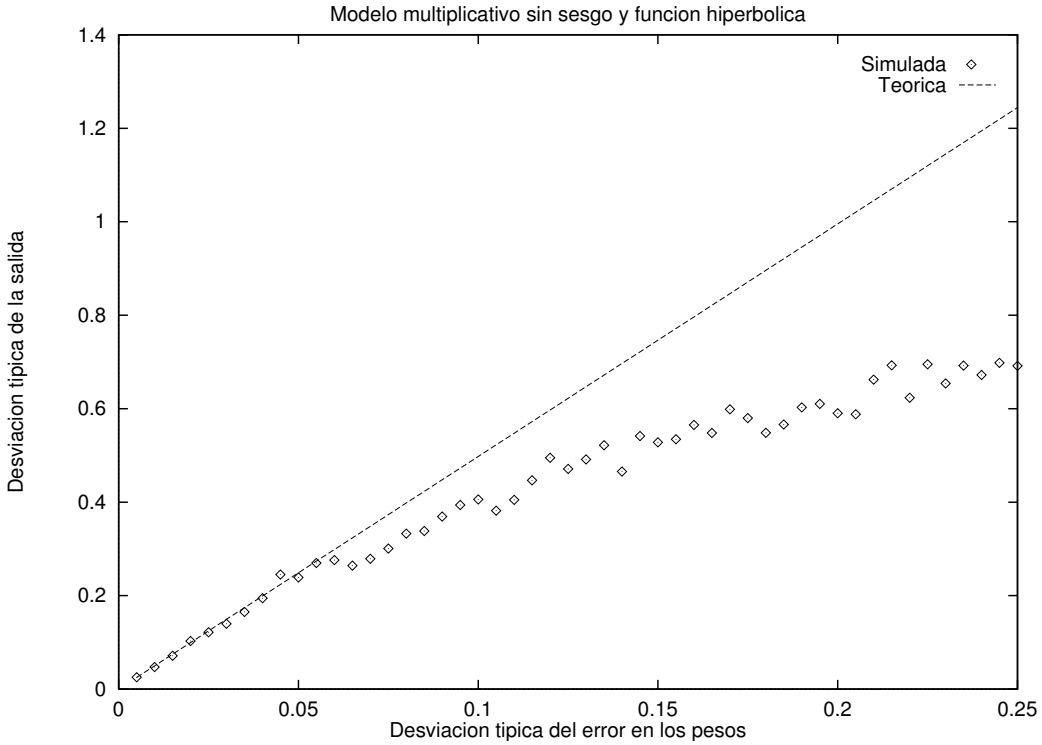

Figura 2.12. Validación de la expresión de la sensibilidad estadística usando un aproximador de la función seno.

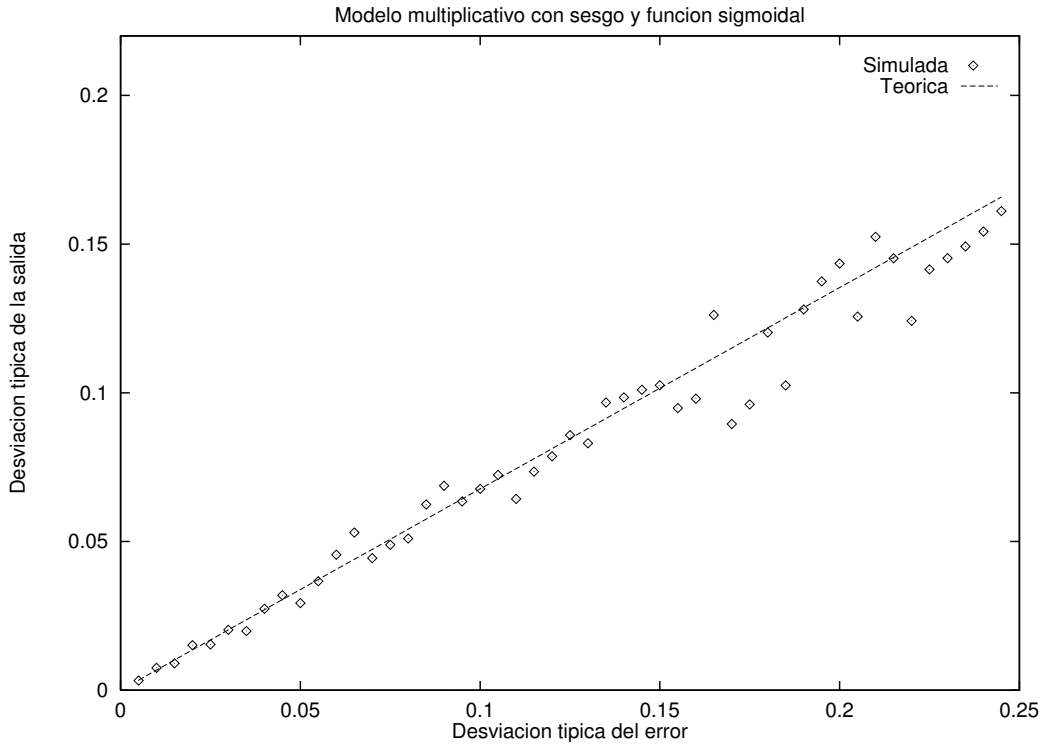

Figura 2.13. Validación de la expresión de la sensibilidad estadística usando un predictor de la serie de Mackey-Glass.

Cabe destacar que, como ya se ha mencionado antes, en el cálculo de la sensibilidad estadística se hicieron aproximaciones de primer orden, asumiendo que tales perturbaciones eran pequeñas. Por esta razón, esta medida predice de forma tanto más precisa la degradación del error de salida cuanto más pequeña sea la perturbación considerada. Este hecho, que se ha puesto de manifiesto en el apartado anterior, vuelve a observarse en las figuras correspondientes al modelo de desviación multiplicativo.

En los casos representados en las Figuras 2.9 y 2.12 la sensibilidad estadística media del MLP es bastante alta y por ello, dichos MLPs son fuertemente degradados al modificar los valores nominales de los pesos. Por esta razón puede observarse que la recta obtenida a partir de la expresión de la sensibilidad estadística (2.28) se separa más del resultado obtenido mediante simulación, comparado con el resto de los ejemplos mostrados: el error de salida (respecto de la salida del MLP sin perturbar) es tan alto que el valor predicho para el mismo deja de ser válido cuando la desviación típica de las perturbaciones multiplicativas es mayor que 0.1. En los otros casos considerados la aproximación es válida para perturbaciones con desviación típica hasta de 0.2 (Figura 2.10 y Figura 2.11) y de 0.25 (Figura 2.13). En cualquier caso se aprecia que la recta obtenida a partir de la expresión 2.28 constituye una cota superior, de forma que puede considerarse como una medida adecuada de la tolerancia a desviaciones.

Las perturbaciones de tipo multiplicativo permiten modelar la tolerancia analógica de los componentes electrónicos. De esta forma, en una estructura que presentase una sensibilidad de 0.35 y cuyas componentes de circuito tuvieran un 10% de tolerancia (σ=0.1), la desviación típica del error de salida sería aproximadamente de 0.035 (=0.1·0.35).

Debido a las razones anteriormente expuestas, una menor sensibilidad estadística debe implicar una mayor estabilidad del MLP cuando los pesos de sus neuronas son perturbados. Con objeto de reflejar este hecho las Figuras 2.14 y 2.15 muestran las curvas de error cuadrático de salida (respecto de la salida deseada) para conjuntos de pesos con distintos valores de sensibilidad estadística media. La Figura 2.14 se corresponde con el MLP que implementa a la función seno y la Figura 2.15 se refiere al predictor de la serie temporal. Se aprecia en ambas que las prestaciones se degradan más rápidamente al crecer la magnitud de la desviación en aquellos MLPs con un valor de sensibilidad estadística mayor.

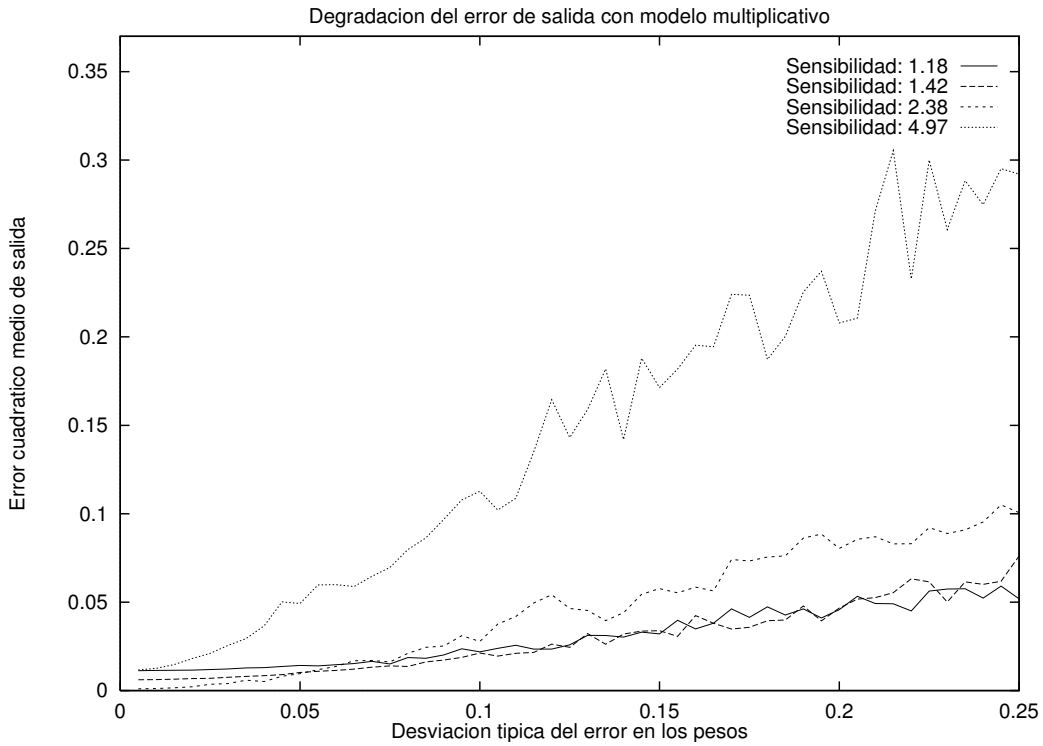

Figura 2.14. Degradación de las prestaciones frente a perturbaciones multiplicativas en los pesos usando un aproximador de la función seno.

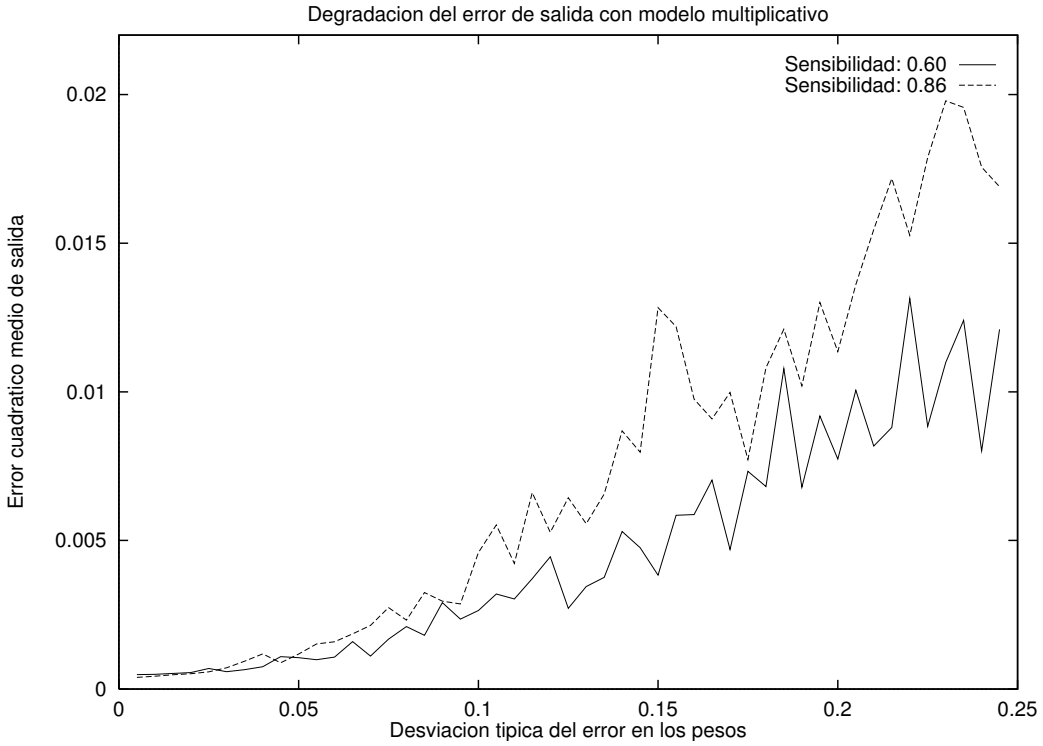

Figura 2.15. Degradación de las prestaciones frente a perturbaciones multiplicativas en los pesos usando un predictor de la serie de Mackey-Glass.

2.7. CONCLUSIONES

En este capítulo se han obtenido expresiones para medir la sensibilidad estadística de un perceptrón frente a desviaciones en los pesos. Esta medida indica cuánto varía la salida de una neurona al variar el valor de sus pesos para una entrada dada. Se define así la sensibilidad estadística para cada una de las neuronas, por consiguiente, se trata de una magnitud local en cuyo cálculo influyen las sensibilidades de las neuronas de las capas previas. De esta forma se dispone de una figura de mérito para estimar qué partes de la red son más sensibles, y cuales son más robustas frente a cambios en los valores efectivos de los pesos. En concreto resulta que, dado que la sensibilidad estadística de las neuronas se va propagando hacia las salidas de la red, es posible definir una medida global de la tolerancia a faltas paramétricas promediando la sensibilidad estadística de los nodos de salida para los distintos patrones de entrada.

Tal como se indica en [EDW97, EDW98C] los modelos de faltas catastróficas o de anclaje ("stuck-at") son poco realistas en el contexto de la tolerancia a fallos de redes neuronales artificiales, y tiene más sentido considerar las faltas de desviación para la mayor parte de las implementaciones existentes. Así pues, se han considerado dos modelos de desviación en los pesos, uno aditivo y otro multiplicativo, y se han obtenido las correspondientes expresiones de la sensibilidad estadística para cada uno de estos modelos.

Los resultados experimentales han mostrado la validez de las expresiones obtenidas y prueban que cuanto menor sea la sensibilidad estadística media mejor se mantienen las prestaciones del MLP. De esta forma, para obtener redes robustas interesa escoger aquellas que presenten una menor sensibilidad a igualdad de prestaciones en cuanto a error de salida tal como se propone en [CHO92]. Por tanto, la medida de la sensibilidad estadística media de la red constituye un criterio de selección entre distintas configuraciones de pesos.

Estas medidas ya fueron propuestas en [CHO92], sin embargo en dicho trabajo se utiliza notación matricial para expresar la sensibilidad estadística media de la red, quedando oculta cuál es la dependencia existente entre los distintos elementos que forman la red. Así, en este capítulo hemos identificado los factores que influyen en la sensibilidad estadística de cada neurona y la relación de la misma con la sensibilidad estadística de aquellas neuronas a las que se encuentra conectada.

De esta manera se puede observar a partir de las expresiones obtenidas para calcular la sensibilidad estadística de una neurona (expresiones (2.27) y (2.28)) que ésta es menor si las magnitudes de los pesos son pequeñas, lo cual corrobora la hipótesis indicada en [CHI94]. Este hecho se cumple en MLPs con una sola capa oculta debido a que los términos cruzados se anulan y los pesos aparecen siempre elevados al cuadrado, sin embargo, para MLPs con un número mayor de capas ocultas se puede apreciar en las expresiones (2.17) y (2.24) que también es necesario que

el resultado de la combinación de los pesos con los términos cruzados debe ser pequeña puesto que en este caso aparecen productos cruzados de pesos siendo esta relación no trivial. Ello indica que aunque obviamente, cuanto menores sean las magnitudes menor será la sensibilidad estadística, no es suficiente con limitar los valores de los pesos sino que es preciso que además el aprendizaje se distribuya de manera que se obtenga una configuración óptimamente tolerante a desviaciones en los mismos, como también se indica en [EDW98A]. Además, tal como se pone de manifiesto en [CHI94], restringir la magnitud de los pesos degrada las prestaciones de aprendizaje por lo que es necesario incrementar el número de neuronas.

También se deduce de estas expresiones que la sensibilidad estadística se hace cero cuando la derivada de la función de activación es nula, lo cual se corresponde a la situación en que la señal de la neurona está saturada. Este último hecho es razonable, puesto que si la señal de salida está saturada, una pequeña variación en los pesos la seguirá dejando saturada, por lo que la respuesta para el patrón presentado seguirá siendo la misma.

Se puede realizar un estudio más preciso de las distintas conexiones, puesto que la influencia de cada peso en la sensibilidad estadística de una determinada neurona puede ser desglosada de las expresiones obtenidas. Por ejemplo, si se eleva al cuadrado la expresión (2.28) correspondiente a la sensibilidad estadística de MLPs con una capa oculta considerando el modelo de desviación multiplicativo, se obtiene que:

$$(S_i^m)^2 = (\partial f_i^m)^2 \sum_{j=1}^{N_{m-1}} (w_{ij}^m)^2((y_j^{m-1})^2 + (S_j^{m-1})^2) \qquad m=1,2 \qquad (2.30)$$

De esta forma, la contribución de la conexión j al cuadrado de la sensibilidad estadística de la neurona i perteneciente a la capa m, puede expresarse como:

$$(s_{ij}^m)^2 = (\partial f_i^m)^2 \ (w_{ij}^m)^2 \ ((y_j^{m-1})^2 + (S_j^{m-1})^2) \qquad m=1,2 \qquad (2.31)$$

Así, por medio de (2.31) es posible estudiar particularmente la influencia de variaciones multiplicativas en el valor del peso w_{ij}^m en la salida de la neurona. Análogamente, en el caso de considerar desviaciones de tipo aditivo, la influencia del peso w_{ij}^m en la salida de la neurona puede expresarse como:

$$(s_{ij}^m)^2 = (\partial f_i^m)^2 \ ((y_j^{m-1})^2 + (w_{ij}^m \ S_j^{m-1})^2) \qquad m=1,2 \qquad (2.32)$$

Las expresiones (2.31) y (2.32) son más adecuadas para estudiar la influencia de los pesos en la tolerancia a fallos de MLPs que la saliencia ("saliency"). La saliencia es una medida relacionada con la capacidad de generalización y mide la suavidad de las curvas de error. Un valor pequeño de la saliencia implica cambios más suaves en la superficie de error, de forma que el riesgo de sobreentrenamiento ("overfitting") es menor. Experimentalmente se ha comprobado que la capacidad de generalización parece estar ligada con la tolerancia a fallos por lo que en [EDW95] se propone utilizar la saliencia media como medida de la robustez de una red. De este modo, en [EDW95, EDW97, EDW98C] se usa la diagonal de la matriz hessiana como medida de la tolerancia a faltas paramétricas aunque, como los mismos autores refieren, la relación entre generalización y tolerancia no está demasiado clara. La medida de la curvatura de error tiene cierta lógica, puesto que si esta curva tiene poca pendiente, una perturbación en los pesos provocará cambios suaves del error de salida.

La sensibilidad estadística media, en cambio, constituye una medida cuantitativa de los cambios producidos en las respuestas del MLP cuando sus pesos son perturbados, siendo así más indicada para evaluar la robustez de MLPs frente a tales perturbaciones. Si la sensibilidad estadística es muy pequeña, la salida del MLP perturbado es prácticamente la misma que cuando está libre de perturbaciones, lo que obviamente se corresponde a una situación en que el error de salida apenas varía y, por tanto, la pendiente de la curva de error debe ser suave. Este hecho será mostrado en el Capítulo 6, donde se pone de manifiesto la relación entre la sensibilidad estadística y la saliencia.

En el próximo capítulo, haciendo uso de las medidas de sensibilidad estadística introducidas se presentará un algoritmo de aprendizaje que tiene por objeto minimizar la sensibilidad estadística media de la red durante el proceso de entrenamiento (simultáneamente con la minimización del error de salida), obteniendo directamente configuraciones de pesos más tolerantes a desviaciones, sin afectar a las prestaciones del perceptrón y sin necesidad de incrementar el número de neuronas constituyentes para una estructura dada.

Capítulo 3

Minimización de la Sensibilidad Estadística Media: Regla ARR

En el capítulo anterior se obtuvieron expresiones que permiten calcular la sensibilidad estadística en cualquier nodo de un perceptrón multicapa. La medida de la sensibilidad estadística media proporciona un criterio de selección entre varias configuraciones de pesos según se indica en [CHO92]. Sin embargo, es interesante tratar de utilizar esta información durante el proceso de aprendizaje de forma que se obtengan configuraciones robustas, manteniendo el resto de las prestaciones respecto al error de aproximación o de clasificación. De esta forma, en el presente capítulo se propone un algoritmo de entrenamiento que incluye la sensibilidad estadística en la regla de aprendizaje para obtener configuraciones con una baja sensibilidad a desviaciones en los pesos evitándose, por tanto, incrementar el número de neuronas con objeto de aumentar la redundancia de la red para obtener así una mayor robustez. El algoritmo que proponemos puede considerarse como una técnica de regularización explícitamente especificada.

El capítulo se organiza de la siguiente manera: en primer lugar se obtienen nuevas reglas de aprendizaje para los modelos de desviación aditivo y multiplicativo, respectivamente. Estas reglas son usadas sobre diversos problemas para ser comparadas con la regla de retropropagación clásica (ARC). Por último, se muestran los resultados de simulación que permiten validar el procedimiento propuesto utilizando análisis de la T de Student y análisis de la varianza (ANOVA).

3.1. ALGORITMO DE RETROPROPAGACIÓN ROBUSTO

Usualmente los MLPs se entrenan utilizando procedimientos que prácticamente están todos basados en el llamado algoritmo de retropropagación. Dicho algoritmo se ha descrito en el Apartado 1.4. Durante el proceso de aprendizaje, una vez fijada una estructura para el MLP, un conjunto de N_p patrones de entrenamiento para los que se conoce la salida deseada d_i son presentados a la red. Estas entradas se propagan obteniéndose las correspondientes salidas y_i. El objetivo del aprendizaje es minimizar el error cuadrático de salida, $\varepsilon(p)$, calculado como:

$$\varepsilon(p) = \frac{1}{2} \sum_{i=1}^{N_M} (d_i - y_i)^2 \tag{3.1}$$

siendo N_M el número de neuronas de la capa de salida y p un patrón de entrada.

Para minimizar $\varepsilon(p)$, los valores de los pesos se modifican en sentido opuesto al gradiente de ε con respecto a w tras presentar cada patrón de entrada utilizando la siguiente regla de aprendizaje [LIP87, ANZ94]:

$$(w_{ij}^{m})^{t+1} = (w_{ij}^{m})^{t} + \alpha((-\nabla\varepsilon)_{ij}^{m})^{t} + \beta((w_{ij}^{m})^{t} - (w_{ij}^{m})^{t-1}) \tag{3.2}$$

El algoritmo de aprendizaje que utiliza la expresión (3.2) para actualizar los valores de los pesos será referido de ahora en adelante como “Algoritmo de Retropropagación Clásico” (ARC).

El proceso de entrenamiento se repite durante un cierto número de iteraciones llamadas épocas, donde cada época consiste en la presentación de los N_p patrones de entrenamiento al MLP. En el Capítulo 2 se ha visto que no todas las configuraciones de pesos presentan la misma tolerancia a desviaciones, aunque tengan similares prestaciones en cuanto a aprendizaje. En [CHO92] se propone evaluar la sensibilidad estadística media para distintas configuraciones de pesos con similar error cuadrático medio de salida y seleccionar aquella que presente la menor sensibilidad. No obstante, ello supone realizar múltiples entrenamientos sobre una misma estructura con el consiguiente incremento en el tiempo de cómputo.

La base del procedimiento propuesto consiste en añadir en la expresión (3.2) la sensibilidad estadística de las neuronas de salida como un parámetro más a minimizar, junto con el error cuadrático de salida.

En principio, es posible minimizar la sensibilidad estadística de la red, y por tanto maximizar su tolerancia a desviaciones en los pesos si se usa una regla similar a la expresión (3.2):

$$(w_{ij}^{m})^{t+1} = (w_{ij}^{m})^{t} + \gamma((-\nabla S)_{ij}^{m})^{t} \tag{3.3}$$

donde $(\nabla S)_{ij}^{m}$ es el gradiente de la sensibilidad estadística media respecto al peso w_{ij}^{m}, y al parámetro γ lo bautizamos con el nombre de "factor de robustez".

El problema es que la regla (3.3) es útil para minimizar la sensibilidad estadística media pero el perceptrón no aprende el conjunto de patrones de entrenamiento, por lo que es deseable combinar las reglas (3.2) y (3.3) de forma apropiada para que durante el proceso de aprendizaje, además de minimizar el error de salida se minimice simultáneamente la sensibilidad de la red.

Se trata, por consiguiente, de un problema de optimización multiobjetivo [GRA94] puesto que hay que minimizar dos magnitudes conjuntamente, el error cuadrático y la sensibilidad estadística de salida. Ambos objetivos pueden estar en conflicto puesto que la minimización de uno de ellos podría implicar el aumento del otro [GRA94, BER95]. Una posible solución para remediar este problema es jerarquizar las magnitudes a minimizar, primando una de ellas frente a la otra. Teniendo en cuenta que el principal objetivo sigue siendo que la red aprenda, se puede ponderar adecuadamente el factor de robustez para que el término relacionado con la sensibilidad sea menor que los relacionados con el aprendizaje y de esta manera se combinan entre sí en la siguiente expresión:

$$(w_{ij}^{m})^{t+1} = (w_{ij}^{m})^{t} + \alpha((-\nabla\varepsilon)_{ij}^{m})^{t} + \beta((w_{ij}^{m})^{t} - (w_{ij}^{m})^{t-1}) + \gamma((-\nabla S)_{ij}^{m})^{t} \tag{3.4}$$

Experimentalmente hemos encontrado que un valor adecuado para γ en cada época viene dado por:

$$\gamma = K\ \varepsilon\ S \tag{3.5}$$

donde ε y S son el error cuadrático medio (ECM) y la sensibilidad estadística media (SSM) de la red, tal como se definió en (2.7), obtenidos con la época anterior, y K es un factor constante usado para ponderar el factor de robustez. Se puede observar que la magnitud de este parámetro cambia dinámicamente en función de ε y S, de forma que cuando disminuye el error o la sensibilidad estadística, disminuye su influencia, permitiendo así la convergencia del algoritmo. Con el valor de γ definido de esta manera, si el error disminuye también lo hace proporcionalmente la influencia del factor de robustez manteniendo así la jerarquía deseada. Por otro lado, γ aumenta con la sensibilidad haciendo que este factor cobre importancia. Así pues, si el error crece en beneficio de la sensibilidad o viceversa, el factor γ trata de equilibrar la situación durante el entrenamiento.

El algoritmo propuesto, al cual llamaremos "Algoritmo de Retropropagación Robusto" (ARR), y que usa la regla (3.4), equivale al algoritmo clásico ARC si el factor de robustez se hace cero. Puede verse este nuevo término como un estabilizador y la nueva regla como una técnica de regularización explícitamente especificada [MAO93]. El problema a resolver a continuación consiste en el cálculo del gradiente de la sensibilidad estadística de salida para cada uno de los modelos de desviación considerados. Únicamente se van a presentar las expresiones para MLPs de tres capas ya que, por las razones expuestas en el capítulo anterior, esto no supone una pérdida de generalidad y en cambio simplifica en gran medida las expresiones obtenidas puesto que no es necesario tener en cuenta los términos cruzados siendo también más fácil analizar las contribuciones que aparecen en dichas expresiones (Apartado 2.6). Con ello se consigue una mayor claridad en la exposición y en el estudio de las propiedades de la ley de aprendizaje de cara a aumentar la tolerancia de la red.

3.2. CÁLCULO DEL GRADIENTE DE LA SENSIBILIDAD ESTADÍSTICA

De forma general, tal como se desprende de las expresiones (2.17) y (2.24), la sensibilidad estadística de una neurona i perteneciente a la capa m para los dos modelos de desviación considerados puede expresarse como el producto de dos términos: la derivada de la función de activación (∂f_i^m) y una raíz cuadrada (R_i^m),

$$S_i^m = \partial f_i^m \; R_i^m \tag{3.6}$$

El cálculo del gradiente implica obtener la derivada de la sensibilidad estadística media de la red, S, respecto de cada uno de los pesos, w_{ij}^m:

$$(\nabla S)_{ij}^m = \frac{dS}{dw_{ij}^m} \tag{3.7}$$

donde SSM viene dada por la expresión (2.7).

Por tanto, en general van a aparecer dos términos. Así, al calcular el gradiente para un peso j de una neurona i de la capa de salida (m=M) para un patrón de entrada dado, se obtiene:

$$(\nabla S)_{ij}^M \propto \frac{dS_i^M}{dw_{ij}^M} = \partial f_i^M \; \frac{\partial R_i^M}{\partial w_{ij}^M} + R_i^M \; \frac{\partial(\partial f_i^M)}{\partial w_{ij}^M} \tag{3.8}$$

Para el resto de las capas se siguen obteniendo expresiones similares en cada una de las neuronas en las que claramente se distinguen estos dos términos.

El primer término de la expresión (3.8), $\partial f_i^M \frac{\partial R_i^M}{\partial w_{ij}^M}$ trata de hacer pequeña la magnitud de los

pesos. La derivada de la función de activación, ∂f_i^m, es siempre positiva o nula por ser la función de activación creciente mientras que la derivada de R_i^M es un factor directamente proporcional al valor del peso w_{ij}^M, tal como se apreciará en las expresiones que se obtendrán al calcular el gradiente. Como los pesos se ajustan en sentido inverso a dicho gradiente, la contribución de este término es la de restar una cantidad proporcional a la magnitud del peso haciendo disminuir dicha magnitud.

El segundo término de la expresión (3.8), $R_i^M \frac{\partial(\partial f_i^M)}{\partial w_{ij}^M}$, trata de saturar la señal de salida . Al

ser R_i^M un valor siempre positivo, la segunda derivada de la función de activación es la que va a caracterizar la tendencia de este segundo factor. En las zonas de saturación o en las que la función es aproximadamente lineal, la segunda derivada es prácticamente nula por lo que en tales casos este término no va a afectar. En cambio, en las zonas cercanas a la zona de saturación, donde la función de activación es cóncava o convexa, la segunda derivada va a ser positiva o negativa, respectivamente. En estos casos, este segundo término tiende a variar la magnitud del peso de forma que la neurona proporcione una salida saturada y por tanto su primera derivada sea nula; en este caso la sensibilidad estadística se haría igual a cero tal como se deduce de las expresiones de la misma ((2.17) y (2.24)).

El cálculo exacto del gradiente implica realizar la modificación de los pesos al final de cada época [ANZ94], sumando las modificaciones obtenidas para cada patrón de test y adaptando los pesos tras haber presentado todos los patrones de entrenamiento. Sin embargo, al igual que suele hacerse en el algoritmo clásico los pesos se ajustan cada vez que se presenta un patrón de entrada. Por esta razón, la influencia del segundo término de la expresión (3.8) puede ser contradictoria para distintos patrones de entrada, ya que para unos tratará de modificar el valor de los pesos de forma que la salida esté saturada en un sentido, mientras que para otros tratará de modificarlos en sentido opuesto; incluso puede entrar en conflicto con el primer término de (3.8) ya que para saturar la señal de salida puede ser necesario incrementar considerablemente la magnitud de los pesos. Este hecho se ha comprobado experimentalmente considerando ambos términos en el cálculo del gradiente, apreciándose una clara degradación de las prestaciones en cuanto a error y sensibilidad estadística, puesto que el algoritmo no converge. Conseguir que la señal esté saturada a un valor extremo para cada patrón de entrada no es posible salvo en casos excepcionales (por ejemplo, un clasificador). Debido a las razones expuestas, se va a despreciar este segundo término, es decir, se va a suponer que la derivada de la función de activación, ∂f_i^m

es una constante para todo i y todo m. Esta aproximación equivale a ajustar la función de activación por una función lineal con saturación y permite además agilizar los cálculos.

Teniendo en cuenta esta aproximación, se va a proceder al cálculo del gradiente de la sensibilidad para MLPs de tres capas considerando los dos modelos de desviación propuestos.

3.2.1. Cálculo del gradiente usando el modelo aditivo

Las expresiones de la sensibilidad para este modelo vienen dadas por:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{j=1}^{N_0} (y_j^0)^2} \tag{3.9}$$

para las neuronas de la capa oculta y por:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{ij}^2 S_j^1)^2)} \tag{3.10}$$

para las neuronas de salida.

Se trata de obtener el gradiente de la sensibilidad estadística media instantánea de las neuronas de salida en cada iteración, la cual viene dada por la expresión:

$$S = \frac{1}{N_2} \sum_{r=1}^{N_2} S_r^2 \tag{3.11}$$

donde $S_r^2(p)$ es la sensibilidad estadística de la neurona r perteneciente a la capa de salida (m=2) para un determinado patrón de entrada. Por otra parte, como el término que antecede a la suma $(1/N_2)$ es una constante, dicho término no va a aparecer explícitamente en las expresiones, ya que se puede considerar incluido en el factor de robustez γ.

a) Cálculo del gradiente con respecto a los pesos de las neuronas de salida

Considerando un peso w_{ik}^2 que conecta una neurona de salida i con una neurona k de la capa oculta, se tiene que:

$$
\begin{aligned}
(\nabla S)_{ik}^2 &= \frac{d}{dw_{ik}^2}\sum_{r=1}^{N_2} S_r^2 = \frac{dS_i^2}{dw_{ik}^2} = \partial f_i^2 \frac{dR_i^2}{dw_{ik}^2} \\
&= \partial f_i^2 \frac{d}{dw_{ik}^2}\sqrt{\sum_{j=1}^{N_1}((y_j^1)^2 + (w_{ij}^2 S_j^1)^2)} \\
&= \partial f_i^2 \frac{w_{ik}^2 (S_k^1)^2}{\sqrt{\sum_{j=1}^{N_1}((y_j^1)^2 + (w_{ij}^2 S_j^1)^2)}} \\
&= \frac{\partial f_i^2}{R_i^2} w_{ik}^2 (S_k^1)^2 = \frac{(\partial f_i^2)^2}{S_i^2} w_{ik}^2 (S_k^1)^2
\end{aligned}
\tag{3.12}
$$

Nótese que cuando un término aparece elevado al cuadrado, dicho término se escribe entre paréntesis para evitar la confusión con un superíndice.

Se observa que en el desarrollo anterior sólo hay que considerar la contribución del peso w_{ik}^2 a la neurona de salida a la cual pertenece, y el valor de dicho peso será modificado en una magnitud proporcional a su magnitud y al cuadrado de la sensibilidad de la neurona de la capa anterior a la que se encuentra conectado (neurona k de la capa 1).

Por tanto, los pesos de las neuronas de la capa de salida se modifican de acuerdo con la expresión (3.12) con objeto de disminuir la sensibilidad estadística de las neuronas de salida. Esta modificación sumada a la correspondiente al algoritmo clásico de aprendizaje permitirá que el peso se actualice de forma que también se minimice el error de salida. A continuación se va a calcular el gradiente de la sensibilidad de salida respecto de los pesos de la primera capa oculta.

b) Cálculo del gradiente con respecto a los pesos de las neuronas de la capa oculta.

En este caso, hay que tener en cuenta que el valor de los pesos de las neuronas de la capa oculta influyen sobre todas las neuronas de salida puesto que se encuentran conectadas entre sí. De esta manera, una modificación en el valor de un peso de una neurona de la capa oculta afectará a la sensibilidad de todas las neuronas de salida.

Considerando un peso w_{ik}^1 que conecta la neurona i de la capa oculta con la entrada k, se va a proceder al cálculo del gradiente de la sensibilidad de la red respecto de los pesos de la capa

oculta de forma similar a como se ha realizado en el caso de los pesos pertenecientes a las neuronas de la capa de salida:

$$
\begin{aligned}
(\nabla S)^1_{ik} &= \frac{d}{dw^1_{ik}} \left(\sum_{r=1}^{N_2} S_r^2\right) = \sum_{r=1}^{N_2} \partial f_r^2 \; \frac{dR_r^2}{dw^1_{ik}} \\
&= \sum_{r=1}^{N_2} \partial f_r^2 \; \frac{d}{dw^1_{ik}} \sqrt{\sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{rj}^2 S_j^1)^2)} \\
&= \sum_{r=1}^{N_2} \partial f_r^2 \; \frac{\sum_{j=1}^{N_1} \left(y_j^1 \; \frac{dy_j^1}{dw^1_{ik}} + (w_{rj}^2)^2 \; S_j^1 \; \frac{dS_j^1}{dw^1_{ik}} \right)}{\sqrt{\sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{rj}^2 S_j^1)^2)}} \\
&= \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2} \; (y_i^1 \; \frac{dy_i^1}{dw^1_{ik}} + 0) = y_i^1 \; \frac{dy_i^1}{dz_i^1} \frac{dz_i^1}{dw^1_{ik}} \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2} \\
&= \partial f_i^1 \; y_k^0 \; y_i^1 \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2}
\end{aligned}
\qquad (3.13)
$$

En el desarrollo de la expresión (3.13) hay que tener en cuenta que, a partir de (3.9) se deduce claramente que S_j^1 no depende de w_{ik}^1 si se considera ∂f_i^1 constante y por ello se hace igual a cero.

En resumen, los pesos de las neuronas deben ser modificados según indican las expresiones (3.12) y (3.13) en aras de optimizar la tolerancia a fallos cuando se considera un modelo aditivo de desviaciones en los pesos. Nótese que este procedimiento constituye también un "algoritmo de retropropagación" pero en este se trata de minimizar la sensibilidad de salida empezando por las neuronas de salida y propagando el gradiente hacia las entradas a la red.

3.2.2. Cálculo del gradiente usando el modelo multiplicativo

Las expresiones de la sensibilidad estadística para este modelo difieren de las correspondientes al modelo de desviación aditivo principalmente en que las entradas a las neuronas van multiplicadas por los pesos mediante los que van ponderadas dichas entradas. Estas expresiones vienen dadas por:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{j=1}^{N_0} (y_j^0 \; w_{ij}^1)^2} \tag{3.14}$$

para las neuronas de la primera capa y por:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} (w_{ij}^2)^2 \; ((y_j^1)^2 + (S_j^1)^2)} \tag{3.15}$$

para la neurona de salida en MLPs con una capa oculta.

a) Cálculo del gradiente con respecto a los pesos de las neuronas de salida

Considerando un peso w_{ik}^2 que conecta la neurona de salida i con la neurona k de la capa oculta, se obtiene:

$$\begin{aligned}
(\nabla S)_{ik}^2 &= \frac{d}{dw_{ik}^2} \sum_{r=1}^{N_2} S_r^2 = \frac{dS_i^2}{dw_{ik}^2} = \partial f_i^2 \; \frac{dR_i^2}{dw_{ik}^2} \\
&= \partial f_i^2 \; \frac{d}{dw_{ik}^2} \sqrt{\sum_{j=1}^{N_1} (w_{ij}^2)^2 \; ((y_j^1)^2 + (S_j^1)^2)} \\
&= \partial f_i^2 \; \frac{w_{ik}^2((y_k^1)^2 + (S_k^1)^2)}{\sqrt{\sum_{j=1}^{N_1} (w_{ij}^2)^2 \; ((y_j^1)^2 + (S_j^1)^2)}} \\
&= \frac{\partial f_i^2}{R_i^2} \; w_{ik}^2 \; ((y_k^1)^2 + (S_k^1)^2) = \frac{(\partial f_i^2)^2}{S_i^2} \; w_{ik}^2((y_k^1)^2 + (S_k^1)^2)
\end{aligned} \tag{3.16}$$

El valor obtenido en el caso del modelo multiplicativo es muy similar al que se obtuvo en el caso del modelo aditivo (3.12). La principal diferencia, se encuentra en la contribución de la salida de la neurona y_k^1 de la capa oculta a la que va conectada por medio del peso w_{ik}^2, además de la sensibilidad estadística S_k^1 de dicha neurona, como sucedía en el caso de desviaciones aditivas.

b) Cálculo del gradiente con respecto a los pesos de las neuronas de la capa oculta.

Los pesos de las neuronas pertenecientes a la capa oculta influyen en la sensibilidad de todas las neuronas de salida, por lo que, en la expresión del gradiente de la sensibilidad de la red respecto de tales pesos, aparecerá su contribución a cada neurona de salida, de forma similar a como se obtuvo en (3.13). No obstante, debido a la expresión de la sensibilidad para el caso de desviación multiplicativo en las neuronas de la capa oculta, hay dependencia de S_i^1 respecto de w_{ik}^1 tal como se aprecia en (3.14). Por tanto, suponiendo incluso que ∂f_i^1 es constante se obtiene la contribución explícita del peso w_{ik}^1 en cada neurona de salida mediante los pesos w_{ri}^2 que conectan a la neurona i de la capa oculta con cada neurona r de salida.

Considerando un peso w_{ik}^1 de una neurona de la capa oculta i mediante el cual se conecta a la entrada k, se debe calcular la siguiente expresión:

$$
\begin{aligned}
(\nabla S)_{ik}^1 &= \frac{d}{dw_{ik}^1}\left(\sum_{r=1}^{N_2} S_r^2\right) = \sum_{r=1}^{N_2} \partial f_r^2 \frac{dR_r^2}{dw_{ik}^1} \\
&= \sum_{r=1}^{N_2} \partial f_r^2 \frac{d}{dw_{ik}^1} \sqrt{\sum_{j=1}^{N_1} (w_{rj}^2)^2 \ ((y_j^1)^2 + (S_j^1)^2)} \\
&= \sum_{r=1}^{N_2} \partial f_r^2 \frac{\sum_{j=1}^{N_1} (w_{rj}^2)^2 \left(y_j^1 \frac{dy_j^1}{dw_{ik}^1} + S_j^1 \frac{dS_j^1}{dw_{ik}^1} \right)}{\sqrt{\sum_{j=1}^{N_1} ((w_{rj}^2)^2 \ ((y_j^1)^2 + (S_j^1)^2)}} \\
&= \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2} (w_{ri}^2)^2 \ (y_i^1 \frac{dy_i^1}{dw_{ik}^1} + S_i^1 \frac{dS_i^1}{dw_{ik}^1}) \\
&= \left(y_i^1 \ \partial f_i^1 \ y_k^0 + S_i^1 \partial f_i^1 \frac{w_{ik}^1 (y_k^0)^2}{\sqrt{\sum_{j=1}^{N_0} (y_j^0 \ w_{ij}^1)^2}} \right) \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2} (w_{ri}^2)^2 \\
&= \partial f_i^1 \ y_k^0 \ (y_i^1 + w_{ik}^1 \ \partial f_i^1 \ y_k^0) \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2} (w_{ri}^2)^2
\end{aligned}
\qquad (3.17)
$$

En resumen, los pesos de las neuronas deben ser modificados según indican las expresiones (3.16) y (3.17) para optimizar la tolerancia a faltas de desviación cuando se considera el modelo multiplicativo. Sustituyendo estos valores en la regla (3.4) debidamente ponderados mediante el

factor de robustez γ obtenemos el algoritmo de aprendizaje robusto ARR.

3.3. COMPLEJIDAD DEL ALGORITMO

La Tabla 3.1 resume las expresiones del gradiente instantáneo de la sensibilidad estadística obtenidas en el caso de los modelos de desviación aditivo y multiplicativo.

Nótese que, por ser creciente la función de activación, su derivada es positiva y por tanto la sensibilidad es una magnitud positiva. Por esta razón sería equivalente minimizar el cuadrado de la sensibilidad en cuyo caso se obtienen expresiones algo más sencillas para el gradiente. Esto se debe a que al elevar al cuadrado la sensibilidad estadística, desaparece la raíz cuadrada y por tanto, dicha raíz no aparece en el denominador de la expresión del gradiente.

Tabla 3.1. Valores del gradiente de la sensibilidad.

Peso w_{ik}^m	Modelo aditivo	Modelo multiplicativo
Capa de salida (m=2)	$\frac{(\partial f_i^2)^2}{S_i^2}\ w_{ik}^2\ (S_k^1)^2$	$\frac{(\partial f_i^2)^2}{S_i^2}\ w_{ik}^2\ ((S_k^1)^2 + (y_k^1)^2)$
Capa oculta (m=1)	$\partial f_i^1\ y_k^0\ y_i^1 \sum_{r=1}^{N_2} \frac{(\partial f_r^2)^2}{S_r^2}$	$\partial f_i^1\ y_k^0 (y_i^1 + w_{ik}^1\ \partial f_i^1\ y_k^0) \sum_{r=1}^{N_2} \frac{(\partial f_r^2\ w_{ri}^2)^2}{S_r^2}$

A continuación vamos a realizar algunas consideraciones sobre la complejidad del algoritmo propuesto. Vamos a particularizar para el caso de perceptrones de tres capas y con desviaciones aditivas.

En primer lugar observemos los requisitos de almacenamiento. En las expresiones que aparecen en la Tabla 3.1 se aprecia la conveniencia de almacenar la sensibilidad de cada neurona S_i^m con m=1,2, por lo que el orden de complejidad es O(n), siendo n el número total de neuronas.

Respecto a los requisitos de cómputo para el cálculo de la sensibilidad, suponiendo un mismo coste para todas las operaciones aritméticas usuales, se aprecia a partir de las expresión (3.10) que es preciso realizar 3 multiplicaciones y 1 suma para cada peso dentro del radicando. Asumiendo el mismo coste para la raíz cuadrada y teniendo en cuenta que se multiplica por la derivada de la función de activación, se precisan $4p_i^m+2$ operaciones, siendo p_i^m el número de pesos de la neurona i perteneciente a la capa m. Así, el cálculo de la sensibilidad estadistica de una neurona tiene una complejidad de orden $O(p_i^m)$. Nótese que la derivada es necesaria para el

cálculo del gradiente del error en el algoritmo de entrenamiento clásico, por lo que no supone un coste adicional ni respecto a almacenamiento ni respecto a tiempo de cómputo. Puesto que es necesario evaluar la sensibilidad de todas las neuronas, el coste total es del orden $O(n\overline{p})$, siendo $\overline{p}$ el número medio de pesos de cada neurona.

En el caso del cálculo del gradiente, se necesitan básicamente 4 multiplicaciones y una división y, en el caso de neuronas en la capa oculta, un sumatorio de orden p_i^{m+1} por cada neurona, por lo que la complejidad es de nuevo $O(n\overline{p})$.

En definitiva, el coste total del algoritmo tiene una complejidad $O(n\overline{p})$ siendo n el número de neuronas y $\overline{p}$ el número medio de pesos de cada neurona. Experimentalmente, se ha encontrado que el tiempo requerido cuando se usa ARR, es ligeramente inferior al doble del tiempo necesario para ejecutar ARC. Téngase en cuenta que, usando el algoritmo propuesto en [CHO92] se requieren hacer varios entrenamientos con ARC para seleccionar una red con baja sensibilidad, mientras que ARR proporciona con un sólo entrenamiento una solución aceptable, lo cual supone un ahorro substancial respecto al tiempo requerido para dicho procedimiento.

En la Sección 3.4 se describen resultados que muestran la validez de la regla propuesta en el caso del modelo aditivo de desviación mientras que en la Sección 3.5 se considera el caso multiplicativo. En los experimentos realizados se han tratado varias cuestiones:

1. Si verdaderamente el uso de la regla de aprendizaje robusto proporciona siempre configuraciones de pesos con menor sensibilidad que cuando se usa el algoritmo clásico y si estas configuraciones son realmente más robustas.

2. Si el uso de la nueva regla implica una degradación en el aprendizaje.

3. La influencia de los parámetros que intervienen en la regla (factor de aprendizaje, momento, factor de robustez y número de neuronas en la capa oculta) en las prestaciones del MLP así como en la sensibilidad de la red.

3.4. VALIDACIÓN DE ARR USANDO EL MODELO DE DESVIACIÓN ADITIVO

El objetivo de las pruebas de validación es comprobar si efectivamente el uso de la expresión (3.4), correspondiente al algoritmo de aprendizaje propuesto (ARR), en lugar de la expresión (3.2), correspondiente al algoritmo de retropropagación clásico (ARC), proporciona configuraciones de pesos con menor sensibilidad sin afectar a las prestaciones finales del perceptrón. Se ha experimentado con diversos MLPs con una capa oculta correspondientes a

distintas aplicaciones. Fijada la estructura de tales perceptrones, éstos han sido entrenados usando las mismas condiciones iniciales (mismos valores iniciales de los pesos y mismos parámetros de aprendizaje) usando las dos reglas, ARC y ARR, ésta última considerando la expresión (3.4) particularizada para el caso del modelo aditivo de desviación.

Los ejemplos considerados se basan en los presentados en [CHO92]. La Tabla 3.2 muestra la descripción de los MLPs considerados y se interpreta de la siguiente manera: por ejemplo, MLP1 es un perceptrón con 2 entradas, 4 neuronas en la capa oculta y 1 neurona de salida, el perceptrón es un aproximador de la función $xe^{-x} + y\ tanh(x)$; para el entrenamiento se han utilizado 20 patrones de entrada y han tenido lugar 10000 épocas antes de parar el proceso de entrenamiento. En cada época el conjunto de entrenamiento completo, es decir, los 20 patrones, ha sido presentado a la red. El resto de las filas de la Tabla 3.2 siguen una interpretación similar.

Tabla 3.2. Descripción de los ejemplos

Ejemplo	Función	Neuronas/capa	Nº patrones	Nº épocas
MLP1	$xe^{-x} + y\ tanh(x)$	2,4,1	20	10.000
MLP2	$x \oplus y$	2,5,1	4	10.000
MLP3	$0.5(x^2 + y\ tanh(x))$ $0.5x + 0.3y$	2,8,2	100	5.000
MLP4	$xye^{-x} + y\ tanh(y)$	2,5,1	100	5.000
MLP5	$xyz\ (x + y + z)^{1/2}$	3,6,1	25	10.000
MLP6	*Decodificador de LEDs*	7,6,4	16	5.000

Los resultados obtenidos muestran que ambos algoritmos proporcionan configuraciones de pesos con prestaciones en cuanto a error de salida similares, si bien cuando se ha usado ARR estas configuraciones presentan una menor sensibilidad estadística media. En la Tabla 3.3 se muestran el error cuadrático de salida (ECM) y la sensibilidad estadística media (SSM) de red finales obtenidos con ambos procedimientos (ARC y ARR). Los resultados de la tabla están promediados sobre 40 pruebas para cada algoritmo, en las cuales se variaron los pesos iniciales mientras que los parámetros de aprendizaje α y β permanecieron fijados ambos al valor 0.9. La tabla muestra también el valor tomado para el parámetro K, usado para mantener la jerarquía de parámetros a optimizar tal y como se explicó en el Apartado 3.2. En todos los casos se ha utilizado la función sigmoidal como función de activación.

Tabla 3.3. Resultados de los experimentos para la validación del algoritmo ARR.

	ARC		ARR		
Ejemplo	ECM	SSM	ECM	SSM	K
MLP1	0.001275	0.711809	0.001702	0.500245	5
MLP2	0.011701	1.184025	0.018194	0.466902	10
MLP3	0.000193	0.404906	0.000469	0.364863	10
MLP4	0.000776	0.554860	0.000741	0.475694	25
MLP5	0.007828	0.743155	0.008642	0.337206	50
MLP6	0.002868	0.617671	0.042054	0.359850	200

En todos los casos se muestra que el valor de la SSM es menor cuando se usa ARR como algoritmo de entrenamiento. En concreto, para el ejemplo correspondiente a MLP5 se obtiene una SSM 2.2 veces inferior, mientras que el ECM es del mismo orden.

En la Figura 3.1 se muestra como varía la sensibilidad de salida en cada época para una secuencia del experimento MLP4. Puede observarse que en el caso de usar ARR la sensibilidad tiende a bajar durante el proceso de entrenamiento, mientras que en el caso de usar el algoritmo clásico, ARC, la sensibilidad no está en principio acotada y nótese que para 5000 épocas la sensibilidad de la red entrenada con ARC es 2.5 veces mayor que la entrenada con ARR. Este hecho, se repite siempre que los perceptrones son entrenados mediante ARR, mientras que en los entrenados con ARC las variaciones de los pesos siguen caminos en los que la sensibilidad puede variar de cualquier manera.

En el siguiente experimento, los pesos obtenidos con ambos algoritmos se han modificado respecto de sus valores nominales usando el modelo de desviación aditivo. Así, cada peso w_{ij}^m toma el valor $w_{ij}^m \pm \delta$, siendo δ una variable aleatoria con una desviación típica σ fijada de antemano. Para cada valor de σ se han realizado 4.000 tests escogiendo una de las 40 configuraciones de pesos obtenidas con el experimento MLP4. En la Figura 3.2 se muestra la degradación media del error cuadrático medio de salida para cada valor de σ cuando se han usado ambos algoritmos. En cambio, la Figura 3.3 representa el incremento del relativo del error, $\Delta\varepsilon_r$, respecto del error de salida obtenido con los valores nominales de los pesos, ε_0. El valor $\Delta\varepsilon_r$ viene dado por:

$$\Delta\varepsilon_r = \frac{|\varepsilon - \varepsilon_0|}{\varepsilon_0} \tag{3.18}$$

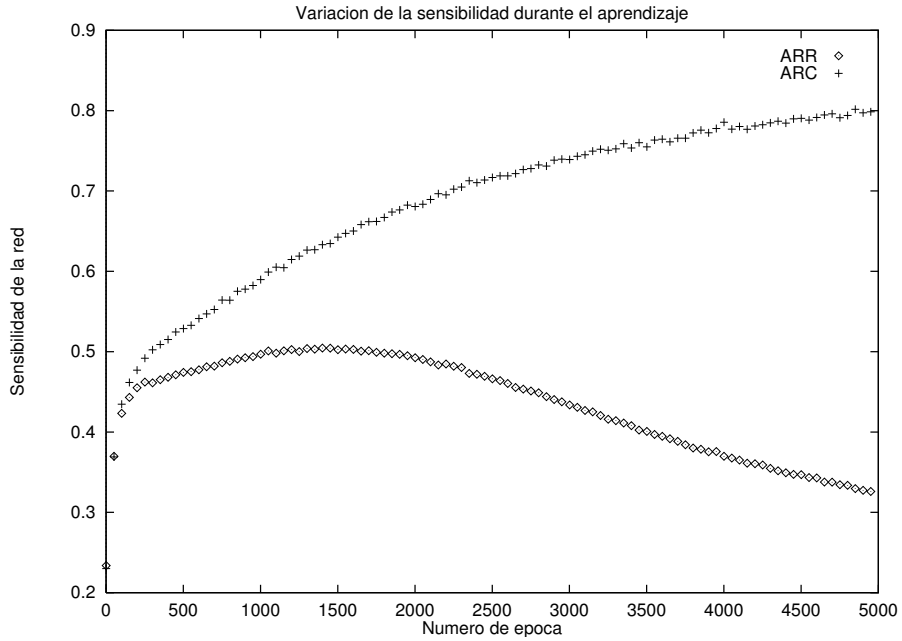

Figura 3.1. Evolución de la sensibilidad de la red durante el proceso de aprendizaje en MLP4.

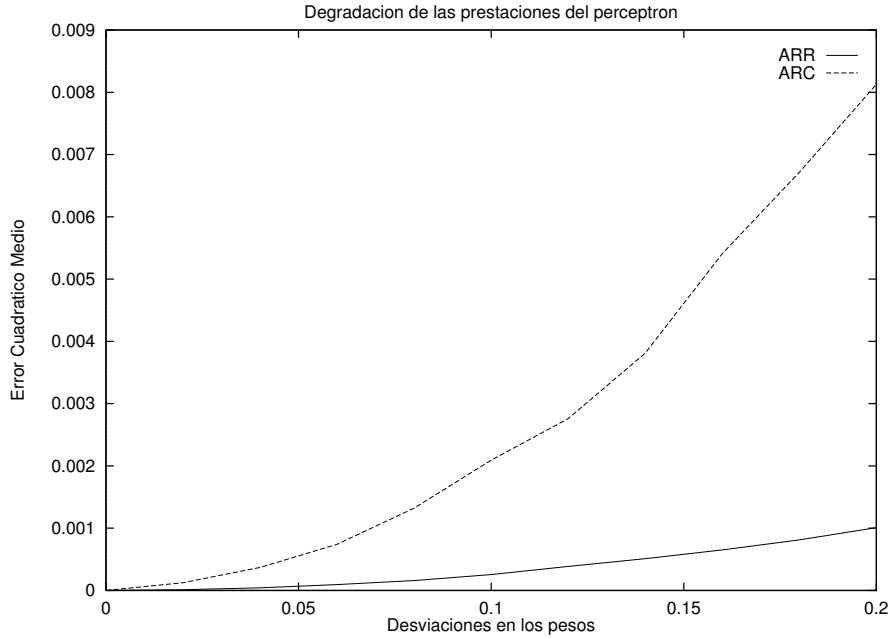

Figura 3.2. Degradación del error cuadrático medio del perceptrón frente a desviaciones aditivas en los pesos en MLP4.

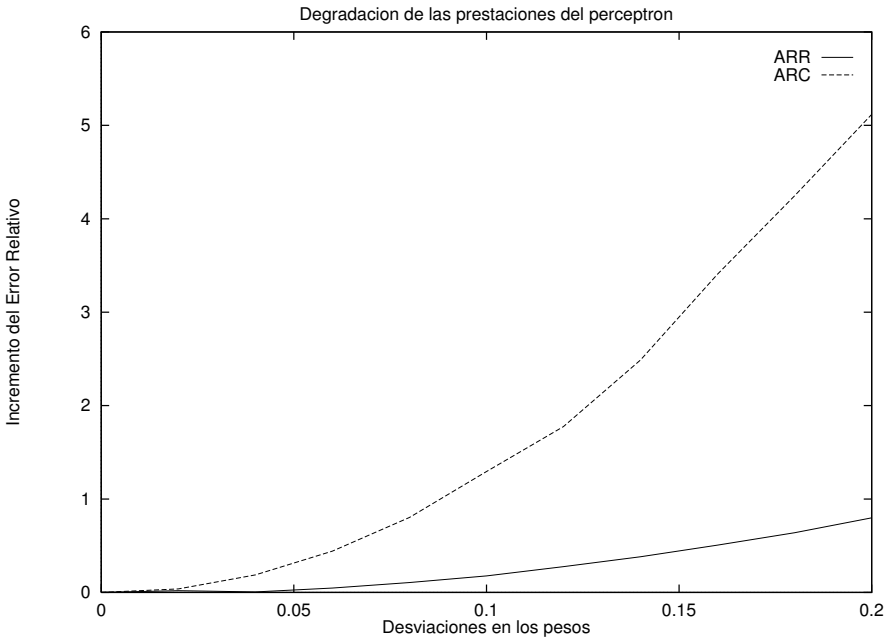

Figura 3.3. Incremento del error relativo frente a desviaciones aditivas en los pesos en MLP4.

Se puede apreciar a partir de las Figuras 3.2 y 3.3, que la configuración de pesos obtenida mediante ARR se degrada mucho menos que la obtenida mediante ARC al aumentar la desviación en los pesos, manteniendo de esta forma las prestaciones de la red incluso para valores moderadamente altos de σ.

En el siguiente apartado se mostrarán los resultados obtenidos cuando se considera el modelo de desviación multiplicativo, el cual nos parece más realista desde el punto de vista de una implementación analógica. Para este caso se han realizado experimentos mucho más exhaustivos sobre ejemplos algo más complicados, usando el análisis de la varianza (ANOVA) con objeto de validar el algoritmo propuesto.

3.5. VALIDACIÓN DE ARR USANDO EL MODELO DE DESVIACIÓN MULTIPLICATIVO

De forma similar a como se ha descrito en la Sección 3.5, hemos entrenado diferentes perceptrones de tres capas correspondientes a distintas aplicaciones usando el algoritmo clásico

(ARC) y el algortimo de retropropagación robusto (ARR). Las aplicaciones consideradas cubren tres tipos de problemas: un aproximador (la función seno [SUD94]), un predictor (la serie temporal de Mackey-Glass [WAN94]) y un clasificador (el problema de las dos espirales [YOS96]). Se han simulado perceptrones cuyas neuronas presentan una entrada de sesgo, con una única neurona de salida y con función de activación de tipo sigmoidal.

Los valores del factor de aprendizaje (α) y del momento (β) han sido fijados a 0.9 y 0 (es decir, no se considera), respectivamente, mostrando la Tabla 3.4 el valor considerado para el resto de los parámetros. Algunos de estos parámetros han sido fijados a valores utilizados en la bibliografía [SUD94], y otros han sido escogidos tras realizar un cierto número de pruebas con objeto de obtener unas prestaciones relativamente buenas, si bien probablemente no constituyan los mejores valores, ya que nuestro interés no es el de obtener una estructura óptima para cada problema sino el de validar el algoritmo propuesto. No obstante, más adelante se realiza un estudio estadístico de la influencia de estos parámetros cuando toman distintos valores usando como herramienta el Análisis de la Varianza (ANOVA) [FIS36]. La Tabla 3.4 se interpreta de la siguiente manera: en el caso del predictor, la fila correspondiente de dicha tabla indica que el perceptrón consta de 3 entradas, 11 neuronas en la capa oculta y 1 neurona de salida. El conjunto de aprendizaje cuenta con 500 patrones y el conjunto de test con 127. Durante el entrenamiento el conjunto de aprendizaje completo fue mostrado 1000 veces y el valor del parámetro K se fijó a 1. Los otros dos ejemplos se interpretan de forma similar a partir de dicha tabla.

Tabla 3.4. Descripción de los ejemplos estudiados.

Problema	Neuronas/capa	Patrones entrenamiento/test	Nº épocas	K
Aproximador	1,11,1	100/25	2000	1
Predictor	3,11,1	500/127	1000	1
Clasificador	2,9,1	1000/250	1000	0.1

3.5.1 Validación del algoritmo ARR.

Hemos testeado las dos reglas de aprendizaje (ARC y ARR) para cada problema. Para cada una de las reglas se han ejecutado 200 pruebas con el objetivo de validar estadísticamente el algoritmo de retropropagación robusto. Los resultados muestran que ambos algoritmos proporcionan configuraciones que presentan un error de salida similar respecto de la salida deseada, pero las configuraciones de pesos obtenidos mediante ARR presentan una menor sensibilidad que las obtenidas mediante ARC.

En las tablas 3.5 y 3.6 se muestra el error cuadrático medio (ECM) junto con el correspondiente intervalo de confianza al 95% obtenidos mediante ARC y ARR, respectivamente. También se muestran los valores de la desviación típica muestral en cada caso. En el caso del problema de las dos espirales, en lugar del error cuadrático medio, dichas tablas muestran el error de clasificación en tanto por ciento. El intervalo de confianza al 95% se ha calculado como 1.96 multiplicado por el error estándar, donde dicho error se define como el cociente de la desviación típica muestral y la raíz cuadrada del número de muestras, que en este caso es 200. Estos resultados se han obtenido usando el conjunto de patrones de test una vez que los perceptrones han sido entrenados con la regla correspondiente.

Tabla 3.5. Error de salida final obtenido con ARC.

Problema	Media	Desviación típica
Aproximador	0.000721 ± 0.000050	0.000363
Predictor	0.000110 ± 0.000003	0.000024
Clasificador	(8.84 ± 0.91) %	6.6 %

Tabla 3.6. Error de salida final obtenido con ARR.

Problema	Media	Desviación típica
Aproximador	0.000403 ± 0.000042	0.000302
Predictor	0.000100 ± 0.000004	0.000031
Clasificador	(11.34 ± 0.56) %	4.04 %

Comparando los resultados de la Tabla 3.5 con los de la Tabla 3.6 se puede apreciar que en el caso del aproximador y del predictor el ECM, cuando se usa ARR, no sólo no se degrada sino que incluso se mejora. En el caso del clasificador sin embargo, se obtiene un error de clasificación mayor con ARR, pero comparable al obtenido mediante ARC.

A partir de los resultados obtenidos para la sensibilidad estadística media (SSM) (Tablas 3.7 y 3.8) se desprende que en los tres ejemplos considerados se obtiene un valor de sensibilidad claramente inferior cuando se usa ARR en lugar de ARC, de hecho se obtiene un valor del orden del 60% más pequeño que el que se obtendría con ARC, lo que supone un decremento considerable de la sensibilidad.

Tabla 3.7. Sensibilidad estadística media obtenida con ARC.

Problema	Media	Desviación típica
Aproximador	2.22 ± 0.04	0.29
Predictor	1.00 ± 0.03	0.22
Clasificador	3.35 ± 0.12	0.89

Tabla 3.8. Sensibilidad estadística media obtenida con ARR.

Problema	Media	Desviación típica
Aproximador	1.33 ± 0.02	0.17
Predictor	0.61 ± 0.01	0.04
Clasificador	2.11 ± 0.06	0.45

En las Figuras 3.4 y 3.5 se muestran los histogramas de frecuencia correspondientes al error y a la sensibilidad para las dos reglas aplicadas, considerando los perceptrones que aproximan la función seno.

A partir de la Figura 3.4 se deduce que, en el caso del aproximador de la función seno, el ECM obtenido con ambas reglas (ARC y ARR) es del mismo orden, puesto que si ambos histogramas se representaran conjuntamente se encontrarían solapados. Sin embargo, la Figura 3.5 muestra que las sensibilidades obtenidas pertenecen a curvas que apenas solapan por un extremo y que, además, la sensibilidad estadística media menor se corresponde con la aplicación de ARR durante el entrenamiento del MLP.

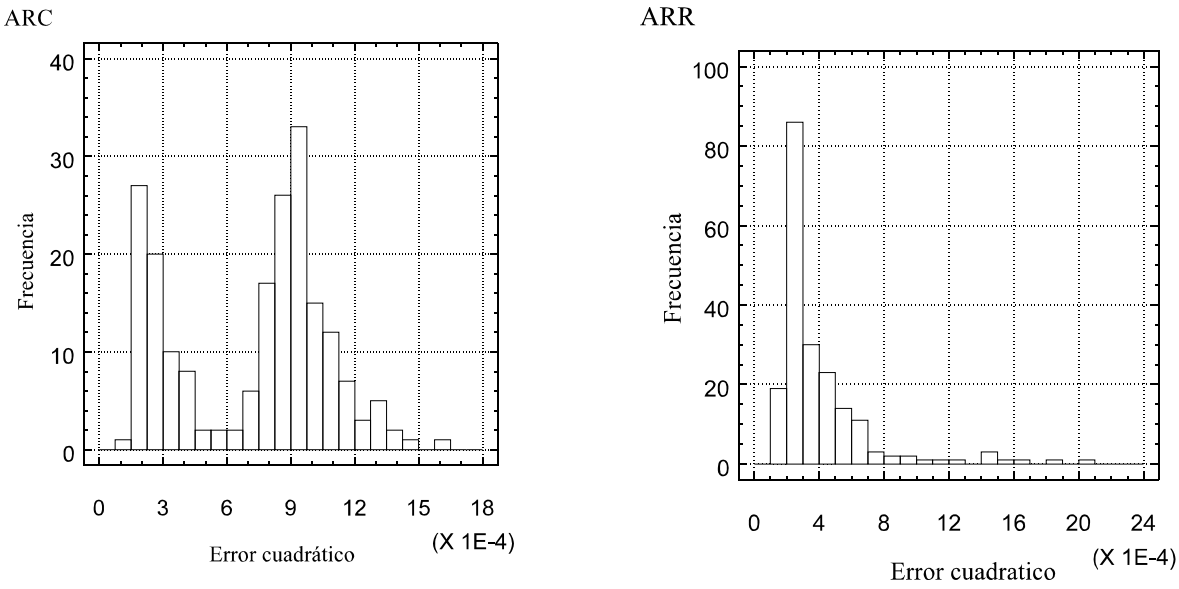

Figura 3.4. Histogramas de frecuencia del error cuadrático medio para el aproximador.

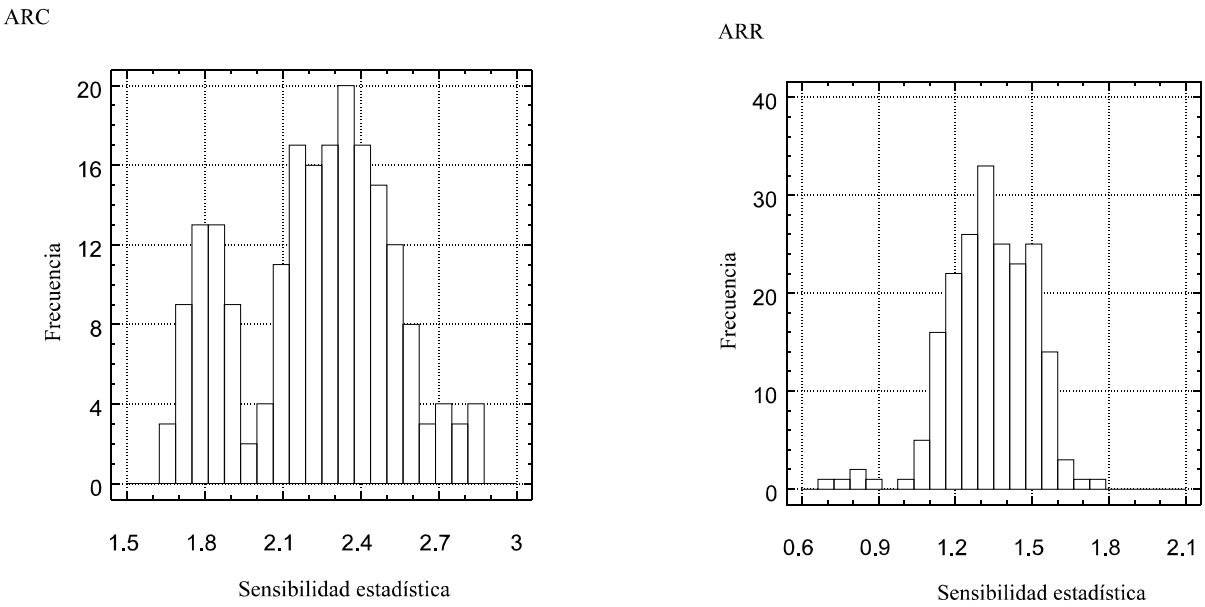

Figura 3.5. Histogramas de frecuencia de la sensibilidad estadística para el aproximador.

En las siguientes figuras (3.6 a 3.9) que presentan los histogramas correspondientes a los otros dos ejemplos anteriormente mencionados se aprecia esta misma conclusión.

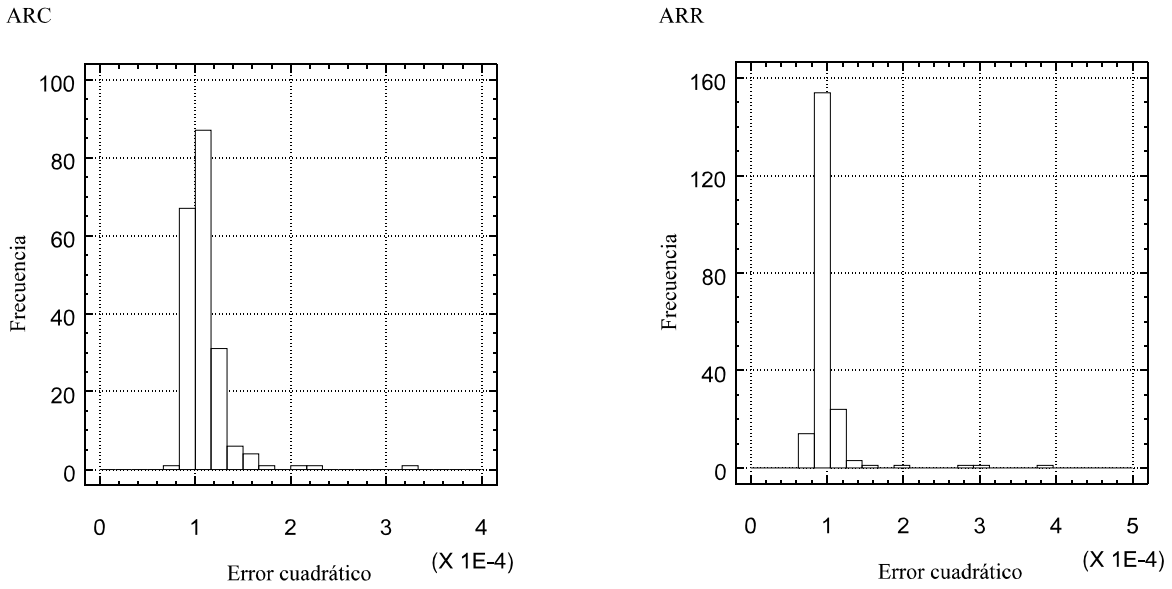

Figura 3.6. Histogramas de frecuencia del error cuadrático de salida para el predictor.

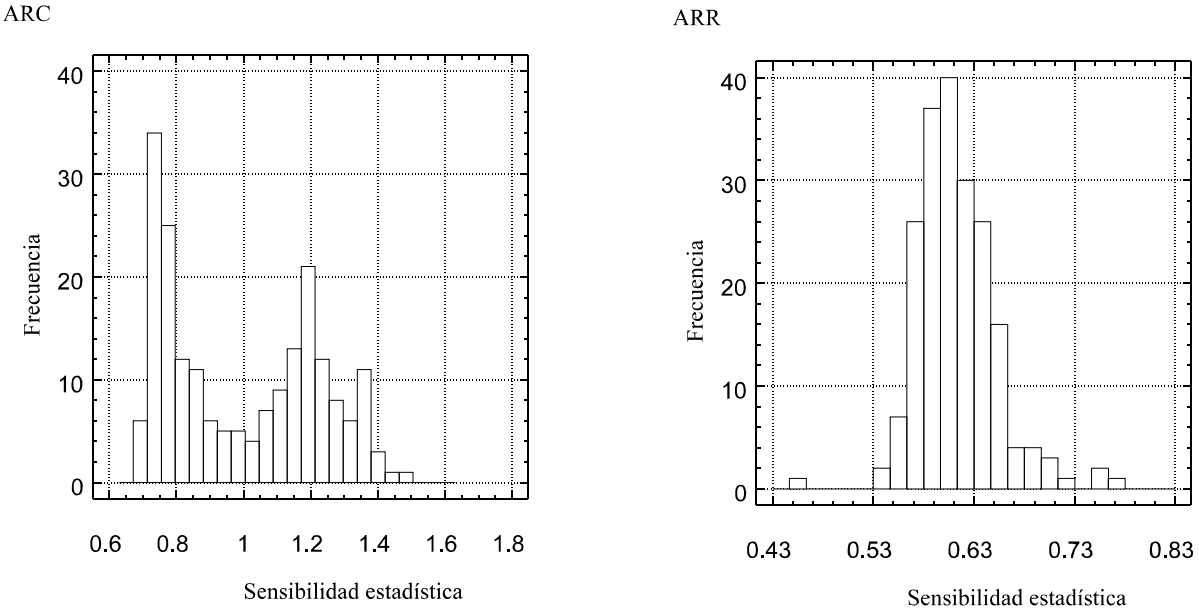

Figura 3.7. Histogramas de frecuencia de la sensibilidad estadística para el predictor.

En la Figura 3.8, correspondiente al clasificador para el problema de las dos espirales, se muestra el error de clasificación en tanto por ciento, en vez del error cuadrático medio.

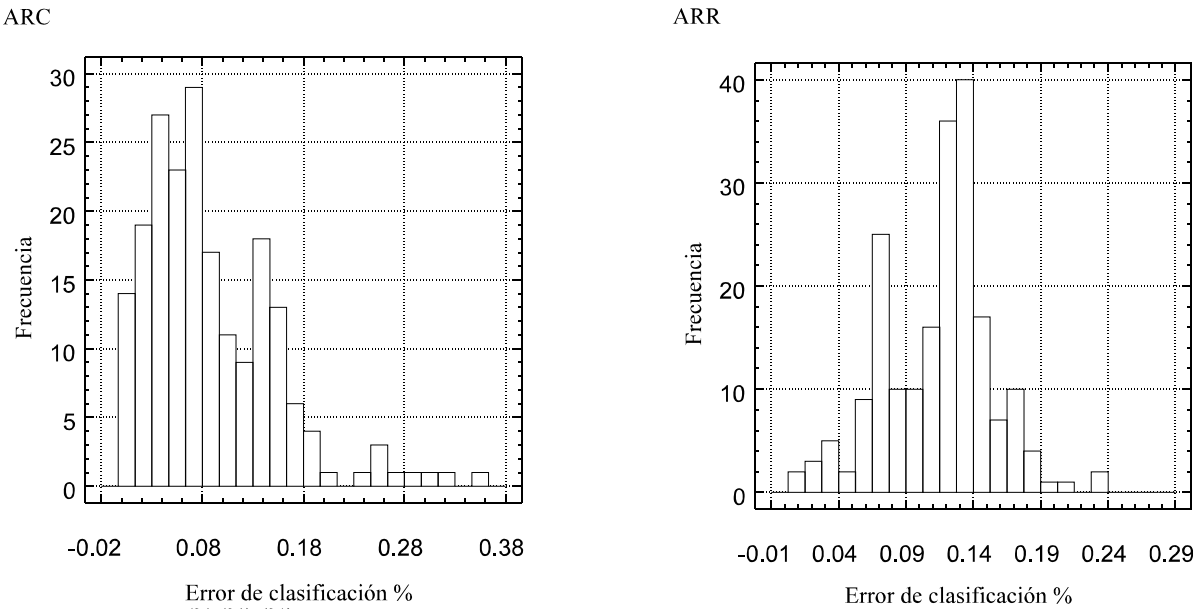

Figura 3.8. Histogramas de frecuencia del error de clasificación para el clasificador.

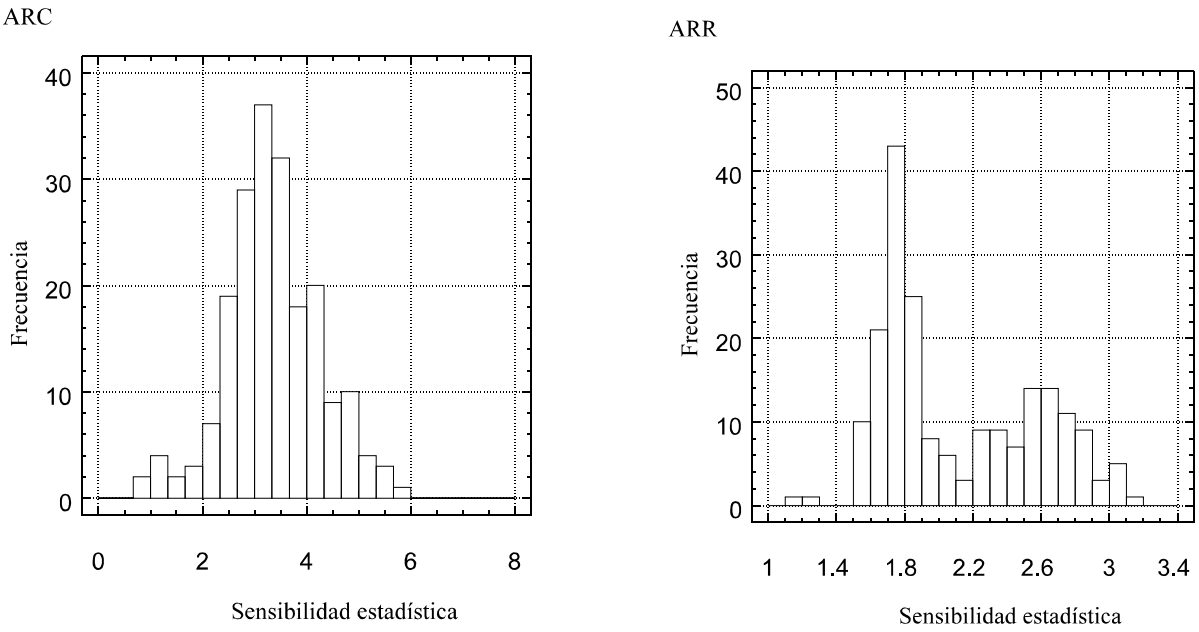

Figura 3.9. Histogramas de frecuencia de la sensibilidad estadística para el clasificador.

En vista de los histogramas presentados en las figuras anteriores (3.4 a 3.9) se pueden concluir dos hechos bastante relevantes:

a) El error de salida obtenido con ambos algoritmos está dentro del mismo rango pero las sensibilidades pertenecen a rangos diferenciados, es decir, la sensibilidad estadística media obtenida con ambas reglas es claramente distinta y, además, son menores en el caso de usar como regla de aprendizaje la del algoritmo robusto (ARR),

b) El error de salida obtenido con ARR presenta una mayor frecuencia de datos alrededor de la media que en el caso de ARC. Esto parece indicar que ARR explora mínimos anchos (suaves) en los cuales una pequeña desviación en el valor de los pesos es tolerada, por lo que presentan una menor sensibilidad a desviaciones. En otras palabras, usando ARC hay una probabilidad mayor de que la solución quede estancada en mínimos locales con alta sensibilidad estadística, en los cuales una leve variación en el valor de los pesos corresponde a un valor de error distinto del mínimo (es como si escapara del mínimo), degradando las prestaciones del MLP. Cuando se usa ARR, en cambio, no se acepta cualquier mínimo de error de salida sino sólo aquellos que producirán un determinado grado de tolerancia.

Estos dos hechos, que se repiten en mayor o menor medida en los tres ejemplos considerados, junto con los resultados mostrados en las tablas 3.5 a 3.8, confirman la validez de la regla ARR. No obstante en el siguiente apartado se realizará un test de la T de Student con objeto de estudiar la significación estadística del procedimiento propuesto frente al algoritmo clásico.

3.5.2. Análisis de la T de Student

Para comparar si el error cuadrático medio (ECM) y la sensibilidad estadística media (SSM) difieren entre el algoritmo clásico (ARC) y el propuesto (ARR) se utilizó el test de T de Student para muestras apareadas. Este test concluye si los valores medios poblacionales con ambos algoritmos se pueden considerar iguales o distintos a partir del conjunto de las 200 muestras observadas, con un determinado error prefijado de antemano ($p<0.01$). Se consideran las muestras apareadas debido a que los dos algoritmos se aplican sobre un mismo conjunto de datos, con idénticas condiciones iniciales aunque se han entrenado con algoritmos distintos.

El test de hipótesis planteado sería:

H_0: los valores medios proporcionados por los dos algoritmos para el error cuadrático medio (o la sensibilidad estadística media), son iguales

H_1: los valores medios difieren

Caso de aceptar la hipótesis alternativa H_1, se verá a posteriori cuál de los dos ofrece resultados menores para la magnitud considerada.

3.5.2.1. Resultados para el aproximador de la función seno

En la Tabla 3.9 se reflejan los valores medios de sensibilidad estadística y error cuadrático, así como las desviaciones típicas. En ella se puede observar que las dos variables analizadas tienen valores medios más bajos con el algoritmo ARR (0.0007218 con ARC frente a 0.0004031 con ARR para el ECM, y 2.2267 frente a 1.3329 para la SSM).

En la Tabla 3.10 se describe el resultado de la aplicación del test T de Student: en primer lugar, aparece la media de las diferencias entre los resultados obtenidos con ARR y ARC; en segundo lugar un intervalo de confianza al 95% para dicha diferencia, y por último, el valor experimental de la t y la significación o error cometido.

Tabla 3.9. Valores medios de ECM y SSM para el aproximador.

Variable	Algoritmo	Media	Desv. Típica
ECM	ARC	0.0007218	0.0003624
	ARR	0.0004031	0.0003011
SSM	ARC	2.2267	0.2940
	ARR	1.3329	0.1678

Tabla 3.10. Resultados de la inferencia estadística realizada para el ECM y la SSM.

		Intervalo de confianza al 95%			
Variable	Media de las diferencias	Lím. inferior	Lím superior	t	signif.
ECM	0.0003186	0.000252	0.000385	9.463	0.000
SSM	0.8937	0.8480	0.9395	38.521	0.000

La significación indica la probabilidad de que el aceptar la hipótesis alternativa (hay diferencias entre ambos algoritmos) sea falso. Este error es menor 0.001 lo que indica que de cada 1000 experimentos en 1 no se producirían diferencias significativas en cuanto a ECM y a SCM. Es

decir ambos métodos proporcionan soluciones estadísticamente distintas. A igualdad del valor de la significación, cuanto mayor sea el valor del factor t, mayor es la significación estadística de la diferencia de resultados.

Observando el intervalo de confianza, puede apreciarse en la Tabla 3.10 que, tanto para el ECM como para la SSM, el valor 0 no está incluido entre los límites y esto indica que existen diferencias significativas entre las dos medias. Además la diferencia del ECM con ambos métodos se encuentra en un 95% de los casos entre 0.000252 y 0.000385, mientras que para la SSM esta diferencia se encuentra entre 0.8480 y 0.9395. En ambos casos los resultados son menores con ARR puesto que las diferencias son positivas.

Para el caso del aproximador se concluye que se obtiene menores valores para el ECM y la SSM cuando se usa ARR. Además, estos resultados no son producto del azar, sino que se ha probado que se obtienen diferencias significativas respecto de los resultados proporcionados por ARC. Para la SSM estas diferencias son más pronunciadas tal como se puede apreciar a partir de los límites del intervalo de confianza para la diferencia de medias. Cuanto mayor sea el valor del factor t mayor es la significación de estas diferencias y para la SSM es 38.251, mientras que para el ECM es 9.643, lo que indica que la diferencia de resultados es más significativa respecto de la SSM que respecto del ECM.

3.5.2.2. Resultados para el predictor de la serie temporal

Los resultados correspondientes se muestran en las tablas 3.11 y 3.12. Puede observarse en la Tabla 3.11 que de nuevo se alcanzan menores valores para el ECM y la SSM en media al usar ARR. En concreto el ECM medio sobre las 200 muestras con ARC es 0.000110 frente a 0.000995 con ARR. Respecto a la SSM la diferencia es más notable, ya que con ARC es 0.99695 mientras que con ARR es 0.61251.

Tabla 3.11. Valores medios de ECM y SSM para el predictor.

Variable	Algoritmo	Media	Desv. Típica
ECM	ARC	0.000110	0.000024
	ARR	0.000100	0.000031
SSM	ARC	0.99695	0.2286
	ARR	0.61251	0.03955

Tabla 3.12. Resultados de la inferencia estadística realizada para el ECM y la SSM.

		Intervalo de confianza al 95%			
Variable	Media de las diferencias	Lím. inferior	Lím superior	t	signif.
ECM	0.00001	0.0000047	0.0000153	3.691	0.000
SSM	0.38444	0.3510121	0.4178749	22.676	0.000

Observando la Tabla 3.12 se deduce del intervalo de confianza al 95% que existen diferencias significativas en los resultados obtenidos con ARR para el ECM y la SSM respecto de los obtenidos con ARC. Además se obtienen mejores resultados con ARR al ser las diferencias positivas.

También se deduce que estas diferencias son más pronunciadas en el caso de la SSM (están entre 0.351 y 0.418 en el 95% de los casos) que con el ECM (entre 0.0000047 y 0.0000153). El valor de t nos indica que al aceptar la hipótesis alternativa como cierta se está cometiendo menor error en el caso de la SSM (t=22.676) que en el caso del ECM (t=3.691), aunque en ambos casos este error es menor que 0.001.

3.5.2.3. Resultados para el clasificador del problema de las 2 espirales

Para el clasificador, los resultados se muestran en las tablas 3.13 y 3.14. En vez del ECM se muestra el error de clasificación (EC) expresado en tanto por 1. Puede deducirse que de nuevo hay diferencias significativas ya que en ningún caso el intervalo de confianza contiene al valor 0, y de hecho el valor de la significación es inferior a 0.001, lo que indica que la hipótesis alternativa (hay diferencias significativas) es falsa 1 de cada 1000 veces aproximadamente.

Tabla 3.13. Valores medios de ECM y SSM para el clasificador.

Variable	Algoritmo	Media	Desv. Típica
EC	ARC	0.08846	0.06554
	ARR	0.1134	0.04038
SSM	ARC	3.35657	0.8906
	ARR	2.11257	0.45605

Tabla 3.14. Resultados de la inferencia estadística realizada para el EC y la SSM.

		Intervalo de confianza al 95%			
Variable	Media de las diferencias	Lím. inferior	Lím superior	t	signif.
EC	-0.02494	-0.036022	-0.013858	-4.438	0.000
SSM	1.243996	1.1004006	1.387592	17.083	0.000

Así, se encuentran diferentes resultados al aplicar ARR respecto de aplicar ARC. Sin embargo, se obtienen mejores resultados con ARC respecto del error de clasificación (EC). El valor de EC con ARC es 0.08846 frente a 0.1134 obtenido con ARR. Además se obtienen diferencias negativas lo que indica que los valores menores de EC se obtienen con el algoritmo clásico.

Respecto a la sensibilidad estadística media (SSM), los mejores resultados se obtienen con ARR (2.11257) frente al valor obtenido al aplicar ARC (3.35657), estando las diferencias entre 1.10 y 1.39 en el 95% de los casos.

El valor de t nos indica que la significación es mayor caso de considerar la SSM (t=17.083) ya que en el caso del EC t es igual a -4.438.

En definitiva, con estos análisis se ha puesto de manifiesto que:

A) ARC y ARR proporcionan resultados con diferencias estadísticamente significativas. Es decir, estas diferencias varían con el experimento de aprendizaje realizado.

B) La sensibilidad estadística presenta valores diferentes para ARC y ARR, siendo estas diferencias estadísticamente significativas. Como los valores de la sensibilidad estadística media son menores para ARR, se puede concluir que, efectivamente con este aprendizaje se obtienen redes más robustas.

C) Aunque se obtienen diferencias significativas en cuanto a error de salida, estas diferencias son pequeñas y, de hecho, el valor de la t de Student es bastante menor que cuando se considera la SSM, lo que indica que aún teniendo diferencias significativas, esta significación es menor. En el caso del aproximador y del predictor, los mejores resultados para el ECM se obtienen con ARR, mientras que en el caso del clasificador se obtienen mejores prestaciones con ARC respecto del error de clasificación.

De esta manera, respecto a la SSM se puede concluir que se obtienen mejores prestaciones cuando se usa ARR, y que además este resultado es estadísticamente significativo. En cambio,

respecto al aprendizaje no se puede concluir que ARR mejore ni empeore la solución, ello depende del problema concreto, así como de los valores fijados para los parámetros de la regla de aprendizaje.

3.5.3. Influencia de los parámetros de la regla ARR.

Con este tercer experimento se pretende analizar la influencia en el proceso de aprendizaje de: el factor de aprendizaje α, el factor de momento β, el factor de robustez γ, y el número de neuronas de la capa oculta. Para realizar este estudio se ha utilizado una potente herramienta estadística, el Análisis de la Varianza (ANOVA). La teoría y metodología de ANOVA fue desarrollada principalmente por R.A. Fisher en la década de los veinte [FIS36]. Es una técnica estadística que permite analizar y comparar experimentos en los cuales una respuesta cuantitativa y unidimensional es descrita como función de otras variables, llamadas factores, de forma cuantitativa o cualitativa. Nuestro objetivo al realizar este análisis es determinar qué factor o factores son los más influyentes en el valor de la sensibilidad de la red que se obtiene al final del algoritmo de aprendizaje ARR. Se observa que es precisamente el factor de robustez γ el principal responsable de cara a minimizar la sensibilidad.

En un primer paso, el análisis de la varianza permite verificar cuáles son los factores que producen mayores alteraciones en la salida cuando sus valores cambian, indicando si los efectos debido a cambios en los mismos dentro de ciertos rangos, llamados niveles, son o no equivalentes. Los niveles de un determinado factor que no son estadísticamente diferentes son llamados *grupos homógeneos*, y de esta manera la selección entre los distintos niveles pertenecientes al mismo grupo homógeneo no tiene una repercusión significativa en la salida. Los factores considerados para el análisis aplicado al perceptrón son α, β, γ y el número de neuronas de la capa oculta. La influencia de dichos factores en el error de salida y en la sensibilidad de la red va a ser analizada mediante dicha técnica estadística.

Para realizar el análisis de la varianza es necesario realizar múltiples tests de la regla de aprendizaje fijando distintos valores (niveles) para los factores a analizar. Hemos considerado para cada factor los niveles que se muestran en la Tabla 3.15. La elección de estos valores se ha realizado tras simular distintas configuraciones y estudiar cuáles producen cambios significativos. Para cada una de las combinaciones posibles de valores de estos factores se han realizado tres pruebas. Estos mismos valores han sido considerados en los dos ejemplos utilizados: el aproximador del seno y el predictor de la serie temporal.

Tabla 3.15. Factores y niveles considerados para el análisis ANOVA.

Factor	Niveles
α	0.3 0.6 0.9
β	0.0 0.5 0.9
γ	0.0 0.01 0.1 1.0
No. neuronas	8 11 15

Antes de obtener las conclusiones del análisis de la varianza, es necesario comprobar las hipótesis de partida para realizar correctamente el estudio. Es decir, hay que comprobar la aleatoriedad de las muestras, la independencia de las variables, la normalidad de las distribuciones y la homogeneidad de las varianzas. Se pueden verificar las hipótesis del modelo mediante un análisis pormenorizado de los residuos [BOX89, MON91]. Estos residuos son las cantidades que quedan después de eliminar las contribuciones sistemáticas del modelo propuesto. Si las hipótesis relativas al modelo son ciertas, se espera encontrar, aparte de las restricciones impuestas por el análisis mismo, que los residuos varíen aleatoriamente. Por el contrario, si se descubre que los residuos contienen tendencias sistemáticas inexplicables, hay que sospechar del modelo. Por lo tanto, debe construirse y analizarse una tabla de residuos como requisito inprescindible anterior a cualquier conclusión estadística. La comprobación de las hipótesis del modelo se ha realizado previamente usando el análisis de los residuos para su verificación.

Los resultados del estudio ANOVA de la sensibilidad en el caso del aproximador y la serie temporal, se muestran en la Tabla 3.16, donde se aprecia la F de Snedecor experimental F_{exp} junto con el valor p de significación. Como se puede apreciar, el factor γ tiene la mayor relevancia estadística en la respuesta de sensibilidad del MLP, respecto de los otros factores analizados, ya que cuanto menor sea el valor p o mayor sea el valor de F_{exp}, mayor es la relevancia del factor correspondiente. Se aprecia también que en el caso del aproximador el número de neuronas en la capa oculta apenas influye, de manera que se puede concluir que el hecho de añadir arbitrariamente más neuronas en dicha capa con objeto de aumentar la redundancia de la red no tiene porqué corresponderse con una disminución efectiva de la sensibilidad tal como alguna vez se ha insinuado [NIJ89], a menos que vaya acompañada de una distribución adecuada del aprendizaje entre las neuronas que la constituyen, utilizando un algoritmo apropiado [CHI94]. En el caso del predictor el número de neuronas es significativo (p=0.0000) si bien su influencia es menor comparada a la del resto de los parámetros que presentan el mismo valor de p y valores de F_{exp} mayores. En [STE90] se prueba que, en el caso de estructuras de tipo Adaline con función de activación de tipo escalón, el número de neuronas por capa no influye en la probabilidad de error lo cual parece corroborar las conclusiones obtenidas en nuestro análisis.

Tabla 3.16. Análisis de los principales factores que influyen en el error de salida y en la sensibilidad.

	Aproximador				Predictor			
	ECM		SSM		ECM		SSM	
Efecto principal	F_{exp}	p	F_{exp}	p	F_{exp}	p	F_{exp}	p
α	33.67	0.0000	43.50	0.0000	138.67	0.0000	10.04	0.0001
β	39.44	0.0000	9.14	0.0001	174.68	0.0000	2.51	0.0832
γ	1.51	0.2128	127.67	0.0000	4.13	0.0068	78.74	0.0000
Nº neuronas	1.86	0.1580	0.27	0.7651	11.27	0.0000	14.85	0.0000

Un análisis similar se ha realizado con el error cuadrático medio de salida. En este caso, tal como se ve en la Tabla 3.12 el factor γ no es estadísticamente relevante, siendo α y β los factores con mayor peso estadístico en la respuesta de error del MLP. Esto también es especialmente interesante porque nos dice que el nuevo factor, el factor de robustez, apenas afecta al error de aproximación en el proceso de aprendizaje, el cual depende fundamentalmente de los factores α y β.

Tabla 3.17. Test de rango múltiple para el factor γ en el análisis de la sensibilidad.

	Aproximador del seno				Serie temporal			
Nivel	Media	Grupos Homogéneos			Media	Grupos Homogéneos		
γ=1	1041	X			525	X		
γ=0.1	1800		X		758		X	
γ=0	1961			X	882			X
γ=0.01	1917			X	894			X
	Límite para establecer diferencias significativas: ± 0.106				Límite para establecer diferencias significativas: ± 0.054			

La Tabla 3.17 muestra los tres grupos homógeneos que se han establecido entre los valores del factor γ en el análisis de la sensibilidad. Uno de los grupos incluye γ=0 y γ=0.01, y los otros dos grupos son para γ=0.1 y γ=1, respectivamente. Entre estos tres grupos no hay intersección lo cual significa que los tres grupos son completamente diferentes desde un punto de vista estadístico. Este hecho indica que la sensibilidad del MLP obtenida tras el aprendizaje con ARR está fuertemente afectada por el valor de γ utilizado en dicho algoritmo y, así, se obtienen distintos

valores de la sensibilidad cuando se modifica el valor de este parámetro.

3.5.4. Estudio de la tolerancia a faltas de desviación multiplicativas.

El objetivo de este experimento es demostrar que los perceptrones entrenados con el algoritmo ARR son más tolerantes a faltas de desviación multiplicativas. Para ello, hemos simulado desviaciones de tipo multiplicativo en los pesos de las configuraciones proporcionadas por el algoritmo clásico (ARC) y el robusto (ARR). Todos los pesos w_{ij}^m han sido perturbados dentro de un rango $\pm\delta w_{ij}^m$ con δ variando desde 0.02 hasta 0.4 (desde un 2% al 40% respecto de su valor nominal). Las Figuras 3.10 y 3.11 muestran respectivamente el incremento del error cuadrático medio y del error relativo de salida frente a desviaciones en los pesos en el caso del aproximador de la función seno para los MLPs entrenados con ambas reglas. Estos resultados están promediados sobre 40 conjuntos de pesos para cada regla, y para cada valor de δ se han considerado 20 pruebas (cada prueba consiste en una variación aleatoria de todos los w_{ij}^m dentro del rango correspondiente).

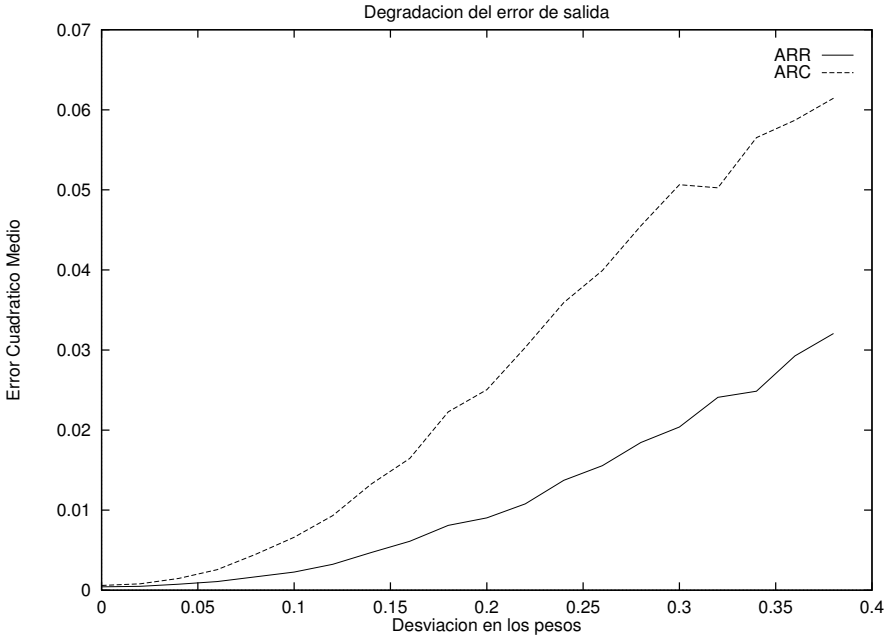

Figura 3.10. Incremento del error cuadrático medio del aproximador

Se observa que la degradación de las prestaciones en los perceptrones entrenados con ARR es mucho menor que en aquellos entrenados mediante la regla clásica (ARC). Las Figuras 3.12 a 3.15 muestran resultados similares obtenidos con los ejemplos del predictor y del clasificador, respectivamente. En el caso del clasificador, aunque el efecto no sea tan evidente como en los otros anteriores se observa como incluso, aunque inicialmente las prestaciones obtenidas con el

algoritmo clásico son mejores que las obtenidas mediante el algoritmo robusto, al crecer las desviaciones, las prestaciones de las configuraciones obtenidas mediante ARC se degradan en mayor grado respecto de las de los perceptrones entrenados usando la regla ARR.

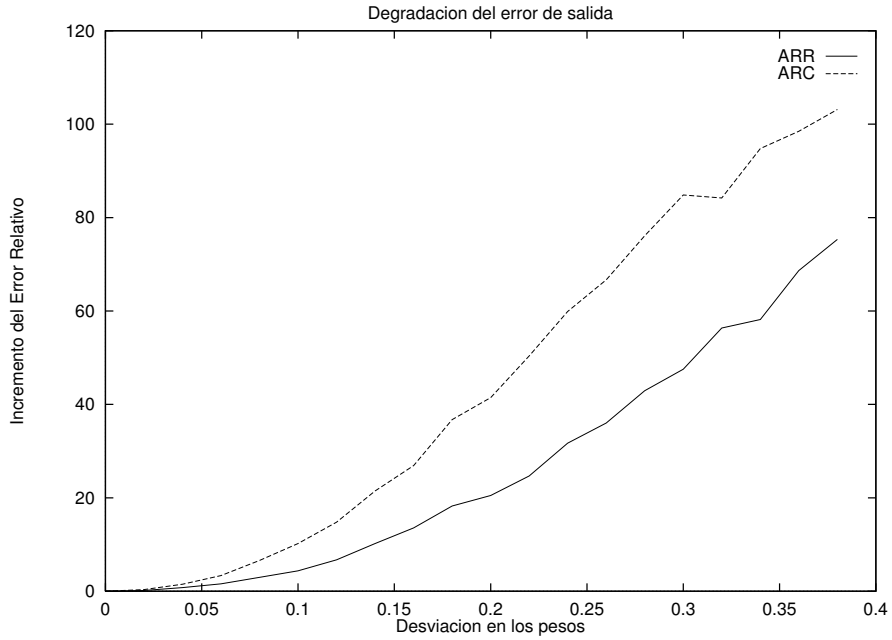

Figura 3.11. Incremento del error relativo en el aproximador.

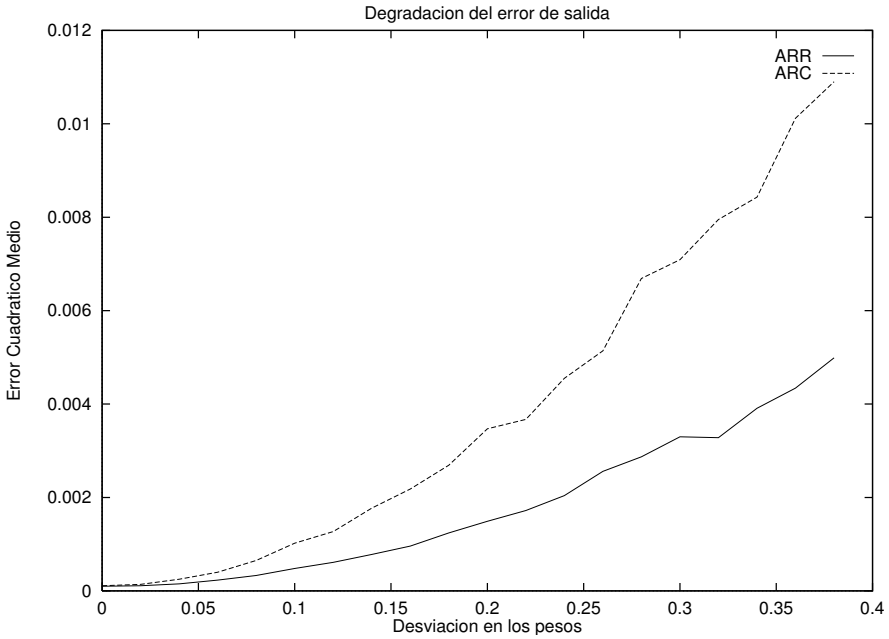

Figura 3.12. Incremento del error cuadrático medio en el predictor.

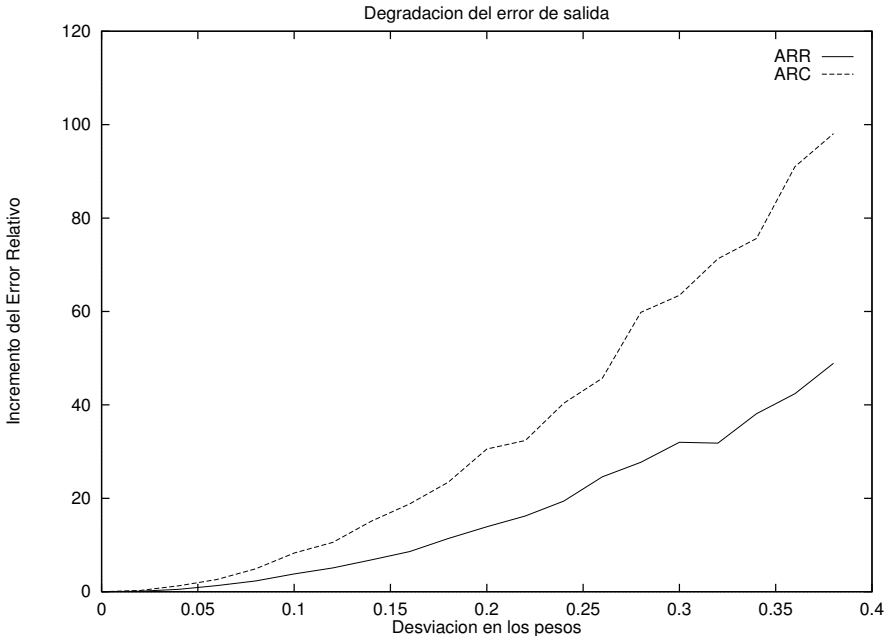

Figura 3.13. Incremento del error relativo en el predictor.

La Figura 3.14, correspondiente al clasificador del problema de las 2 espirales, se refiere al error de clasificación, el cual es intrínsecamente bastante tolerante a desviaciones respecto a los otros ejemplos. Esto se debe a que aunque el clasificador en vez de dar una salida de 1.0 dé una salida de 0.7, el patrón sigue clasificándose bien, pero incluso en este caso una menor sensibilidad estadística media implica una menor degradación de sus prestaciones.

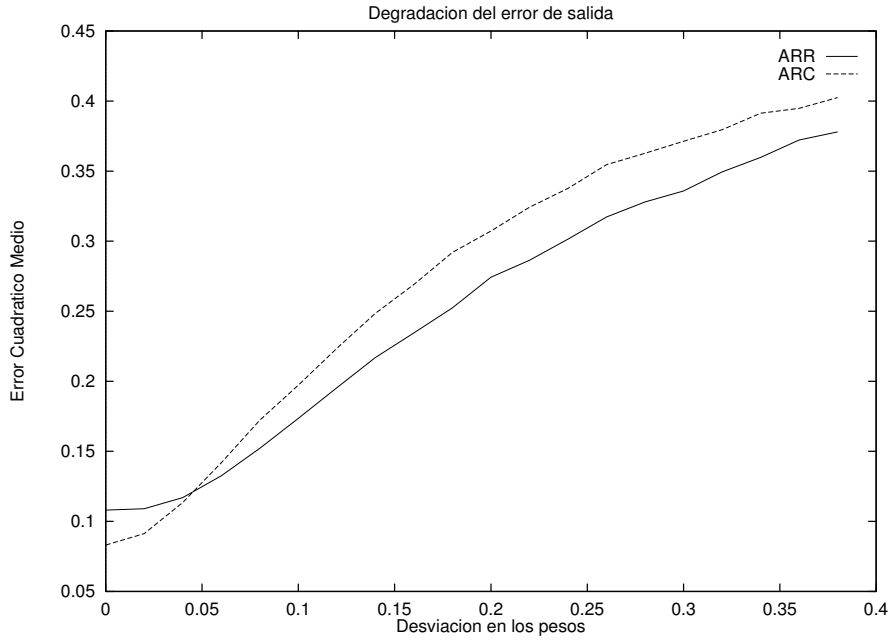

Figura 3.14. Incremento del error de clasificación en el clasificador.

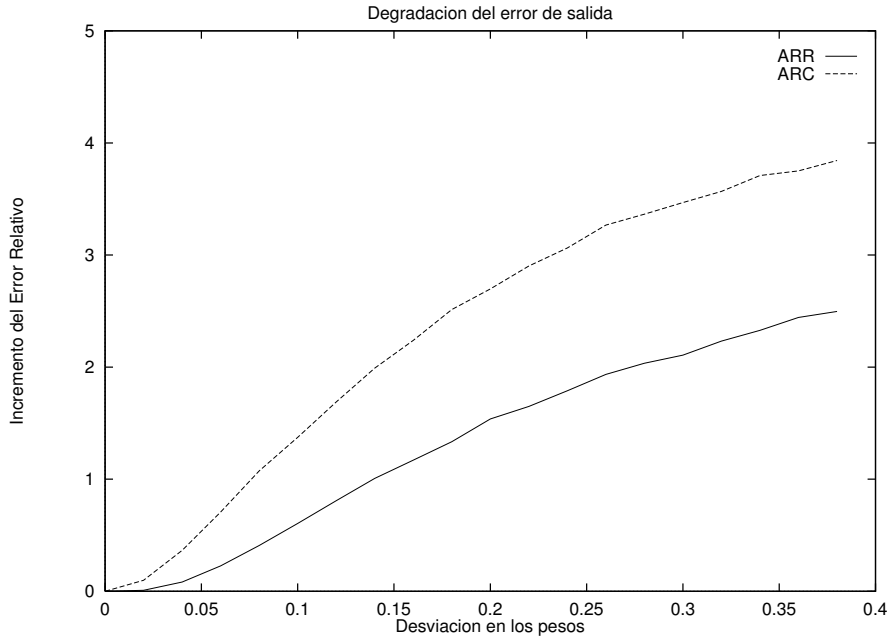

Figura 3.15. Incremento del error relativo en el clasificador.

3.5.5. Estudio de la tolerancia a faltas de anclaje.

También hemos estudiado el efecto de la tolerancia de los perceptrones cuando se ven afectados por faltas catastróficas [BER93]. De esta forma, hemos simulado faltas de anclaje a 0 de las neuronas de la capa oculta. Este tipo de falta simula la desconexión física de la red de una unidad neuronal de forma que su salida se fija al valor 0 para cualquier valor de las entradas [FEL91]. Este tipo de faltas es considerado en [STE90], si bien en el contexto de la mayoría de las implementaciones físicas de RNAs resultan ser poco realistas [EDW95].

En la Tabla 3.18 se muestran los valores del error cuadrático medio (ECM) o error de clasificación (EC) que presentan los MLPs considerados tras el entrenamiento con ARC y ARR.

La Tabla 3.19 muestra la variación de las prestaciones de los MLPs en cuanto a error de salida en función del número de neuronas ancladas a 0 simultáneamente, para los tres problemas considerados, el aproximador, el predictor y el clasificador. En el caso del clasificador, la Tabla 3.19 se refiere al error de clasificación. Los resultados muestran el incremento relativo del error, que viene dado por la expresión (3.18) y que se encuentra promediado tal como se ha descrito en el Apartado 3.5.4, es decir, sobre 40 configuraciones de pesos para cada una de las dos reglas. Las neuronas ancladas se han escogido aleatoriamente, realizándose 20 pruebas distintas para

cada número considerado de neuronas a anclar y cada configuración de pesos considerada .

Tabla 3.18. Error de salida en ausencia de faltas de anclaje.

	Aproximador del seno	Predictor de la serie temporal	Clasificador (2 espirales)
	ECM	ECM	EC (%)
ARC	0.00059	0.00011	8.31 %
ARR	0.00042	0.00010	10.81 %

Tabla 3.19. Incremento relativo del error en presencia de faltas de anclaje.

	Aproximador del seno		Predictor de la serie temporal		Clasificador (2-espirales)	
Nº neuronas ancladas a 0	ARC	ARR	ARC	ARR	ARC	ARR
0	0	0	0	0	0	0
1	83.88	62.07	139.09	33.30	1.29	0.82
2	133.47	108.93	260.91	78.80	2.21	1.47
3	163.51	137.81	368.09	106.70	2.99	2.01
4	197.63	173.10	472.00	145.00	3.65	2.54

Los resultados muestran que incluso en el caso de faltas de anclaje, los perceptrones entrenados mediante ARR mantienen mejor las prestaciones que los entrenados mediante un algoritmo clásico. Por ejemplo, con 1 única neurona anclada a 0, el MLP entrenado con ARC, correspondiente al predictor de la serie temporal, presenta un ECM igual a 0.015 (136 veces mayor que el ECM en ausencia de faltas), mientras que el entrenado con ARR presenta un ECM igual a 0.00343 (34.3 veces mayor que el ECM nominal). Con 3 neuronas ancladas a cero, el aproximador del seno proporciona con ARC un ECM igual a 0.09706 (164 veces mayor que el ECM nominal) mientras que con ARR el ECM es igual a 0.0583 (139 veces mayor).

En el caso del clasificador, el error de clasificación cuando hay neuronas ancladas a cero es similar para los perceptrones entrenados con ambas reglas, aunque el incremento del error relativo es menor en el caso del MLP entrenado con ARR (es decir, se degrada menos respecto del EC nominal). En realidad, el error cuadrático medio en presencia de faltas de anclaje es inferior en el caso de ARR. La explicación de este hecho es la misma que se dio en el apartado

anterior: un clasificador es por naturaleza más tolerante a fallos puesto que el error de clasificación tiene un amplio margen de respuesta respecto a un aproximador o un predictor. Para implementaciones de este tipo, quizás sería posible definir de otra forma la sensibilidad estadística de manera que se obtuviera un algoritmo similar en el caso de clasificadores, más adecuado a su naturaleza. Por ejemplo en el caso de que la red actuase como codificador sería más adecuado ponderar las salidas según el número de bit que representen.

3.6. CONCLUSIONES

En este capítulo se ha presentado un algoritmo de retropropagación modificado al que hemos llamado Algoritmo de Retropropagación Robusto (ARR). Este algoritmo mantiene las mismas prestaciones en cuanto a aprendizaje que los algoritmos clásicos (ARC) pero proporciona configuraciones de pesos que presentan una menor sensibilidad de red, esto es, la red resultante tiene una mayor tolerancia a desviaciones en los pesos de las neuronas que la constituyen.

Este nuevo algoritmo se basa en la modificación de la regla clásica de aprendizaje mediante la adición de un nuevo factor, llamado factor de robustez, el cual se obtiene a partir de la sensibilidad de la red. Las expresiones se han particularizado para los dos modelos de desviación introducidos en el Capítulo 2, el modelo aditivo y el multiplicativo. Se han considerado perceptrones con una capa oculta ya que por las razones que se han expuesto, en este tipo de perceptrones los cálculos involucrados se simplifican y requieren una menor cantidad de recursos de memoria respecto a MLPs con un número mayor de capas, si bien esta misma metodología podría aplicarse a estos últimos.

En los resultados obtenidos se observa que:

A) Los perceptrones entrenados con ARR mantienen las mismas prestaciones de aprendizaje respecto a los entrenados con ARC.

B) Los perceptrones entrenados con ARR presentan una menor sensibilidad a desviaciones que los obtenidos mediante ARC. Esta diferencia en prestaciones es estadísticamente significativa.

C) Los perceptrones con menor sensibilidad mantienen mejor las prestaciones de aprendizaje cuando los pesos son desviados según los modelos de error considerados.

D) Los perceptrones con menor sensibilidad mantienen mejor las prestaciones de aprendizaje incluso en presencia de faltas de anclaje.

E) El factor de robustez introducido en la regla de aprendizaje es el principal responsable de obtener mayor tolerancia a fallos y, además, su efecto sobre el error cuadrático medio es prácticamente despreciable.

F) El incremento arbitrario del número de neuronas no implica necesariamente que las redes sean más robustas.

Los hechos E) y F) se han comprobado estadísticamente a partir de un análisis de la varianza practicado con objeto de estudiar la influencia de los factores que intervienen en la regla de aprendizaje.

El algoritmo propuesto puede ser considerado como una técnica de regularización [MAO93]. De este modo, el nuevo término actuaría como estabilizador suavizando la superficie de error de salida.

En principio, la influencia del término de gradiente añadido a la regla de aprendizaje es la de impedir que los valores de los pesos alcancen magnitudes grandes, pero en combinación con las contribuciones correspondientes a los términos responsables del aprendizaje se alcanza una gran sinergia, de forma que se consiguen configuraciones de pesos con un reparto más uniforme del aprendizaje, alcanzándose así una mayor tolerancia a faltas (paramétricas y catastróficas).

Capítulo 4

Minimización de la Sensibilidad Cuadrática Media a Desviaciones en los Pesos: Regla ARRW

En el presente capítulo se mostrará la relación existente entre el error cuadrático medio de una red sujeta a perturbaciones y la sensibilidad estadística. La Sección 4.1 muestra esta relación y presenta una nueva figura de mérito para medir la tolerancia de la red a la que llamaremos "Sensibilidad Cuadrática Media" (SCM). A partir de la evaluación de la SCM es posible predecir el comportamiento de un MLP frente a perturbaciones en sus pesos. Los resultados experimentales que prueban este resultado se presentarán en la Sección 4.2. La sensibilidad cuadrática (SCM) constituye una medida de la tolerancia más conveniente que la sensibilidad estadística media (SSM). Esta nueva magnitud se calcula a partir de la sensibilidad estadística de las neuronas de salida, así que usando la misma filosofía que en el capítulo anterior, en la Sección 4.3 se desarrollará un nuevo algoritmo de entrenamiento, ARRW, que es similar al algoritmo ARR, pero que trata de minimizar la sensibilidad cuadrática media en lugar de la sensibilidad estadística media. ARRW presenta la ventaja de implicar expresiones menos costosas computacionalmente para el cálculo del gradiente manteniendo las mismas prestaciones que el algoritmo ARR. En la Sección 4.4 se mostrarán los resultados experimentales obtenidos usando ARRW sobre distintos MLPs, probando la eficacia del algoritmo propuesto.

4.1 RELACIÓN ENTRE LAS PERTURBACIONES DEL ERROR CUADRÁTICO MEDIO Y LA SENSIBILIDAD ESTADÍSTICA

Las prestaciones en cuanto a aprendizaje se suelen medir usando el error cuadrático medio ε. Durante el aprendizaje cada patrón de entrenamiento p es presentado a la red y se calcula el error cuadrático ε(p) con objeto de modificar los pesos utilizando una técnica de descenso de gradiente. El valor de ε(p) para un perceptrón con M capas viene dado por:

$$\varepsilon(p) = \frac{1}{2} \sum_{i=1}^{N_M} (d_i(p) - y_i^M(p))^2 \tag{4.1}$$

donde N_M indica el número de neuronas de salida, $y_i^M(p)$ es la salida de la neurona i para el patrón de entrada p y $d_i(p)$ es la salida deseada de la neurona i para ese patrón.

El objetivo durante el aprendizaje es el de minimizar el error cuadrático medio. De esta forma, como figura de mérito para indicar la calidad del entrenamiento se utiliza la media de (4.1) sobre un conjunto de N_p patrones, ya sean los del conjunto de entrenamiento (error cuadrático medio de entrenamiento) o del conjunto de test (error cuadrático medio de test).

Así pues, considerando un conjunto con N_p patrones se define el error cuadrático medio (ECM) como:

$$\varepsilon = \frac{1}{N_p} \sum_{p=1}^{N_p} \varepsilon(p) = \frac{1}{2N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} (d_i(p) - y_i^M(p))^2 \tag{4.2}$$

Si la red, previamente entrenada, está sujeta a desviaciones en los pesos, la salida esperada $y_i^M(p)$ variará en una magnitud $\Delta y_i^M(p)$ y por tanto, el error cuadrático medio nominal se modificará y las prestaciones correspondientes al aprendizaje en general se degradadarán.

Si se realiza un desarrollo de Taylor del error cuadrático medio de la red sujeta a desviaciones, ε', respecto del error nominal tras el entrenamiento, ε_0, se obtiene la siguiente expresión

$$\begin{aligned} \varepsilon' &= \varepsilon_0 + \frac{1}{2N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} \frac{\partial \varepsilon(p)}{\partial y_i^M} \Delta y_i^M + \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} \sum_{j=1}^{N_M} \frac{\partial^2 \varepsilon(p)}{\partial y_i^M \partial y_j^M} \Delta y_i^M \Delta y_j^M \\ &= \varepsilon_0 + \frac{1}{2N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} 2(d_i(p) - y_i^M(p)) \Delta y_i^M + \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} 2 (\Delta y_i^M(p))^2 + 0 \end{aligned} \tag{4.3}$$

Si ahora se calcula el valor esperado de ε' y, teniendo en cuenta que, para los modelos de desviación aditivo y multiplicativo considerados en el Capítulo 2, se cumple que:

a) $E[\Delta y_i^M] = 0 \ \forall i,M$ según la Proposición 2.1.

b) $E[(\Delta y_i^M)^2] \simeq (\sigma S_i^M)^2$, según se desprende de la definición de sensibilidad estadística y de la expresión 2.29, siendo S_i^M la sensibilidad de la neurona i perteneciente a la capa de salida M y σ la desviación típica de las perturbaciones en los pesos.

Entonces, considerando a) y b) se puede obtener el valor esperado de ε', E[ε'] expresado como:

$$E[\varepsilon'] = E[\varepsilon_0] + \frac{\sigma^2}{2N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} (S_i^M)^2 \tag{4.4}$$

Nótese que el segundo sumando de la expresión (4.4) depende de las sensibilidades estadísticas de las neuronas de salida y que, por similitud con (4.2), permite definir una nueva figura de mérito a la que llamaremos *Sensibilidad Cuadrática Media de la red* (SCM), cuyo valor viene dado por:

$$SCM = \frac{1}{2N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} (S_i^M)^2 \tag{4.5}$$

Puede observarse que la sensibilidad cuadrática media constituye una magnitud más apropiada para medir la tolerancia a desviaciones de la red que la sensibilidad estadística media, puesto que su relación con la degradación del error cuadrático medio es directa..

Considerando la expresión (4.5), el valor esperado del ECM degradado puede expresarse de forma compacta como:

$$E[\varepsilon'] = \varepsilon_0 + \sigma^2 \, SCM \tag{4.6}$$

De esta manera, conociendo el ECM nominal obtenido tras el entrenamiento (ε_0) y la sensibilidad cuadrática media (SCM) es posible predecir el comportamiento del MLP cuando sus pesos están sujetos a perturbaciones de tipo aditivo o multiplicativo con desviación típica igual a σ. Esto es especialmente relevante en el caso de que el entrenamiento se realice sobre un computador y los pesos deban ser trasladados a una implementación analógica cuyos componentes electrónicos tengan un margen de tolerancia dado, ya que se puede conocer con precisión como se verá degradado el ECM usando (4.6) y particularizando la SCM para el modelo multiplicativo de desviación.

Otro aspecto interesante es que esta expresión muestra el hecho de que una menor sensibilidad cuadrática media implica una menor degradación del error de salida cuando la red es perturbada

por desviaciones en sus pesos, por lo que queda claro que, a igualdad de prestaciones en cuanto a aprendizaje, es mejor escoger aquella que presente una menor sensibilidad. Utilizando la expresión (4.6) se puede discernir incluso entre configuraciones de pesos que, a pesar de proporcionar peores prestaciones nominales en cuanto a error de salida, por tener una menor sensibilidad se van a comportar mejor frente a perturbaciones y van a proporcionar mejores prestaciones si la configuración va a ser trasladada a una implementación física concreta con un margen de tolerancia analógica en sus elementos conocido de antemano.

Debe recordarse que la sensibilidad estadística mide cómo varía la salida de la red cuando varían los valores de sus pesos y no debe confundirse con la *sensibilidad de salida*, tal como se utiliza en diversas publicaciones [EDW95]. La sensibilidad de salida, definida como la derivada de la salida de la red respecto de los valores de los pesos, mide la dependencia de esta salida respecto del valor de tales pesos. En [EDW95] se presenta un procedimiento similar al algoritmo ARR, que incluye un término a la regla de aprendizaje, el cual trata de minimizar la sensibilidad de salida. Esta minimización tiende a hacer independiente la salida del valor de los pesos, por lo que se opone al aprendizaje para de esta manera conseguir una mejor distribución del mismo entre los pesos que forman la red ya que cabe esperar que al penalizar aquellos pesos más influyentes en el reconocimiento de un determinado patrón, se fuerza al resto de los pesos a aprenderlo. Sin embargo, a partir de los resultados mostrados se observa que esto provoca una degradación de las prestaciones de aprendizaje y que, por otro lado, el aumento de la tolerancia a fallos es pequeño frente al obtenido con el algoritmo clásico ARC. Téngase también en cuenta, que aunque un peso no influya en la salida correspondiente para un determinado patrón de entrada, sin embargo una ligera variación en el valor de dicho peso puede producir grandes cambios en la salida.

En el algoritmo ARR sin embargo, lo que se minimiza junto al ECM, es la sensibilidad estadística, es decir, las variaciones en la salida frente *a cambios* en los valores de los pesos y se ha visto que se obtienen valores para los pesos que, sin perjudicar al aprendizaje, son estables frente a variaciones en dichos valores, y por tanto se ven menos afectados cuando están sujetos a perturbaciones. A diferencia del procedimiento anteriormente comentado, el incremento de la tolerancia a fallos no se hace a costa de oponerse al aprendizaje, sino tratando de que todos las neuronas presenten un valor pequeño de sensibilidad estadística. De esta manera se consigue repartir el aprendizaje uniformemente entre las neuronas de la red usando condiciones que consideramos menos restrictivas que las utilizadas por el algoritmo presentado en [EDW95].

En la sección siguiente se describen experimentos que muestran la validez de la expresión (4.6). Para ello se han perturbado los pesos de diversos perceptrones previamente entrenados y se ha comparado el valor obtenido para el ECM usando la expresión (4.6) con el obtenido a partir de su cálculo directo mediante (4.2).

4.2. VALIDACIÓN DE LA EXPRESIÓN PARA EL ERROR CUADRÁTICO SUJETO A PERTURBACIONES

En este apartado se va a comprobar la validez de la expresión (4.6) obtenida en el apartado anterior sobre distintos perceptrones. La expresión (4.6) se puede aplicar a los dos modelos de desviación considerados, el aditivo y el multiplicativo, sin más que sustituir las expresiones correspondientes a la sensibilidad estadística de las neuronas de salida para un modelo u otro en el cálculo de (4.5).

En los experimentos realizados se parte de perceptrones que presentan un error cuadrático medio nominal ε_0 tras el entrenamiento. En cada test, los valores de los pesos de estos perceptrones van a ser modificados según el modelo aditivo o multiplicativo, aplicando a cada peso una perturbación aleatoria con desviación típica σ. Así, para un valor determinado de σ, se calcula experimentalmente el valor del ECM, ε', a partir de (4.2). El valor esperado del ECM degradado, $E[\varepsilon']$, se calcula como la media de ε' sobre un determinado número de tests, donde cada test consiste en una perturbación aleatoria de todos los pesos del MLP. Este último resultado se compara con el resultado de la evaluación directa de la expresión (4.6) para el valor de σ considerado.

Como ejemplos se han tomado un perceptrón que aproxima la función seno [SUD94] y otro que predice la serie de Mackey-Glass [WAN94]. La estructura de ambos aparece descrita en la Tabla 3.3 del Capítulo 3. Recordemos que el aproximador del seno recibe como entrada la variable independiente y proporciona el valor del seno de dicha variable en su salida. El predictor de la serie temporal recibe a la entrada el valor de la serie para tres valores consecutivos y proporciona a la salida el valor de la siguiente iteración.

En primer lugar se van a mostrar resultados usando el modelo de desviación aditivo en 4.2.1 y, a continuación, los obtenidos usando el modelo multiplicativo en 4.2.2.

4.2.1. Modelo de desviación aditivo

Tal como se ha descrito anteriormente, los valores nominales de los pesos obtenidos tras el entrenamiento son perturbados en cada test. De este modo, acorde con el modelo de desviación aditivo, en cada test todos los pesos w_{ij}^m toman un valor $w_{ij}^m \pm \delta$ siendo δ una variable aleatoria con desviación típica σ y media cero. El valor esperado del error cuadrático medio degradado de la red para cada valor de σ se calcula como la media de este error sobre 100 tests, donde cada test implica una perturbación aleatoria de los pesos según se ha descrito.

La Tabla 4.1 muestra los valores obtenidos para el ECM nominal (ε_0) y la SCM. Ambas magnitudes se han calculado sobre el conjunto de patrones de test una vez finalizado el entrenamiento (los patrones de test son distintos de los usados durante el entrenamiento).

Tabla 4.1. Error cuadrático medio y sensibilidad cuadrática media nominales.

	Aproximador	Predictor
ε_0	0.001	0.0004
SCM	0.1922	0.1347

En la Tabla 4.2 se presentan los valores obtenidos mediante el cálculo directo de E[ε'] junto con el intervalo de confianza al 95% , y los valores predichos mediante la expresión (4.6) para distintos valores de σ. Se aprecia que el valor predicho se aproxima fielmente al valor experimental obtenido mediante el cálculo directo, tanto más cuanto menor sea el valor de σ. Recordemos que para el cálculo de la sensibilidad estadística se supusieron desviaciones pequeñas y que, por tanto, al crecer las desviaciones el modelo deja de ser válido. Aún así constituye una medida bastante precisa.

Tabla 4.2. Comparación entre E[ε'] predicho y experimental.

	Aproximador		Predictor	
σ	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)
0.01	1.029	1.030 ± 0.011	0.444	0.444 ± 0.015
0.02	1.087	1.097 ± 0.028	0.484	0.499 ± 0.032
0.03	1.183	1.296 ± 0.094	0.552	0.593 ± 0.056
0.04	1.317	1.363 ± 0.090	0.646	0.683 ± 0.091
0.05	1.49	1.632 ± 0.145	0.767	1.068 ± 0.185
0.06	1.702	1.864 ± 0.203	0.915	1.168 ± 0.264
0.07	1.952	1.993 ± 0.212	1.09	1.274 ± 0.188
0.08	2.24	2.218 ± 0.241	1.293	1.585 ± 0.284

En las figuras 4.1 y 4.2 se representan los valores predichos y calculados mediante simulación para distintos valores de σ. Los valores calculados experimentalmente se muestran con el intervalo de confianza al 95% obtenido sobre 100 muestras. La Figura 4.1 se refiere al aproximador de la función seno. Puede apreciarse que la curva obtenida mediante la expresión (4.6) se aproxima bastante a la experimental hasta un valor de σ igual aproximadamente a 0.5. Téngase en cuenta que a partir de ese valor de σ el ECM se ha hecho unas 50 veces mayor que el obtenido tras el entrenamiento por lo que las prestaciones se degradan fuertemente y posiblemente el MLP se deba descartar.

El caso del predictor de la serie temporal puede apreciarse en la Figura 4.2. Se observa un comportamiento similar al descrito anteriormente si bien la aproximación del valor predicho al calculado es válido hasta un valor de σ igual a 0.6, en cuyo caso el error cuadrático medio se ha hecho unas 100 veces superior al nominal.

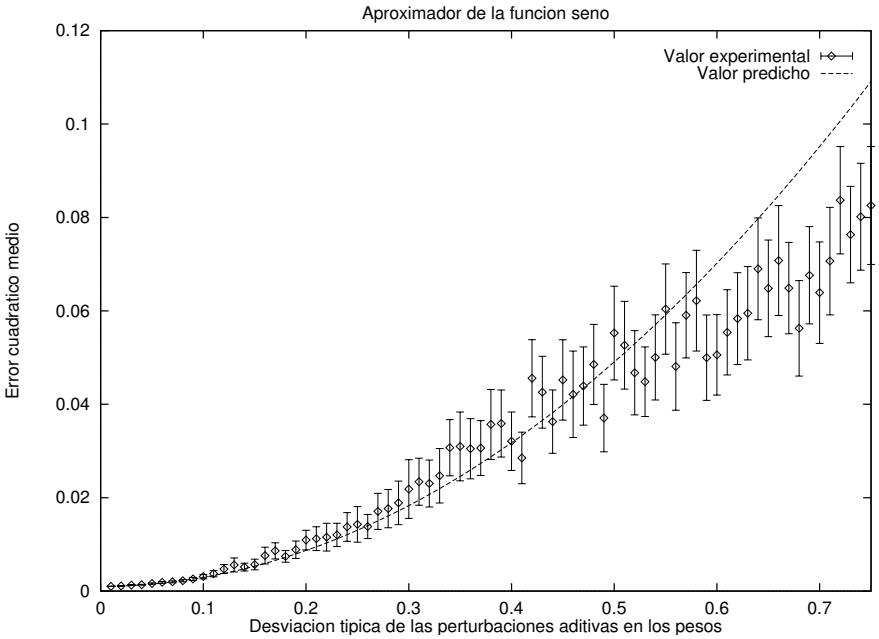

Figura 4.1. Error cuadrático medio experimental y predicho frente a σ.

En ambos casos se muestra la validez de la expresión (4.6) para el caso de desviaciones aditivas, al menos hasta un punto crítico en el que las prestaciones de la red se han degradado bastante debido a la magnitud de las desviaciones en sus pesos. Recordemos que el modelo de error aditivo supone sumar al valor del peso una determinada perturbación y sus efectos no son lineales. Su efecto puede ser tal que no sólo varíe la magnitud del peso sino incluso su signo con lo cual su contribución a la función de activación puede ser crítico. De esta manera, predecir el error cuadrático medio para perturbaciones grandes es muy difícil. Aún así, los resultados

obtenidos son fiables para perturbaciones que, aunque no grandes en magnitud, si producen degradaciones tan importantes que convierten el MLP en inservible, por lo que para efectos prácticos la relación (4.6), en este caso particularizada para desviaciones de tipo aditivo, constituye una medida apropiada predecir el ECM, y por tanto, el uso de la SCM como medida de la tolerancia del MLP es apropiado.

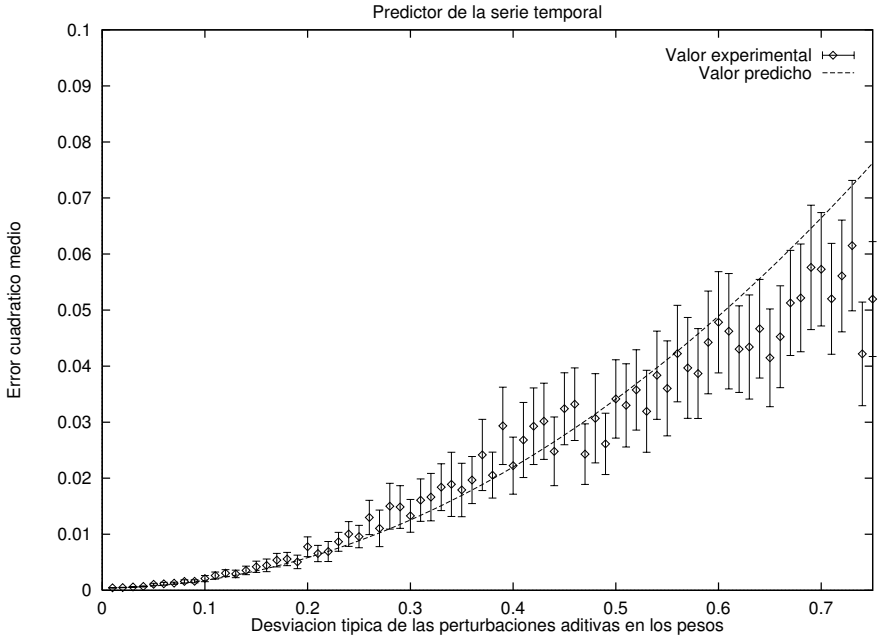

Figura 4.2. Error cuadrático medio experimental y predicho frente a σ.

4.2.2 Modelo de desviación multiplicativo

En el caso de desviaciones multiplicativas en los pesos en cada test todos los pesos w_{ij}^m se modifican a valores $w_{ij}^m (1+\delta)$ siendo δ una variable aleatoria con desviación típica σ y media cero. El valor esperado del error cuadrático medio de la red perturbada para cada valor de σ se calcula como la media de este error sobre 100 tests, donde en cada test se realiza una perturbación aleatoria de todos sus pesos según el modelo multiplicativo. Este modelo de error es especialmente interesante por su relación con el margen de tolerancia en el caso de componentes analógicas. El valor de σ representa el valor de la tolerancia, por lo que cuando los pesos se obtienen mediante simulación en un ordenador y han de ser traspasados a una implementación analógica cuyos componentes presentan un valor de tolerancia conocido, la expresión (4.6) permite predecir cuales serán las prestaciones esperadas para dicha implementación.

La Tabla 4.3 muestra los valores obtenidos para el error cuadrático medio nominal ε_0 y la sensibilidad cuadrática media tras el entrenamiento. Se puede observar que el valor de SCM es de 4.2968 en el caso del aproximador del seno lo que va a implicar una rápida degradación de las prestaciones al crecer el valor de las perturbaciones en los pesos. En cambio, para el predictor, S_{cm} toma un valor de 0.4597 por lo que sus prestaciones serán degradadas en menor cuantía.

Tabla 4.3. Error cuadrático medio y sensibilidad cuadrática media nominales.

	Aproximador	Predictor
ε_0	0.001	0.0004
SCM	4.2968	0.4597

La Tabla 4.4 muestra algunos ejemplos de valores predichos y simulados para distintos valores de σ. Como se ha mencionado antes σ se puede identificar como la tolerancia analógica y por ello, en la Tabla 4.4 se muestra el valor de la misma expresada en %. El valor esperado del ECM de la red perturbada, E[ε'], se presenta con el correspondiente intervalo de confianza al 95%. Este valor se ha calculado sobre 100 muestras.

Tabla 4.4. Comparación entre E[ε'] predicho y calculado.

	Aproximador		Predictor	
Tolerancia (%)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)
0.6	1.164	1.155 ± 0.058	0.447	0.440 ± 0.008
1.2	1.628	1.459 ± 0.127	0.496	0.502 ± 0.020
1.8	2.402	1.941 ± 0.214	0.579	0.610 ± 0.064
2.4	3.485	2.934 ± 0.450	0.633	0.677 ± 0.103
3	4.877	4.045 ± 0.823	0.844	0.853 ± 0.140
3.6	6.578	5.507 ± 1.465	1.026	0.997 ± 0.160
4.2	8.589	7.806 ± 1.633	1.241	1.235 ± 0.233
4.8	10.91	11.71 ± 3.07	1.489	1.420 ± 0.247

En el caso del aproximador, al degradarse el ECM rápidamente, el valor predicho difiere en mayor cuantía del valor experimental aunque, sin embargo, está dentro del intervalo de confianza. En cambio, en el caso del predictor los valores predichos y experimentales prácticamente coinciden.

En las Figuras 4.3 y 4.4 se representan los valores predichos y calculados mediante simulación para distintos valores de σ para el aproximador de la función seno y el predictor de la serie temporal. Se muestran también los márgenes de error obtenidos para cada punto experimental.

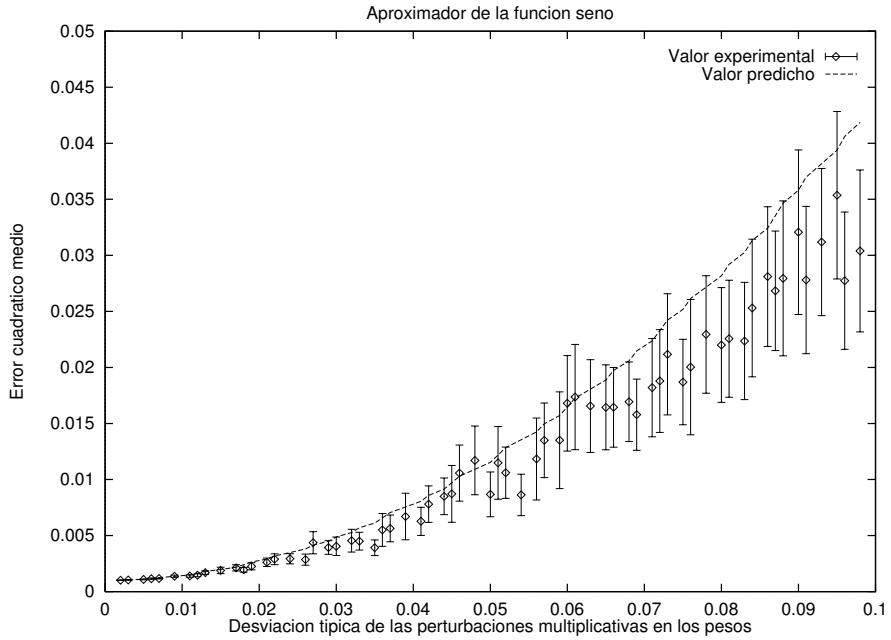

Figura 4.3. Error cuadrático medio experimental y predicho frente a σ.

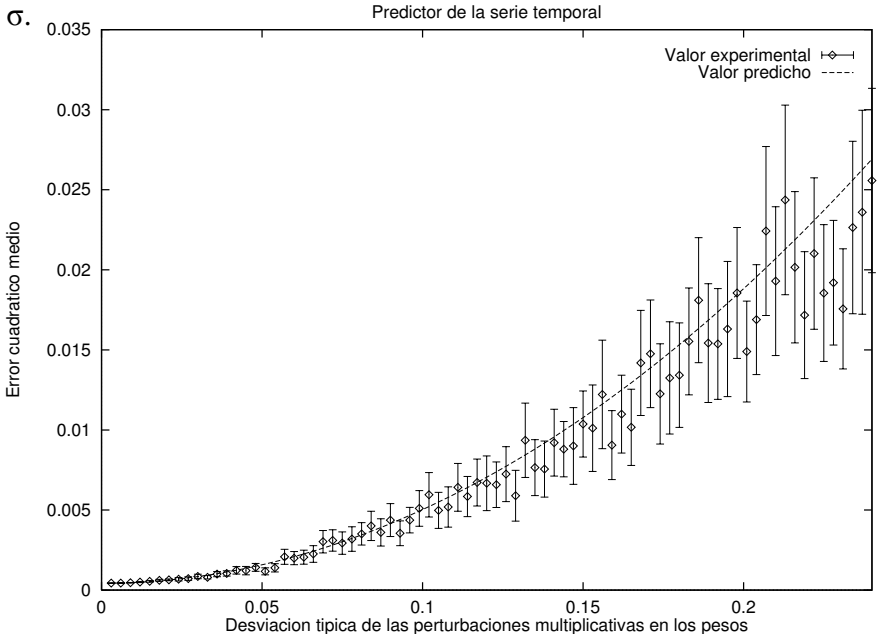

Figura 4.4. Error cuadrático medio simulado y predicho frente a σ.

En el caso del aproximador (Figura 4.3) puede comprobarse que la expresión (4.6) constituye una buena aproximación para valores de σ de hasta 0.08 aproximadamente. Para dicho valor de σ las prestaciones ya se han degradado bastante y la aproximación no sigue siendo tan exacta. En cambio, en el caso del predictor (Figura 4.4) la expresión (4.6) es válida hasta valores de σ del orden de 0.24, o lo que es lo mismo, para un valor de tolerancia en los componentes analógicos del orden del 24%. Nótese que la validez de la expresión (4.6) depende de la magnitud de la degradación y, por tanto del valor obtenido para la sensibilidad cuadrática media, lo cual indica que cuanto menor sea el valor de la SCM mayor será la validez de la expresión (4.6) ya que la red se degrada menos. Por otra parte, para valores de σ en los cuales la aproximación deja de ser válida, el ECM se ha degradado tanto que el MLP posiblemente sea inservible. De esta forma, de manera idéntica a como se concluyó en el Apartado 4.2.1, en la práctica, la tolerancia de MLPs sujetos a desviaciones multiplicativas puede ser evaluada fielmente mediante la SCM particularizada para este tipo de desviaciones, y mediante (4.6) puede predecirse la degradación correspondiente al ECM para un valor determinado en tales desviaciones.

De cualquier forma en las Figuras 4.1 a 4.4 se aprecia que el valor predicho constituye una cota superior, por lo que el ECM esperado no va ser mayor que el valor predicho por lo que aún para grandes desviaciones la SCM es una figura de mérito razonable para evaluar la tolerancia de un MLP.

4.3. MINIMIZACIÓN DE LA SENSIBILIDAD CUADRÁTICA MEDIA. ALGORITMO ARRW.

En el Capítulo 3 se presentó el algoritmo ARR y se mostró su eficacia para obtener configuraciones de pesos que, sin afectar a las prestaciones de error de salida, presentaban una menor sensibilidad estadística y, por tanto, una mayor tolerancia a desviaciones en los pesos. En el Apartado 4.1 se ha obtenido una relación directa entre el error cuadrático medio y una nueva magnitud a la que hemos llamado sensibilidad cuadrática media (SCM). La SCM se obtiene a partir de la sensibilidad estadística de las neuronas de salida y por tanto, parece obvio que la minimización de una implica la minimización de la otra. Además, puesto que la SCM es una medida más apropiada de la tolerancia a desviaciones, por su relación directa con el ECM de la red sujeta a desviaciones, parece más apropiado desarrollar un algoritmo que minimice dicha magnitud.

Como la sensibilidad se calcula como el producto de una raíz cuadrada por la derivada de la función de activación según las expresiones (2.27) y (2.28), en el valor de la SCM desaparece la raíz cuadrada y, por tanto, las expresiones del gradiente de la misma son más sencillas con lo

que su costo computacional se reduce también.

En este apartado se va a introducir una regla de aprendizaje similar a la presentada en el Capitulo 3, en la que además del error cuadrático medio se va a minimizar la sensibilidad cuadrática media a desviaciones en los pesos. Este algoritmo se puede considerar una técnica de regularización explícitamente especificada, ya que junto al error cuadrático medio se va a minimizar un estabilizador de segundo orden añadido explícitamente [MAO93].

4.3.1 Algoritmo de Retropropagación Robusto frente a desviaciones en los pesos (ARRW)

En el algoritmo ARC se modifican los valores de los pesos siguiendo una algoritmo de descenso de gradiente en el cual se busca la minimización del ECM que presenta la red a un conjunto de patrones de entrada. De esta manera, cada vez que se presenta un patrón, se calcula el error cuadrático entre las salidas obtenidas y deseadas para el mismo, y los pesos se modifican para minimizar dicho error cuadrático.

Vista la relación obtenida entre la degradación del error cuadrático medio y la sensibilidad cuadrática media cuando los pesos se perturban, se define el error cuadrático instantáneo en presencia de perturbaciones ε'(p) como:

$$\varepsilon'(p) = \frac{1}{2}\sum_{i=1}^{N_M}(d_i(p) - y_i^M(p))^2 + \frac{\sigma^2}{2}\sum_{i=1}^{N_M}(S_i^M(p))^2 \qquad (4.7)$$

donde $y_i^M(p)$ y $d_i(p)$ constituyen las salidas obtenidas y deseadas de las neuronas de salida (capa M) para el patrón de entrada p y $S_i^M(p)$ es la sensibilidad estadística de la neurona de salida i para el patrón de entrada p. Nótese que la expresión (4.7) indica que al error cuadrático nominal (en ausencia de perturbaciones) se le suma un término que depende de la sensibilidad cuadrática y que mide lo que variará el error cuadrático si se producen desviaciones de orden σ.

La minimización de la expresión (4.7) implica la reducción del error cuadrático así como de la sensibilidad cuadrática de la red. Nótese la similitud con la regla de aprendizaje robusto ARR, propuesta en el Capítulo 3. Si identificamos el valor de σ^2 con el factor de robustez γ, y teniendo en cuenta que la sensibilidad estadística es una magnitud positiva obtenemos una regla totalmente análoga.

Sin embargo en la nueva regla propuesta (ARRW) se minimiza la siguiente expresión:

$$\varepsilon'(p) = \frac{1}{2}\sum_{i=1}^{N_M}(d_i(p) - y_i^M(p))^2 + \frac{\gamma}{2}\sum_{i=1}^{N_M}(S_i^M(p))^2 \qquad (4.8)$$

donde a γ le seguimos llamando factor de robustez y varía dinámicamente durante el entrenamiento según la expresión (3.5) con objeto de mantener la jerarquía de objetivos a minimizar durante el proceso de aprendizaje y asegurar la convergencia de la solución.

A continuación se va a proceder al cálculo del gradiente de la sensibilidad cuadrática para perceptrones con una sóla capa oculta considerando los modelos aditivo y multiplicativo. La sensibilidad cuadrática instantánea se define como:

$$S_c = \frac{1}{2} \sum_{i=1}^{N_M} (S_i^M)^2 \tag{4.9}$$

4.3.2. Cálculo del gradiente de la sensibilidad cuadrática usando el modelo aditivo

El objetivo es obtener el gradiente de la expresión (4.8) para un perceptrón de tres capas y considerando el modelo aditivo de desviaciones en los pesos. De forma similar a como se hizo en el capítulo anterior se va a calcular el gradiente para las neuronas de salida en primer lugar y, a continuación el gradiente para los pesos de la capa oculta.

Recordemos que la sensibilidad estadística de una neurona i de la capa oculta (capa 1) viene dada por la siguiente expresión:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{j=1}^{N_0} (y_j^0)^2} \tag{4.10}$$

y la sensibilidad de las neuronas de la capa de salida (capa 2) por:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{ij}^2 S_j^1)^2)} \tag{4.11}$$

Como se trata de minimizar la expresión (4.9), si se sustituye la expresión (4.11) en (4.9) se obtiene:

$$S_c = \frac{1}{2} \sum_{r=1}^{N_2} \left((\partial f_r^2)^2 \sum_{j=1}^{N_1} ((y_j^1)^2 + (w_{rj}^2 S_j^1)^2) \right) \tag{4.12}$$

4.3.2.1 Cálculo del gradiente para los pesos de las neuronas de salida

Considerando un peso w_{ik}^2 que conecta una neurona de salida i con una neurona k de la capa oculta, se tiene que:

$$\begin{aligned}(\nabla S_c)_{ik}^2 &= \frac{1}{2}\,\frac{d}{dw_{ik}^2}\sum_{r=1}^{N_2}(S_r^2)^2 = \frac{1}{2}\,\frac{d(S_i^2)^2}{dw_{ik}^2} \\ &= \frac{1}{2}\,(\partial f_i^2)^2\,\frac{d}{dw_{ik}^2}\sum_{j=1}^{N_1}((y_j^1)^2 + (w_{ij}^2 S_j^1)^2) \\ &= \frac{1}{2}\,(\partial f_i^2)^2\;2\;w_{ik}^2(S_k^1)^2 = (\partial f_i^2)^2\;w_{ik}^2(S_k^1)^2\end{aligned} \tag{4.13}$$

Obsérvese que al igual que se hizo en el Apartado 3.3 se ha tomado la derivada de la función de activación como una constante. A la vista de la expresión (3.12) se aprecia la similitud con la expresión (4.13) si bien esta última es más sencilla y menos costosa computacionalmente por haber una división menos.

4.3.2.2 Cálculo del gradiente para los pesos de las neuronas de la capa oculta

Considerando un peso w_{ik}^1 que conecta la neurona i de la capa oculta con la entrada k, se va a proceder al cálculo del gradiente de la sensibilidad de la red respecto de los pesos de la capa oculta de forma similar a como se ha realizado en el caso de los pesos pertenecientes a las neuronas de la capa de salida:

$$
\begin{aligned}
(\nabla S_c)^1_{ik} &= \frac{d}{dw^1_{ik}} \left(\frac{1}{2} \sum_{r=1}^{N_2} (S^2_r)^2\right) \\
&= \frac{1}{2} \sum_{r=1}^{N_2} (\partial f^2_r)^2 \ \frac{d}{dw^1_{ik}} \left(\sum_{j=1}^{N_1} ((y^1_j)^2 + (w^2_{rj} S^1_j)^2)\right) \\
&= \frac{1}{2} \sum_{r=1}^{N_2} (\partial f^2_r)^2 \sum_{j=1}^{N_1} \left(2\ y^1_j \ \frac{dy^1_j}{dw^1_{ik}} + 2\ (w^2_{rj})^2\ S^1_j \ \frac{dS^1_j}{dw^1_{ik}} \right) \qquad (4.14) \\
&= \sum_{r=1}^{N_2} (\partial f^2_r)^2 \ (y^1_i \ \frac{dy^1_i}{dw^1_{ik}} + 0) = y^1_i \ \frac{dy^1_i}{dz^1_i} \frac{dz^1_i}{dw^1_{ik}} \sum_{r=1}^{N_2} (\partial f^2_r)^2 \\
&= \partial f^1_i \ y^0_k \ y^1_i \sum_{r=1}^{N_2} (\partial f^2_r)^2
\end{aligned}
$$

En resumen, los pesos de las neuronas deben ser modificados según indican las expresiones (4.13) y (4.14) para minimizar la sensibilidad cuadrática cuando se considera un modelo aditivo de desviaciones en los pesos. Obsérvese de nuevo la similitud con las expresiones del gradiente de la sensibilidad estadística obtenidas en el Apartado 3.2.1.

4.3.3. Cálculo del gradiente de la sensibilidad cuadrática usando el modelo multiplicativo

En el Capítulo 2 se obtuvieron las expresiones de la sensibilidad estadística cuando se considera un modelo de desviación multiplicativo y éstas vienen dadas para las neuronas de la capa oculta por:

$$S^1_i = \partial f^1_i \sqrt{\sum_{j=1}^{N_0} (y^0_j \ w^1_{ij})^2} \qquad (4.15)$$

y para las neuronas de la capa de salida por:

$$S^2_i = \partial f^2_i \sqrt{\sum_{j=1}^{N_1} (w^2_{ij})^2 \ ((y^1_j)^2 + (S^1_j)^2)} \qquad (4.16)$$

para la neurona de salida en MLPs con una capa oculta.

4.3.3.1. Cálculo del gradiente para los pesos de las neuronas de salida

Considerando un peso w_{ik}^{2} que conecta la neurona de salida i con la neurona k de la capa oculta, se obtiene:

$$\begin{aligned}(\nabla S_c)_{ik}^2 &= \frac{1}{2}\,\frac{d}{dw_{ik}^2}\sum_{r=1}^{N_2}(S_r^2)^2 = \frac{1}{2}\,\frac{d(S_i^2)^2}{dw_{ik}^2}\\ &= \frac{1}{2}\,(\partial f_i^2)^2\,\frac{d}{dw_{ik}^2}\sum_{j=1}^{N_1}(w_{ij}^2)^2\,((y_j^1)^2 + (S_j^1)^2)\\ &= \frac{1}{2}\,(\partial f_i^2)^2\;2\;w_{ik}^2((y_k^1)^2 + (S_k^1)^2)\\ &= (\partial f_i^2)^2\;w_{ik}^2\;((y_k^1)^2 + (S_k^1)^2)\end{aligned} \tag{4.17}$$

4.3.3.2. Cálculo del gradiente para los pesos de las neuronas de la capa oculta

Considerando un peso w_{ik}^{1} de una neurona de la capa oculta i mediante el cual se conecta a la entrada k, se debe calcular la siguiente expresión:

$$\begin{aligned}(\nabla S_c)_{ik}^1 &= \frac{1}{2}\,\frac{d}{dw_{ik}^1}\,(\sum_{r=1}^{N_2}(S_r^2)^2) = \frac{1}{2}\sum_{r=1}^{N_2}(\partial f_r^2)^2\frac{d}{dw_{ik}^1}\sum_{j=1}^{N_1}(w_{rj}^2)^2\,((y_j^1)^2 + (S_j^1)^2)\\ &= \frac{1}{2}\sum_{r=1}^{N_2}(\partial f_r^2)^2\sum_{j=1}^{N_1}(w_{rj}^2)^2\left(2\;y_j^1\,\frac{dy_j^1}{dw_{ik}^1} + 2\;S_j^1\,\frac{dS_j^1}{dw_{ik}^1}\right)\\ &= \sum_{r=1}^{N_2}(\partial f_r^2)^2\;(w_{ri}^2)^2\;(y_i^1\,\frac{dy_i^1}{dw_{ik}^1} + S_i^1\,\frac{dS_i^1}{dw_{ik}^1})\\ &= \left(y_i^1\;\partial f_i^1\;y_k^0 + S_i^1\partial f_i^1\;\frac{w_{ik}^1(y_k^0)^2}{\sqrt{\sum_{j=1}^{N_0}(y_j^0\;w_{ij}^1)^2}}\right)\sum_{r=1}^{N_2}(\partial f_r^2)^2\;(w_{ri}^2)^2\\ &= \partial f_i^1\;y_k^0\;(y_i^1 + w_{ik}^1\;\partial f_i^1\;y_k^0)\sum_{r=1}^{N_2}(\partial f_r^2)^2\;(w_{ri}^2)^2\end{aligned} \tag{4.18}$$

De forma similar a como se hizo en la regla ARR los valores calculados con las expresiones (4.16) y (4.17) se utlizan durante el aprendizaje tras presentar un patrón de entrada para modificar los valores de los pesos. Estos valores se ponderan con el valor del factor de robustez γ para mantener debidamente la jerarquía de objetivos a minimizar. Las expresiones (4.16) y (4.17) son muy similares a las obtenidas para el gradiente de la sensibilidad estadística del Capítulo 3, y por tratarse la sensibilidad de una magnitud positiva las prestaciones de la nueva regla, donde se minimiza la sensibilidad cuadrática en lugar de la sensibilidad estadística deben ser por tanto similares, si bien el uso de esta nueva regla (ARRW) implica un cierto ahorro computacional por tratarse de expresiones más sencillas. Además, la SCM es una medida de la tolerancia más apropiada ya que está relacionada de forma directa con la degradación del ECM.

La regla ARRW se ha obtenido a partir de la relación obtenida entre el error cuadrático de la red sujeta a perturbaciones y la sensibilidad estadística tal como se aprecia en la expresión (4.6). La minimización de la expresión (4.7), por tanto, permite obtener configuraciones de pesos que presentan un error cuadrático medio estable frente a perturbaciones en los pesos.

En la Sección 4.4 se mostrarán los resultados obtenidos a partir de la regla ARRW comparados con los que se obtienen usando el algoritmo clásico (ARC).

4.4. VALIDACIÓN DE LA REGLA ARRW

El algoritmo ARRW se ha evaluado en tres MLPs. Los MLPs considerados se describen en la Tabla 4.5. Los dos primeros se corresponden con el aproximador y el predictor ya considerados en pruebas anteriores. El tercero es un clasificador que resuelve el problema de Hart [HAR68]. Dicho clasificador dispone de 2 neuronas de entrada, 11 neuronas en la capa oculta y 2 neuronas de salida, y debe clasificar un punto a partir de sus coordenadas cartesianas en 2 clases, de forma que se toma como salida la clase correspondiente a la neurona de salida ganadora. Las dos clases A y B están confinadas dentro de un cuadrado, tal como aparece en la Figura 4.5 correspondiendo una clase a la zona izquierda (A) y la otra a la derecha (B).

Tabla 4.5. Descripción de los MLPs usados como ejemplo.

Problema	Neuronas/capa	Patrones entrenamiento/test	Nº épocas
Aproximador	1-11-1	100/25	2000
Predictor	3-11-1	500/127	1000
Clasificador	2-11-2	400/100	2000

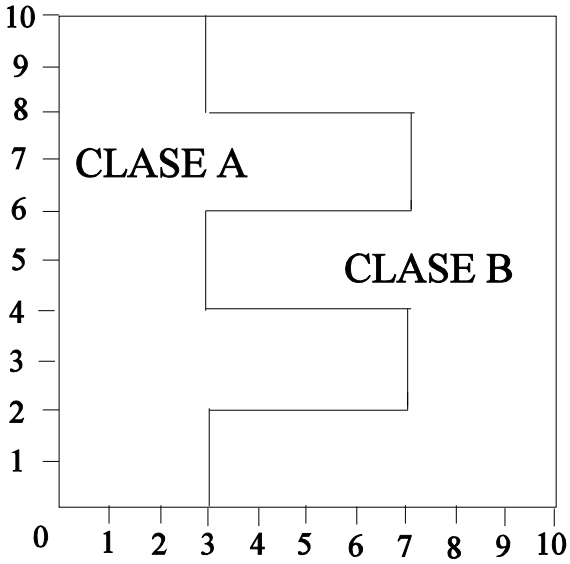

Figura 4.5. Problema de Hart.

En primer lugar, se mostrarán los resultados de comparar la regla ARRW para el modelo aditivo de desviación con el algoritmo clásico en 4.4.1. En el Apartado 4.4.2 se muestran los resultados correspondientes al modelo multiplicativo.

4.4.1. Comparación del algoritmo clásico con ARRW para perturbaciones aditivas

Los tres ejemplos considerados se han entrenado usando el algoritmo clásico y la regla ARRW particularizada para el caso de desviaciones aditivas usando las expresiones (4.12) y (4.13). Se han utilizado distintos valores de la constante K del factor de robustez en ARRW, coincidiendo con el algoritmo ARC cuando K es cero. Para cada valor de K se han realizado 40 entrenamientos partiendo de distintos valores iniciales para los pesos.

4.4.1.1. Resultados para el aproximador de la función seno

La Tabla 4.6 muestra los resultados de error cuadrático medio (ECM), sensibilidad estadística media (SSM) y sensibilidad cuadrática media (SCM) obtenidos con el algoritmo ARRW para distintos valores de la constante K del factor de robustez. Estos resultados se muestran con un intervalo de confianza al 95 %.

Tabla 4.6 Resultados usando el modelo aditivo para el aproximador.

Constante K	Error cuadrático medio (x 1e-3)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	0.796 ± 0.112	0.440 ± 0.011	0.125 ± 0.007
1	0.681 ± 0.102	0.401 ± 0.009	0.103 ± 0.005
2	0.624 ± 0.094	0.375 ± 0.008	0.090 ± 0.004
3	0.616 ± 0.101	0.366 ± 0.006	0.086 ± 0.003
4	0.614 ± 0.097	0.348 ± 0.004	0.078 ± 0.002
5	0.624 ± 0.162	0.346 ± 0.005	0.076 ± 0.004
6	0.623 ± 0.087	0.333 ± 0.005	0.071 ± 0.002
7	0.785 ± 0.106	0.328 ± 0.004	0.069 ± 0.002
8	0.660 ± 0.079	0.318 ± 0.005	0.064 ± 0.002
9	1.094 ± 0.539	0.306 ± 0.012	0.061 ± 0.004
10	1.023 ± 0.550	0.305 ± 0.011	0.060 ± 0.003

Para realizar la comparación entre ARC y ARRW vamos a centrarnos en el caso en que K es igual a 8. Se puede apreciar que en el caso de usar ARC se obtiene un ECM igual a 0.000796, superior al obtenido con ARRW (0.000660). Además, con ARRW se obtiene una SCM de 0.064 frente a la de ARC que es igual a 0.125, es decir la mitad aproximadamente. Vista la validez de la expresión (4.6) ésto nos indica que el perceptrón entrenado con ARRW degradaría su error cuadrático en una magnitud aproximadamente igual a la mitad de lo que lo haría si se ha entrenado con ARC en presencia de desviaciones aditivas.

Se aprecia en la Tabla 4.6 que usando el algoritmo ARRW se obtienen en general mejores prestaciones en cuanto a aprendizaje respecto del algoritmo clásico (menor valor del ECM). Se observa que la sensibilidad disminuye al aumentar el factor de robustez. Sin embargo, cuando K se hace superior a 8, aunque la sensibilidad sigue disminuyendo, las prestaciones de aprendizaje son menores debido a que las modificaciones debidas al gradiente de la sensibilidad prevalecen sobre las del gradiente del error cuadrático.

Teniendo en cuanta que la sensibilidad estadística es una magnitud positiva, hemos considerado la equivalencia de minimizar dicho valor (ARR) con la minimización de la sensibilidad cuadrática (ARRW). En la Figura 4.6 se representa la sensibilidad cuadrática media en función de la sensibilidad estadística media. Puede observarse que hay una dependencia

aproximadamente lineal entre ambas, por lo que así queda probada experimentalmente esta equivalencia. De esta manera, los resultados y conclusiones obtenidos con ARR mostrados en el Capítulo 3 se pueden aceptar como válidos en el caso de ARRW.

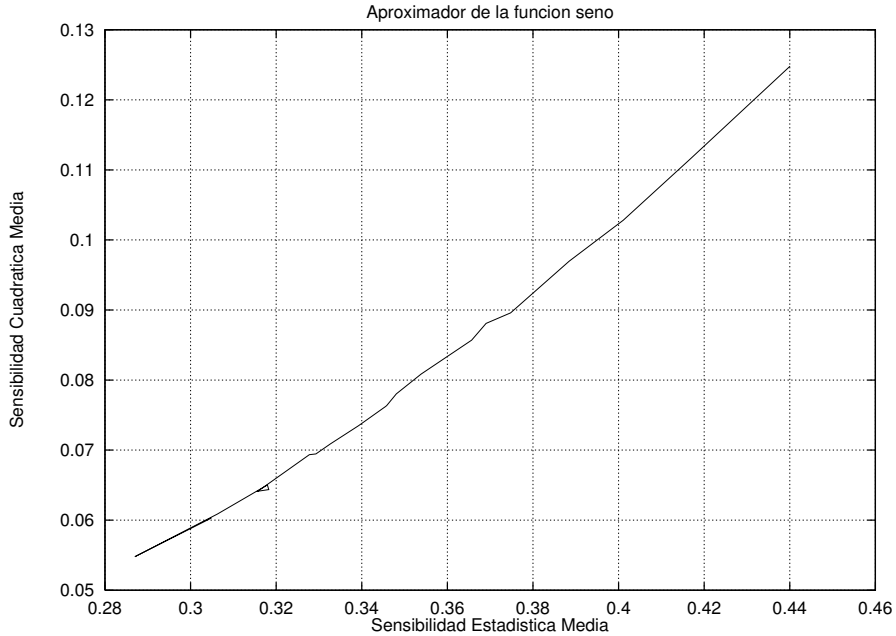

Figura 4.6. Sensibilidad cuadrática frente a sensibilidad estadística.

Las Figuras 4.7 y 4.8 representan, en función de K, el error cuadrático medio y la sensibilidad cuadrática media, respectivamente. En ambas figuras se muestra también la desviación típica muestral. Se observa en la Figura 4.7, igual que en la Tabla 4.6, que el ECM obtenido mediante ARRW es comparable, e incluso inferior, al obtenido con ARC, mientras que la SCM es bastante menor respecto de la obtenida con el algoritmo ARC (K = 0.0) tal como se muestra en la Figura 4.8. Cuando K crece por encima de 8, sin embargo, el error se degrada aunque la sensibilidad sigue disminuyendo. A partir de este punto se puede observar que la desviación típica es bastante mayor, debido a que la solución puede quedar estancada en mínimos locales no demasiado satisfactorios debido a que el parámetro γ no jerarquiza apropiadamente los objetivos a minimizar.

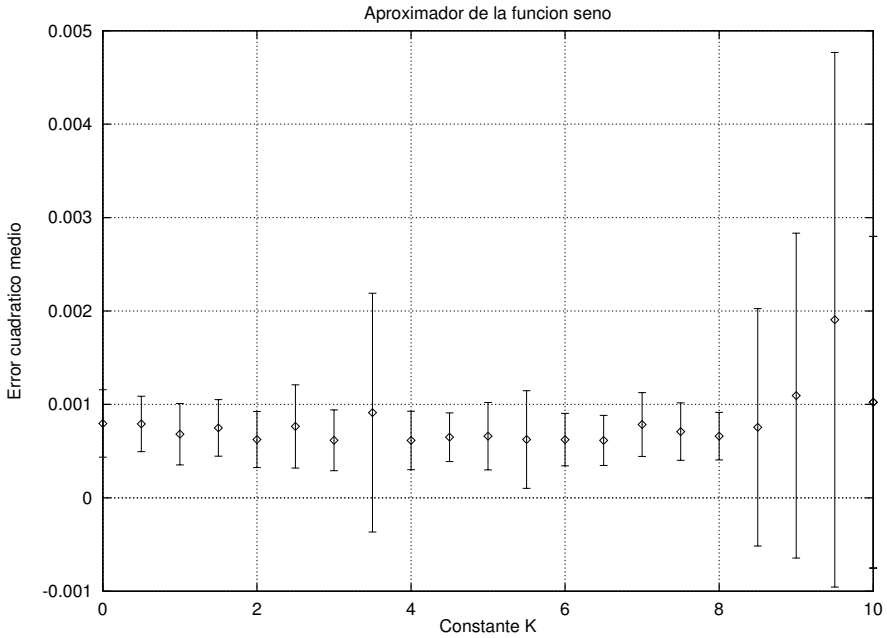

Figura 4.7. Error cuadrático medio frente a K.

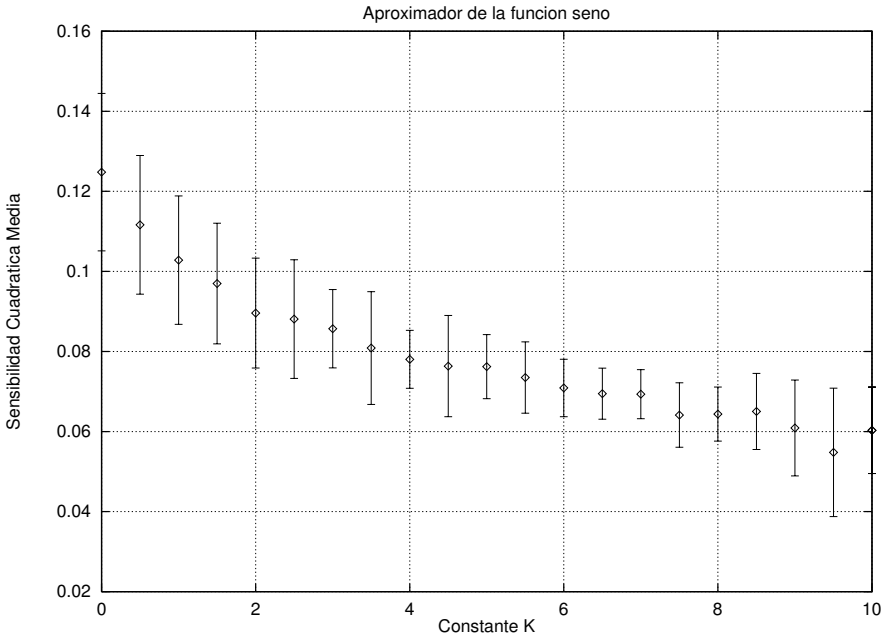

Figura 4.8. Sensibilidad cuadrática frente a K.

4.4.1.2. Resultados para el predictor de la serie de Mackey-Glass

Para esta aplicación de predicción de series temporales se han observado similares resultados a los del ejemplo anterior. La Tabla 4.7 muestra los valores obtenidos para el error cuadrático medio, sensibilidad estadística media y sensibilidad cuadrática media usando distintos valores de la constante del factor de robustez en el algoritmo ARRW. Para cada valor de K se han realizado 40 tests y los resultados se presentan con un intervalo de confianza al 95%.

Los resultados del algoritmo clásico corresponden a la fila donde K=0. Si se compara con el caso en que K=100 se observa que además de obtener una SCM 3.3 veces inferior con ARRW (0.0334 frente a 0.1107 con ARC), se obtiene también un ECM menor (1.57 veces mayor para ARC que para ARRW). Es decir, no sólo se mejora la tolerancia a fallos de la red sino que también se obtiene en este caso cierta mejora en el aprendizaje. Este último hecho, no obstante, tal como se vio a partir de los análisis realizados en el capítulo anterior, es fruto del azar, pero pone de manifiesto que el uso de ARRW como regla de aprendizaje no degrada las prestaciones en cuanto al mismo.

Tabla 4.7. Resultados usando el modelo aditivo para el predictor.

Constante K	Error cuadrático medio (x 1e-4)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	1.13 ± 0.08	0.448 ± 0.009	0.1107 ± 0.0044
10	0.78 ± 0.03	0.298 ± 0.002	0.0475 ± 0.0007
20	0.75 ± 0.01	0.281 ± 0.001	0.0421 ± 0.0004
30	0.75 ± 0.02	0.272 ± 0.002	0.0395 ± 0.0005
40	0.74 ± 0.01	0.268 ± 0.002	0.0382 ± 0.0006
60	0.72 ± 0.01	0.262 ± 0.001	0.0363 ± 0.0004
80	0.72 ± 0.01	0.255 ± 0.001	0.0347 ± 0.0004
100	0.72 ± 0.01	0.251 ± 0.001	0.0334 ± 0.0003
120	1.89 ± 0.175	0.257 ± 0.010	0.0357 ± 0.0025
140	1.26 ± 0.060	0.249 ± 0.004	0.0330 ± 0.0015

Cuando K crece por encima de 100 se observa que las prestaciones del algoritmo ARRW empeoran en cuanto a ECM. Sin embargo, soluciones bastante mejores que las obtenidas con ARC se pueden obtener con ARRW para valores inferiores de K. Se observa también en la Tabla

4.7 que la sensibilidad disminuye considerablemente cuando K es distinto de 0, con lo que queda probada la efectividad del procedimiento propuesto.

La Figura 4.9 representa la sensibilidad cuadrática media frente a la sensibilidad estadística media, observándose una dependencia aproximadamente lineal, de forma análoga a como se apreciaba en la Figura 4.6. De esta manera queda clara la equivalencia entre sensibilidad estadística y sensibilidad cuadrática y por tanto, entre los procedimientos ARR y ARRW, si bien ARRW aporta ventajas en cuanto a coste computacional respecto de ARR, tal y como se ha mencionado anteriormente.

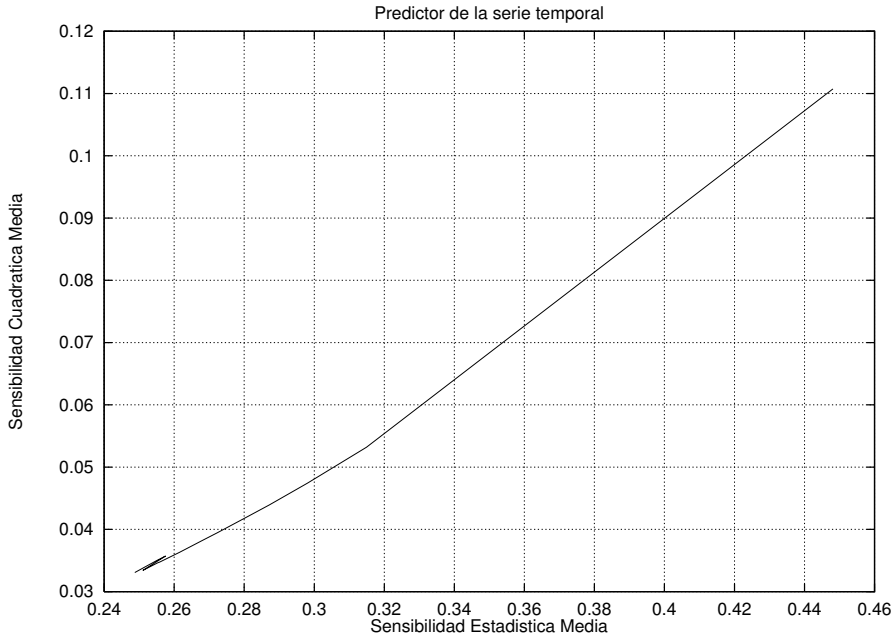

Figura 4.9. Sensibilidad cuadrática frente a sensibilidad estadística.

La Figura 4.10 muestra los valores obtenidos para el error cuadrático medio en función del valor de la constante K en el caso del predictor. Se encuentra representado en forma de diagrama de barras de error, donde dichas barras corresponden a la desviación típica muestral. Se observa que el ECM obtenido para valores de K distintos de 0 es bastante inferior al obtenido con ARC (K=0), y además la desviación típica es pequeña. Esto demuestra que el aprendizaje se ha mejorado al usar ARRW. Sin embargo, debido al desajuste de prioridades para valores de K mayores a 100, el error cuadrático se degrada, aumentando además la desviación típica. Esto indica que el riesgo de quedar atrapado en mínimos locales con baja SCM pero alto ECM es mayor.

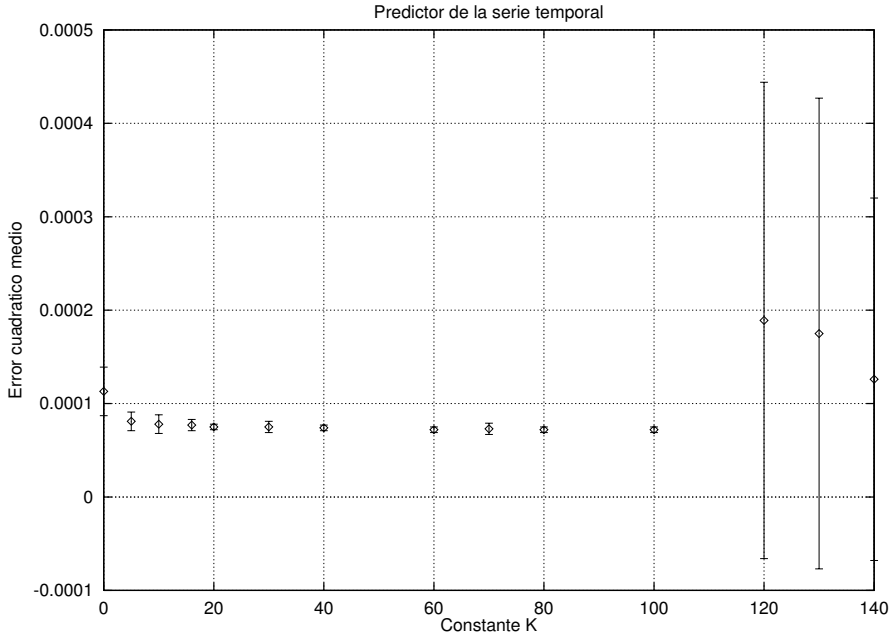

Figura 4.10. Error cuadrático medio frente a K.

La Figura 4.11 representa la sensibilidad cuadrática frente a K. Se aprecia que al aumentar K la SCM disminuye. Al principio, la sensibilidad disminuye rápidamente respecto del caso con K=0, para después iniciar un descenso más lento. El punto óptimo sería el valor más bajo de sensibilidad que presenta un mínimo de error cuadrático, que en este caso se obtiene cuando K es aproximadamente 1.0.

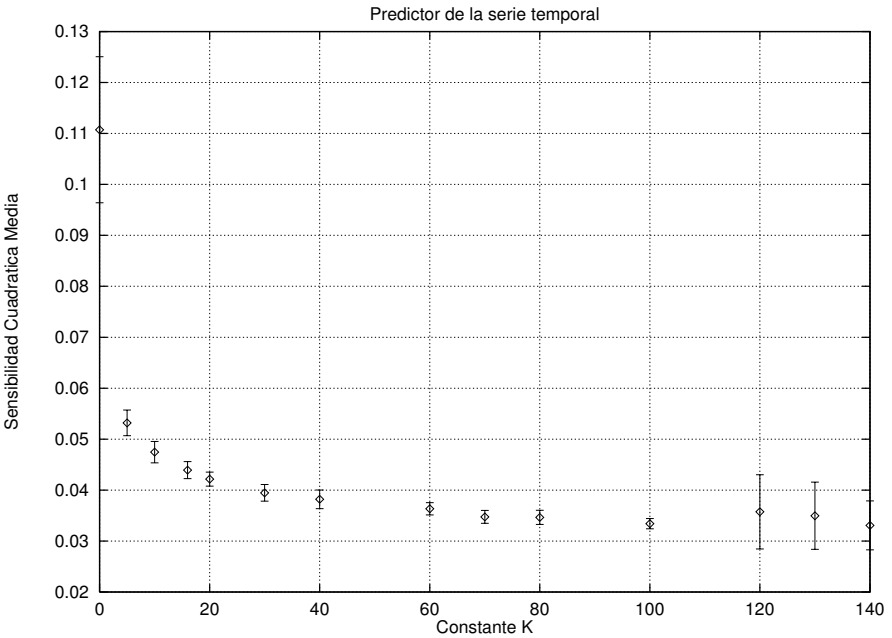

Figura 4.11. Sensibilidad cuadrática frente a K.

4.4.1.3. Resultados para el clasificador del problema de Hart

En el caso de perturbaciones aditivas en el problema de Hart, los resultados obtenidos mediante las expresiones (4.12) y (4.13) se muestran en la Tabla 4.8. Puede apreciarse que, de forma similar a los ejemplos anteriores, es posible encontrar soluciones que sin degradar las prestaciones de aprendizaje, presentan una menor sensibilidad estadística. En la Tabla 4.8 se presenta el error de clasificación en % (EC), en lugar del error cuadrático medio (ECM). El EC es una medida discreta, mientras que el ECM es una medida continua, y es éste último el que se encuentra íntimamente ligado a la SCM. No obstante, cabe esperar que una menor degradación del ECM también se vea acompañada de una menor degradación del EC.

Tabla 4.8. Resultados usando el modelo aditivo para el clasificador.

Constante K	Error de clasificación (%)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	7.22 ± 1.25	0.0948 ± 0.0062	0.0629 ± 0.0065
0.1	6.56 ± 1.27	0.0881 ± 0.0072	0.0569 ± 0.0055
0.5	6.41 ± 1.33	0.0875 ± 0.0064	0.0574 ± 0.0056
1.0	6.43 ± 1.23	0.0923 ± 0.0058	0.0591 ± 0.0063
3.0	6.46 ± 1.10	0.0933 ± 0.0060	0.0558 ± 0.0054
5.0	7.64 ± 1.30	0.1029 ± 0.0051	0.0557 ± 0.0067
8.0	8.04 ± 0.97	0.1011 ± 0.0048	0.0521 ± 0.0058
10	8.32 ± 1.28	0.1027 ± 0.0062	0.0502 ± 0.0062

Puede también apreciarse en la Tabla 4.8 que, en este caso, la disminución de la sensibilidad cuadrática media no se corresponde con la disminución de la sensibilidad estadística media. Este hecho se debe a que el perceptrón dispone de más de una salida, a diferencia de casos anteriores, y puede presentar una salida más tolerante que otra, de forma que concluimos que la media de sensibilidades estadísticas (expresión (2.7)) es una medida menos apropiada para la tolerancia a fallos de la red que la SCM. La minimización de la SCM implica una minimización equitativa de la sensibilidad estadística. Para ilustrar esta explicación, en la Tabla 4.9 se muestra un caso hipotético de un MLP con dos salidas. En la primera columna se muestra la sensibilidad estadística de cada una de las salidas (S_1 y S_2), en la segunda columna se muestra la sensibilidad estadística media (SSM) y, por último la sensibilidad cuadrática media (SCM) se muestra en la tercera columna. Puede apreciarse que para un mismo valor de SSM, el menor valor de SCM se

obtiene para sensibilidades estadísticas de salida más equitativas.

Tabla 4.9. Sensibilidad estadística media y sensibilidad cuadrática media.

S_1	S_2	SSM	SCM
0.1	0.9	0.5	0.41
0.2	0.8	0.5	0.34
0.3	0.7	0.5	0.29
0.4	0.6	0.5	0.26
0.5	0.5	0.5	0.25

Si recordamos los apartados 3.2 y 4.3, en el desarrollo del gradiente se ha supuesto que la derivada de la función de activación es constante, ello se debía a que el término del gradiente debido a dicha derivada estaba en conflicto con el otro término, relacionado con el radicando de la sensibilidad estadística. Sin embargo, como la contribución del término despreciado era la de saturar la señal de salida, para este caso pudiera ser más conveniente usarlo en lugar de la expresión (4.12) puesto que la salida del MLP debe estar saturada en el caso de la clasificación. En este caso, el término correspondiente a las neuronas de salida se puede calcular como:

$$
\begin{aligned}
(\nabla S_c)^2_{ik} &= \frac{1}{2}\,\frac{d}{dw^2_{ik}}\sum_{r=1}^{N_2}(S^2_r)^2 = \frac{1}{2}\,\frac{d(S^2_i)^2}{dw^2_{ik}} \\
&= \frac{1}{2}\,\frac{d}{dw^2_{ik}}(\partial f^2_i)^2\sum_{j=1}^{N_1}((y^1_j)^2 + (w^2_{ij}S^1_j)^2) \\
&= \frac{1}{2}\;2\;\partial f^2_i\;\frac{dz^2_i}{dw^2_{ik}}\;\frac{d^2f^2_i}{(dz^2_i)^2}\sum_{j=1}^{N_1}((y^1_j)^2 + (w^2_{ij}S^1_j)^2) \\
&= y^1_k\;\partial f^2_i\;\partial^2 f^2_i\sum_{j=1}^{N_1}((y^1_j)^2 + (w^2_{ij}S^1_j)^2)
\end{aligned}
\tag{4.18}
$$

En la expresión anterior (4.18) se ha tomado como constante la sumatoria (radicando). Nótese que la expresión completa del gradiente es en realidad la suma de este término (4.18) con el correspondiente a (4.12). El término (4.18) contribuye a saturar la señal de salida, que es lo que interesa en el caso de los clasificadores. De esta forma, se han realizado nuevas pruebas para el problema de Hart considerando (4.18) para las neuronas de salida y (4.13) para las neuronas de la capa oculta (en éstas no tiene porqué interesar que su salida esté saturada). Los resultados se

presentan en la Tabla 4.10.

Tabla 4.10. Resultados usando el modelo aditivo para el clasificador.

Constante K	Error de clasificación (%)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	7.22 ± 1.25	0.0948 ± 0.0062	0.0629 ± 0.0065
0.2	6.11 ± 1.12	0.0881 ± 0.0061	0.0601 ± 0.0053
0.6	6.61 ± 1.38	0.0915 ± 0.0066	0.0584 ± 0.0067
0.8	6.11 ± 1.27	0.0896 ± 0.0054	0.0580 ± 0.0046
1.2	7.33 ± 1.20	0.0929 ± 0.0057	0.0510 ± 0.0063
1.6	7.02 ± 1.27	0.0930 ± 0.0072	0.0535 ± 0.0062
1.8	7.70 ± 1.47	0.0984 ± 0.0075	0.0584 ± 0.0087

Los resultados mostrados en las Tablas 4.8 y 4.10 son similares, por lo que en adelante se usarán los términos (4.12) y (4.13) para el modelo aditivo, y (4.16) y (4.17) para el multiplicativo. Hay que reseñar que cuando se usan los dos términos del gradiente los resultados obtenidos no son buenos, ni se minimiza la sensibilidad ni el error de salida, o bien se obtiene muy poca ventaja respecto del algoritmo clásico, además de ser más costoso computacionalmente que considerando sólo uno de los términos.

Los resultados muestran que usando ARRW se obtienen configuraciones de pesos que presentan una SCM del orden del 81% más pequeña que usando el algoritmo clásico, y con errores de clasificación incluso menores. Se pueden obtener ganancias más elevadas de tolerancia a costa de un error de clasificación algo mayor.

Las Figuras 4.12 y 4.13 muestran el EC y la SCM obtenidos para distintos valores de la constante K. Las barras de error se refieren a la desviación típica muestral. Si bien los resultados presentados en las Tablas 4.8 y 4.10 muestran que es ventajoso usar ARRW, se aprecia en las figuras que dichos resultados no son tan buenos como los obtenidos en los ejemplos anteriores. Ello se debe a que el clasificador presenta por naturaleza una gran tolerancia a pequeñas desviaciones puesto que sus salidas están saturadas. No obstante, se consiguen configuraciones algo más robustas cuando se usa ARRW como algoritmo de aprendizaje.

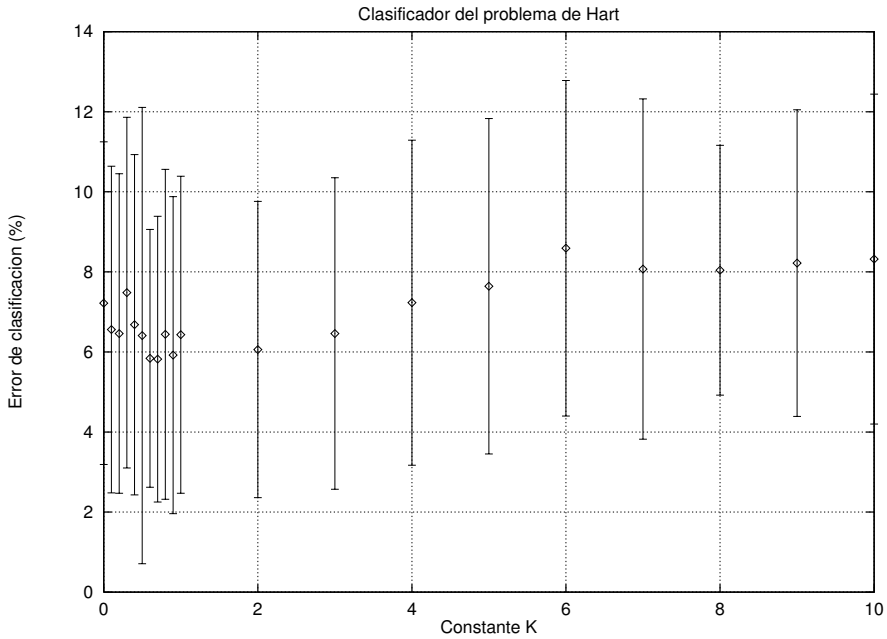

Figura 4.12. Error de clasificación frente a K.

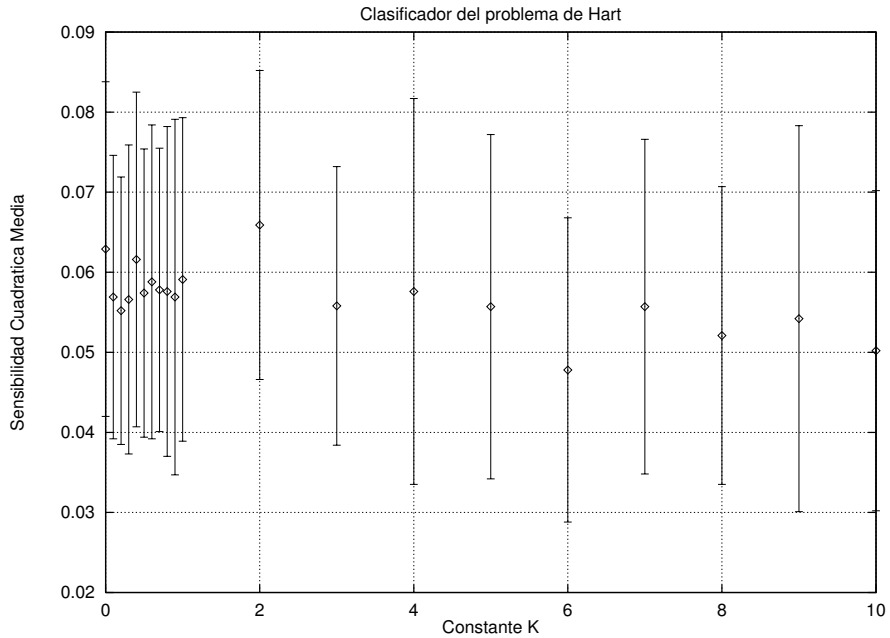

Figura 4.13. Sensibilidad cuadrática media frente a K.

4.4.2. Comparación del algoritmo clásico con ARRW para perturbaciones multiplicativas

En este apartado se muestran los resultados obtenidos cuando se usa la regla ARRW con las expresiones (4.16) y (4.17), correspondientes al modelo de desviación multiplicativo. De forma análoga a como se ha descrito en el apartado anterior, los perceptrones se han entrenado 40 veces con la regla clásica y ARRW para cada valor considerado de K.

4.4.2.1 Resultados para el aproximador de la función seno

La Tabla 4.11 muestra los resultados de ECM, SSM Y SCM obtenidos con el algoritmo ARRW para distintos valores de la constante K del factor de robustez. El resultado obtenido cuando se usa el algoritmo clásico (ARC) se corresponde con la fila en que K se hace 0. Estos resultados se muestran con un intervalo de confianza al 95 %.

Tabla 4.11. Resultados usando el modelo multiplicativo para el aproximador.

Constante K	Error cuadrático medio (x 1e-3)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	0.818 ± 0.096	2.137 ± 0.080	3.122 ± 0.224
0.1	0.680 ± 0.092	2.001 ± 0.071	2.709 ± 0.197
0.2	0.550 ± 0.100	1.948 ± 0.069	2.596 ± 0.193
0.3	0.553 ± 0.076	1.797 ± 0.068	2.161 ± 0.166
0.4	0.620 ± 0.080	1.663 ± 0.043	1.809 ± 0.106
0.5	0.658 ± 0.095	1.509 ± 0.048	1.495 ± 0.099
0.6	0.739 ± 0.091	1.364 ± 0.043	1.198 ± 0.079
0.7	0.809 ± 0.098	1.223 ± 0.030	0.965 ± 0.054
0.8	1.139 ± 0.466	1.121 ± 0.050	0.825 ± 0.061
0.9	0.975 ± 0.120	1.049 ± 0.028	0.704 ± 0.040
1.0	0.910 ± 0.054	1.014 ± 0.021	0.658 ± 0.030

Si nos fijamos en los resultados de la fila en que K=0.7 y los comparamos con la solución del algoritmo clásico (K=0.0) se observa que el error cuadrático medio es similar (0.818 y 0.809,

respectivamente), pero la sensibilidad cuadrática media obtenida con ARRW es unas 3.23 veces menor (0.965) que la obtenida con ARC (3.122). Esto implica que, en principio, el perceptrón entrenado con ARRW es unas 3.23 veces más tolerante a desviaciones multiplicativas que cuando se entrena con el algoritmo clásico.

En general, se observa también en la Tabla 4.11 que para valores de K inferiores a 0.7 se obtiene también un incremento en las prestaciones de aprendizaje, aspecto también observado en los ejemplos del Apartado 4.4.1. Para valores superiores, las prestaciones de aprendizaje se degradan aunque las prestaciones en cuanto a tolerancia siguen aumentando (la SCM disminuye) debido a que la contribución correspondiente en la regla ARRW prima sobre los términos responsables del aprendizaje.

La Figura 4.14 muestra el error cuadrático medio frente a K, mientras que la Figura 4.15 representa la sensibilidad cuadrática media. En ambas figuras, cada punto se representa con barras de error correspondientes a la desviación típica muestral. Puede observarse que mientras que el ECM obtenido con ARRW es del mismo orden que el obtenido con ARC (K=0), la SCM es bastante inferior, estando incluso dentro de distintos rangos para los distintos valores de K considerados.

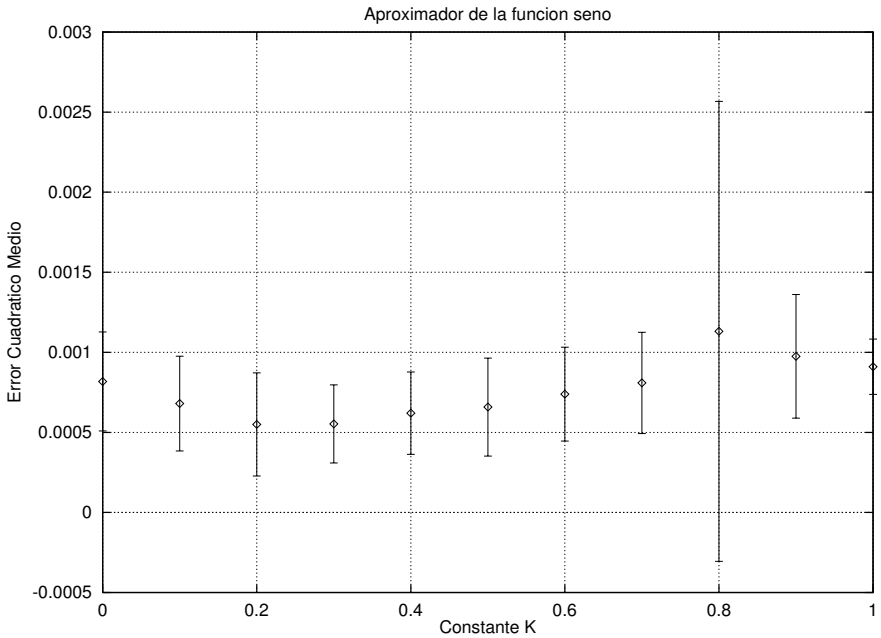

Figura 4.14. Error cuadrático medio frente a K.

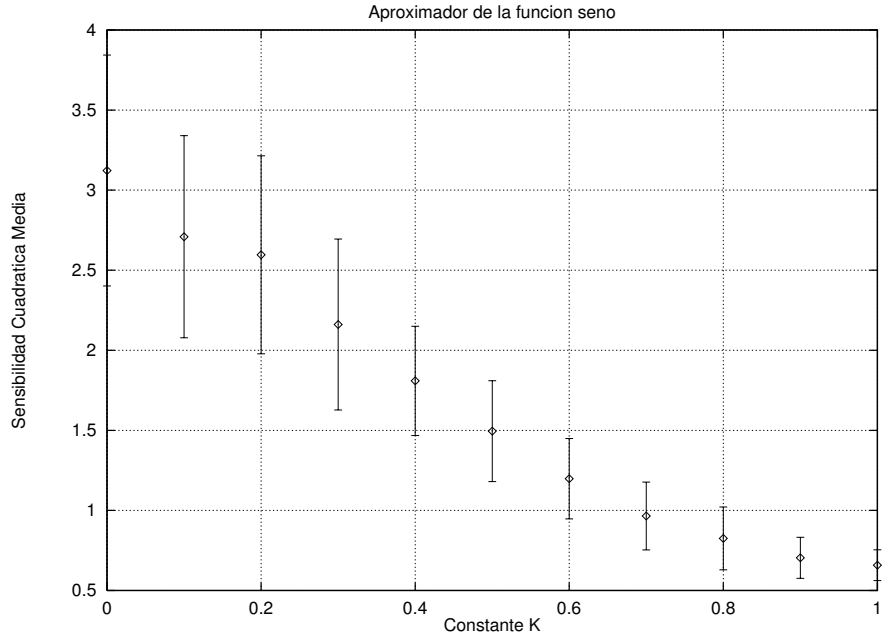

Figura 4.15. Sensibilidad cuadrática frente a K.

4.4.2.2. Resultados para el predictor de la serie de Mackey-Glass

Igual que en los apartados anteriores, la Tabla 4.12 muestra los valores obtenidos en la aplicación de la serie de Mackey-Glss para el error cuadrático medio, sensibilidad estadística media y sensibilidad cuadrática media usando distintos valores de la constante del factor de robustez en el algoritmo ARRW. Para cada valor de K se han realizado 40 tests y los resultados se presentan con un intervalo de confianza al 95%.

Los resultados del algoritmo clásico corresponden a la fila donde K=0. En este caso, para valores de K distintos de 0 se obtienen prestaciones de aprendizaje ligeramente inferiores, si bien son comparables. En cambio, las prestaciones en cuanto a tolerancia son claramente mejores cuando se usa ARRW. Si se compara el resultado obtenido con ARC con el correspondiente a cuando K es igual a 0.9, se puede apreciar que se obtiene una SCM menos de la mitad con ARRW (0.455 con ARC y 0.210 con ARRW).

Las Figuras 4.16 y 4.17 muestran el error cuadrático medio y la sensibilidad cuadrática media obtenidos para distintos valores de K. Para cada punto se representa su desviación típica. Se aprecia que sin degradar las prestaciones de aprendizaje (mismos márgenes de error), sin embargo, se obtienen claras ventajas respecto a tolerancia cuando se usa el algoritmo ARRW.

Tabla 4.12. Resultados usando el modelo multiplicativo para el predictor.

Constante K	Error cuadrático medio (x 1e-4)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	1.06 ± 0.04	0.887 ± 0.062	0.455 ± 0.072
0.1	1.15 ± 0.05	0.801 ± 0.045	0.362 ± 0.041
0.2	1.25 ± 0.11	0.693 ± 0.023	0.270 ± 0.019
0.3	1.15 ± 0.10	0.662 ± 0.012	0.248 ± 0.010
0.4	1.16 ± 0.10	0.643 ± 0.011	0.236 ± 0.008
0.5	1.17 ± 0.11	0.641 ± 0.012	0.235 ± 0.010
0.6	1.15 ± 0.09	0.637 ± 0.012	0.234 ± 0.009
0.7	1.21 ± 0.10	0.619 ± 0.012	0.219 ± 0.009
0.8	1.21 ± 0.09	0.615 ± 0.010	0.218 ± 0.009
0.9	1.22 ± 0.06	0.606 ± 0.010	0.210 ± 0.008
1	1.33 ± 0.13	0.589 ± 0.013	0.199 ± 0.010

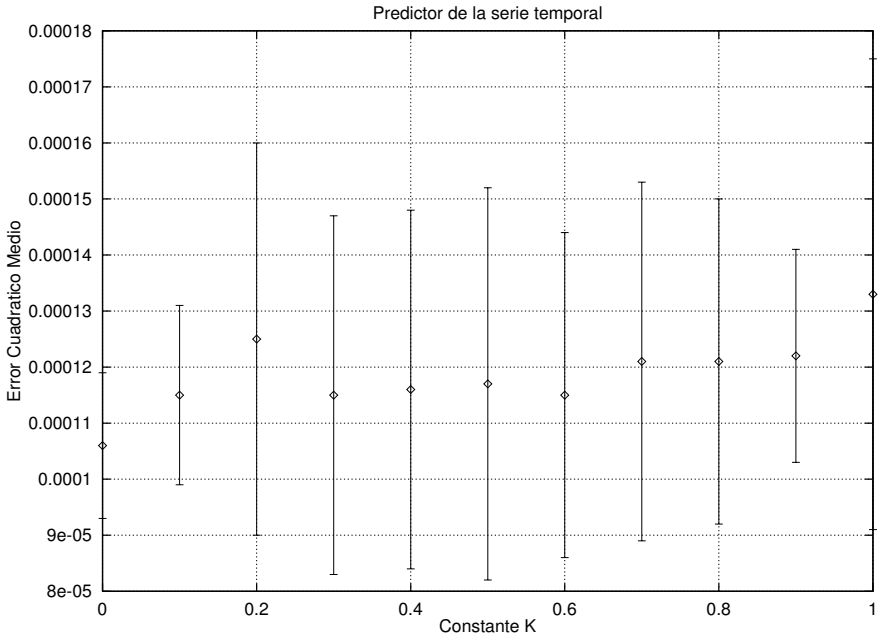

Figura 4.16. Error cuadrático medio frente a K.

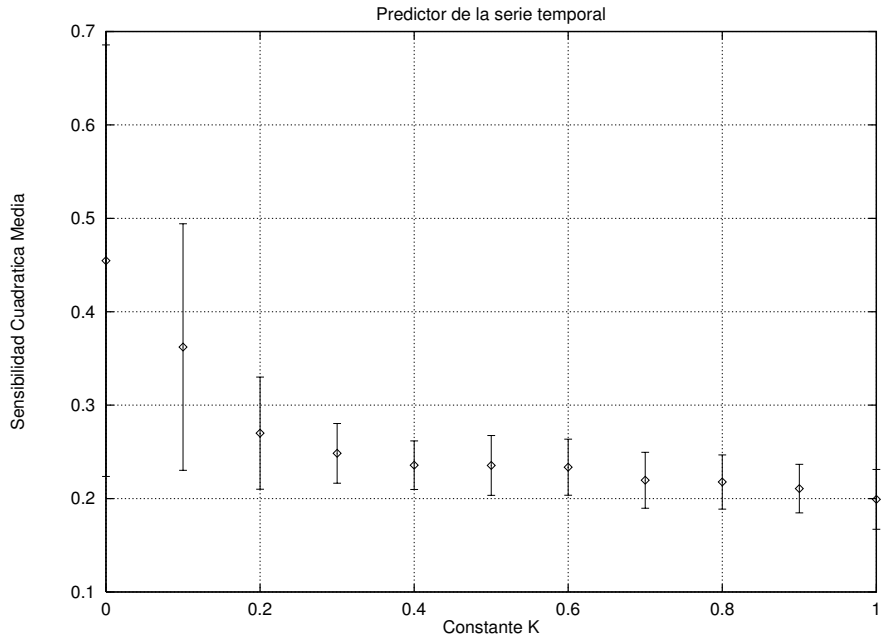

Figura 4.17. Sensibilidad cuadrática frente a K.

4.4.2.3. Resultados para el clasificador del problema de Hart

En el caso de perturbaciones multiplicativas en el problema de Hart, los resultados obtenidos mediante las expresiones (4.16) y (4.17) se muestran en la Tabla 4.13. Puede apreciarse que de forma análoga a los ejemplos considerados anteriormente, es posible encontrar soluciones que sin degradar las prestaciones de aprendizaje, presentan una menor SCM y por tanto mayor tolerancia a faltas paramétricas. En la Tabla 4.13 se presenta el error de clasificación en % (EC), en lugar del ECM.

Si centramos nuestra atención en la columna correspondiente a la SCM se apreciará un decremento considerable de dicha magnitud al incrementar el valor de la constante K. Comparando el resultado obtenido con ARC con el correspondiente a ARRW cuando K=0.175, se puede observar que siendo el EC del mismo orden (4.25% on ARC y 4.37% con ARRW), se obtiene una tolerancia a desviaciones multiplicativas 6.75 veces inferior con ARRW (8.923) respecto de la obtenida con ARC (60.264). En este caso, si bien este resultado es bastante sobresaliente, hay que recordar que los clasificadores de este tipo son ya de por sí bastante robustos, y que una menor SCM implica una menor degradación del ECM pero no existe una relación directa con el EC. De todas formas, parece razonable pensar que una menor degradación del ECM estará relacionada con una menor degradación del EC, por lo que en definitiva lo que obtenemos es una red más robusta que cuando se usa el algoritmo ARC.

Tabla 4.13. Resultados usando el modelo aditivo para el clasificador.

Constante K	Error de clasificación (%)	Sensibilidad estadística media	Sensibilidad cuadrática media
0	4.25 ± 1.11	2.301	60.264 ± 9.222
0.075	4.62 ± 0.61	2.18	31.884 ± 4.015
0.1	3.60 ± 0.48	2.164	27.150 ± 2.047
0.125	3.35 ± 0.63	1.874	17.299 ± 1.525
0.15	4.67 ± 0.68	1.713	12.450 ± 1.442
0.175	4.37 ± 0.67	1.586	8.923 ± 0.866
0.2	5.30 ± 0.80	1.351	5.548 ± 0.619
0.225	6.98 ± 0.89	1.02	2.939 ± 0.495
0.25	8.05 ± 1.02	0.877	2.065 ± 0.390
0.3	11.45 ± 0.43	0.571	0.862 ± 0.059

Las Figuras 4.18 y 4.19 muestran el EC y la SCM obtenidas para distintos valores de la constante K, respectivamente. Las barras de error se refieren a la desviación típica muestral. Como en ejemplos anteriores, se aprecia que los errores de clasificación son del mismo orden pero se obtienen mejores prestaciones en cuanto a tolerancia a fallos. Cuando el término correspondiente a la sensibilidad en el algoritmo ARRW prima sobre los de aprendizaje, se obtienen prestaciones de aprendizaje peores respecto de las de ARC aunque la SCM sigue disminuyendo.

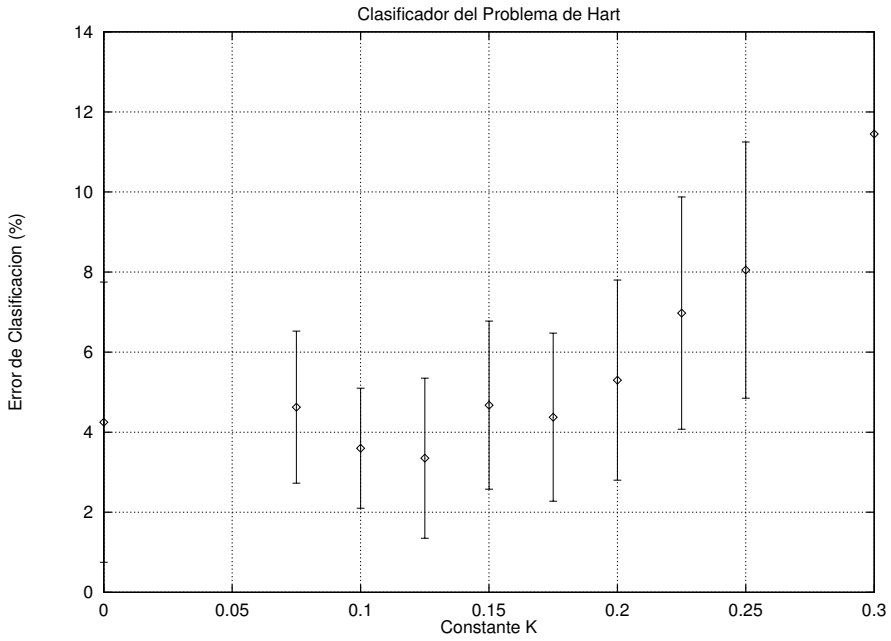

Figura 4.18. Error de clasificación frente a K.

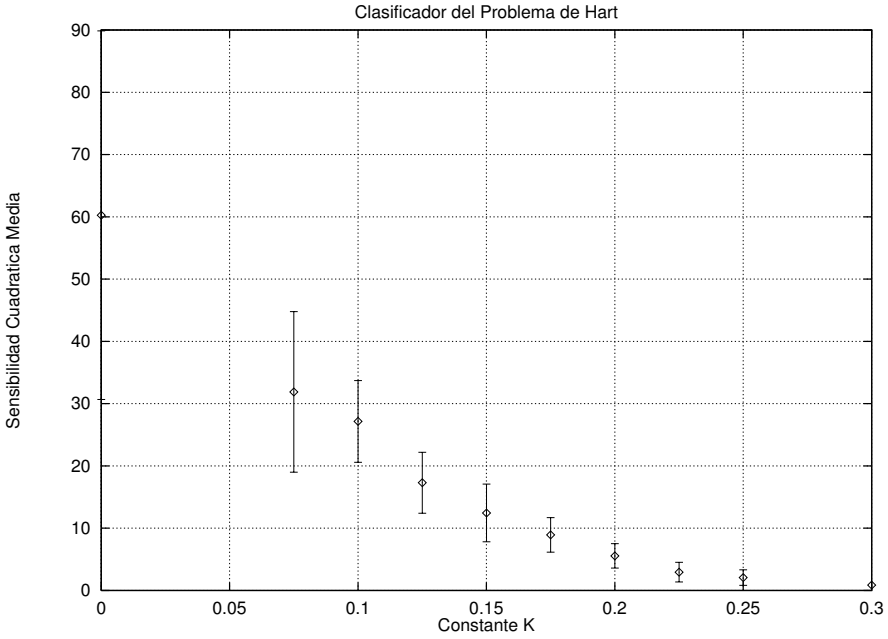

Figura 4.19. Sensibilidad cuadrática media frente a K.

4.5. ESTUDIO DE LA TOLERANCIA A FALLOS EN MLPs ENTRENADOS CON ARRW

De la relación obtenida entre el error cuadrático medio sujeto a perturbaciones y la sensibilidad cuadrática media se desprende que cuanto menor sea el valor de la SCM menor debe ser la degradación del ECM. No obstante en este apartado se presentan los resultados de simulación que demuestran este hecho.

Las configuraciones de pesos obtenidas en los experimentos del Apartado 4.4.2 se han sometido a perturbaciones y se ha calculado el ECM a partir de las salidas proporcionadas por los MLPs. De esta forma, para cada ejemplo considerado, se han sometido las 40 configuraciones de pesos obtenidas mediante ARC y ARRW a perturbaciones multiplicativas, variando la desviación típica de las mismas desde un 1% a un 100%. Para cada valor de la desviación típica se han realizado 20 perturbaciones aleatorias sobre cada una de las configuraciones de pesos. Los resultados de este experimento se muestran en las Figuras 4.20, 4.21 y 4.22, correspondientes al aproximador, el predictor y el clasificador, respectivamente. En el caso del clasificador (Figura 4.22) se muestra el error de clasificación medio.

En el caso del algoritmo ARRW se han escogido las configuraciones de pesos indicadas en las Tablas 4.11 (K=0.7), 4.12 (K=0.9) y 4.13 (K=0.175).

En las tres figuras se aprecia que los perceptrones entrenados con ARRW degradan menos sus prestaciones que aquellos que lo han sido con ARC. Incluso en el caso del clasificador, se puede observar que las prestaciones con respecto al error de clasificación se mantienen mejor, aunque como se ha mencionado, la SCM está directamente relacionada con el ECM y no con el EC.

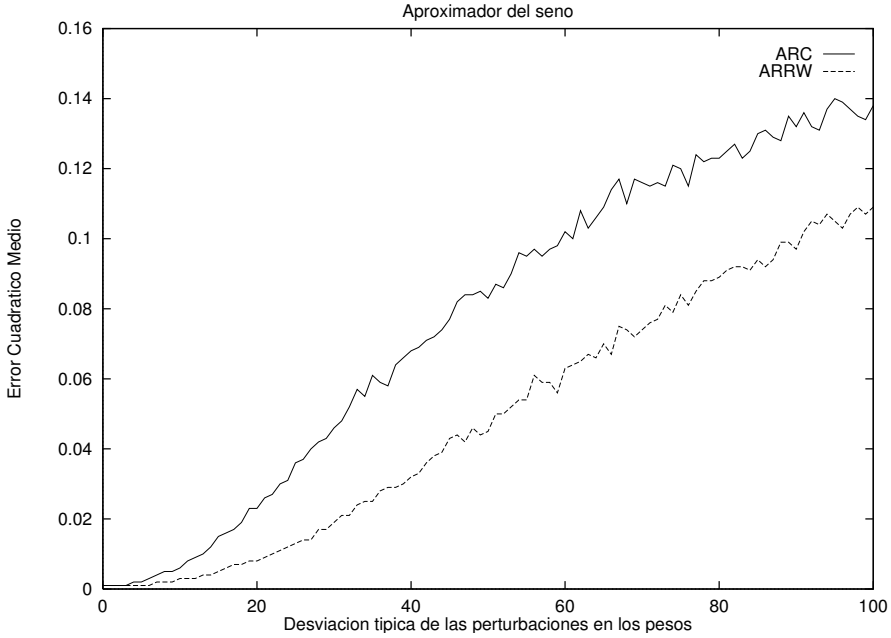

Figura 4.20. Degradación del ECM en el aproximador del seno.

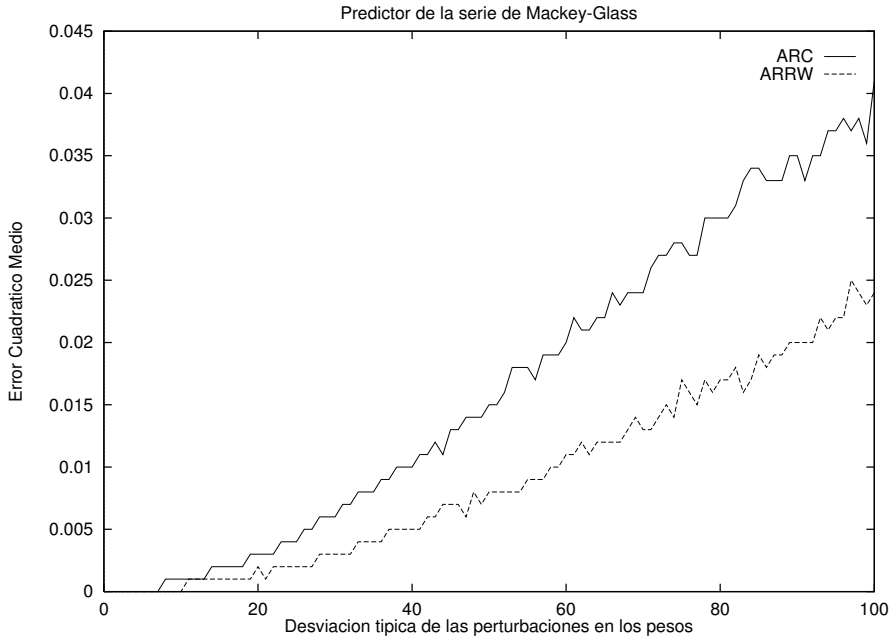

Figura 4.21. Degradación del ECM en el predictor de la serie temporal.

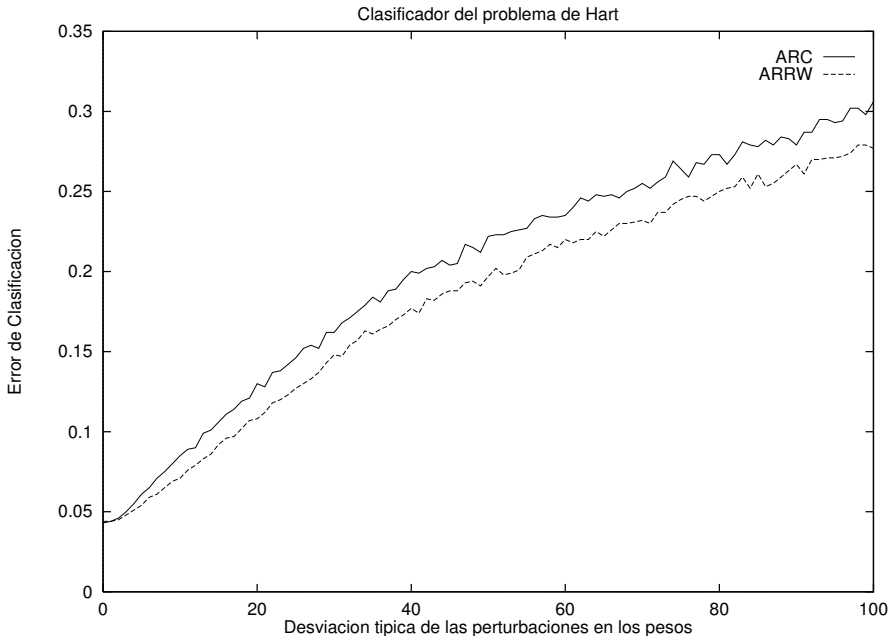

Figura 4.22. Degradación del EC en el clasificador del problema de Hart.

4.6. CONCLUSIONES

En este capítulo se ha mostrado, en primer lugar, la relación explícita del error cuadrático medio (ECM) cuando la red está sujeta a desviaciones en los pesos con la sensibilidad estadística. A partir de esta relación hemos definido una magnitud a la que hemos denominado sensibilidad cuadrática media (SCM). Haciendo uso de la SCM y el ECM obtenidos tras el entrenamiento es posible conocer de antemano la degradación del ECM para un valor determinado de las desviaciones en sus pesos, por lo que se propone la utilización de la SCM como medida fiable de la tolerancia de un MLP.

Comparando la SCM con la sensibilidad estadística media (SSM) propuesta en [CHO92] hemos deducido que la SCM es una medida más apropiada que la SSM, ya que la SCM, además de estar directamente relacionada con la degradación del ECM, mide la uniformidad. Es decir, para valores similares de la SSM, se pueden obtener valores muy distintos de la SCM, indicando el menor de ellos la configuración de pesos para la que las neuronas presentan una sensibilidad estadística similar

También se ha desarrollado un algoritmo similar al propuesto en el Capítulo 3, el cual trata de minimizar la SCM junto con el ECM. Dicho algoritmo, al que hemos denominado ARRW, presenta la ventaja de ser menos costoso computacionalmente, además de ser conceptualmente más adecuado por estar la SCM directamente relacionada con el ECM. Se han presentado diversos ejemplos de aplicación de ARRW mostrando que se obtienen configuraciones de pesos más robustas (menor SCM) que cuando se usa el algoritmo clásico.

Se ha observado en estos ejemplos que para valores de la constante K del factor de robustez que mantienen adecuadamente la prioridad de las magnitudes a optimizar (el ECM y la SCM) se obtiene con ARRW un valor del ECM comparable e incluso inferior (además de una minimización de la SCM) que el obtenido mediante ARC. De aquí se deduce que ARRW no degrada las prestaciones de aprendizaje si se fija adecuadamente el valor de la constante K del factor de robustez.

En el Capítulo 6 se presentan más resultados de la aplicación de ARRW. Allí además se compara con otro algoritmo bastante conocido basado en perturbar el ECM durante el entrenamiento mediante la adición de ruido en los pesos [MUR93]. Mientras tanto, en el Capítulo 5 se va a considerar la perturbación con ruido en las entradas del MLP. Se van a obtener las expresiones de la sensibilidad estadística para tales perturbaciones y se presentará otra regla de aprendizaje que permitirá obtener configuraciones de pesos más inmunes al ruido en las entradas que las obtenidas mediante el entrenamiento con un algoritmo clásico.

Capítulo 5

Estudio de las Perturbaciones en las Entradas y de la Capacidad de Generalización: Regla ARRN

Hasta ahora se ha considerado la sensibilidad estadística a desviaciones en los pesos y, de hecho, se ha supuesto que las entradas al MLP no presentaban errores. En este capítulo, sin embargo, se va a considerar el caso en que las entradas aplicadas al MLP puedan presentar perturbaciones y, siguiendo la filosofía de capítulos anteriores, se van a obtener las expresiones correspondientes a la sensibilidad estadística a ruido en las entradas (Sección 5.2), que serán validadas en la siguiente sección (Sección 5.3). En la Sección 5.4 se muestra la relación existente entre la capacidad de generalización y la SCM a ruido en las entradas, proponiédose el uso de ésta como medida cuantitativa de aquélla. Se presenta un algoritmo similar a ARRW para maximizar la robustez frente a cambios en las entradas al que llamaremos ARRN (Sección 5.5). La Sección 5.6 muestra los resultados obtenidos con ARRN, comparándolos a los que se obtienen con el algoritmo clásico ARC y en la Sección 5.7 se resumen las conclusiones.

5.1. INTRODUCCIÓN

En el desarrollo de los capítulos anteriores se ha considerado siempre que el perceptrón recibía las entradas sin error alguno. En el presente capítulo se va a estudiar la sensibilidad estadística del MLP a desviaciones en las entradas.

Este estudio es relevante desde distintos puntos de vista:

- Para estudiar como se desvirtúan las prestaciones del MLP cuando las entradas se desvían de su valor nominal.

- Para obtener medidas cuantitativas de la tolerancia del MLP cuando sus entradas están afectadas por ruido.

- Para desarrollar algoritmos similares a ARRW (Capítulo 5) que aumenten la inmunidad al ruido en las entradas.

Conocer como se va a comportar el MLP cuando sus entradas se perturban es importante, no sólo porque éste se pueda ver afectado por algún tipo de interferencia en ambientes ruidosos, sino también desde el punto de vista de la implementación de los circuitos electrónicos que proporciona dicha entrada. Por ejemplo, en el caso de entradas analógicas, los sensores proporcionarán la entrada con un cierto margen de tolerancia. Otro aspecto interesante es el referido a implementaciones electrónicas digitales de MLPs, en las que la precisión utilizada para las entradas puede afectar a las prestaciones de los mismos, al igual que afecta la precisión empleada para almacenar los pesos u otras magnitudes.

Se ha propuesto introducir de ruido en las entradas del MLP durante el proceso de aprendizaje para tratar de aumentar la capacidad de generalización de las RNAs a fallos de MLPs [MAO93]. Este procedimiento también se ha mostrado efectivo para incrementar la tolerancia a fallos, ya que equivale a perturbar la función de error durante el entrenamiento [BIS95, EDW96]. De esta forma también puede ser considerado como una técnica de *regularización inherente al algoritmo* [MAO93].

De forma similar a como se ha hecho en capítulos anteriores, en primer lugar se obtendrán las expresiones que permiten calcular la sensibilidad estadística a desviaciones en las entradas. A partir de estas expresiones es posible obtener la sensibilidad cuadrática media, con la cual, se podrá predecir el valor esperado del error cuadrático medio dado un valor determinado de las desviaciones. Un algoritmo similar a ARRW puede usarse para minimizar la sensibilidad cuadrática media (SCM) a desviaciones en las entradas. Este algoritmo, que notaremos como

ARRN, es análogo a ARRW, distinguiéndose únicamente por los términos correspondientes al gradiente y que se usan como estabilizadores. ARRN puede ser considerado una técnica de *regularización explícitamente especificada*. Los resultados obtenidos se compararán con los que se obtienen a partir de ARC.

Por último, cabe destacar que la SCM a desviaciones en los pesos y la SCM a desviaciones en las entradas pueden combinarse mediante la suma los correspondientes gradientes usados en ARRW y en ARRN, de forma que se obtendrían configuraciones de pesos que son más robustas frente al ruido y a perturbaciones en los valores de dichos pesos.

5.2. SENSIBILIDAD ESTADÍSTICA A DESVIACIONES EN LAS ENTRADAS

En este apartado se van a obtener las expresiones de la sensibilidad estadística a desviaciones en las entradas de un MLP. Para su desarrollo se van a considerar los modelos de desviación aditivo y multiplicativo, similares a los considerados para los pesos en los capítulos anteriores. También, se va a asumir que los pesos están libres de errores y permanecen con sus valores nominales.

En primer lugar, en el Apartado 5.2.1 se va a considerar un modelo aditivo de desviación en las entradas, mientras que en el Apartado 5.2.2 se estudiará el modelo multiplicativo.

5.2.1. Modelo aditivo.

En este modelo se asume que las desviaciones en las entradas son procesos de ruido blanco que satisfacen las siguientes condiciones:

a) $E[\Delta y_i^0] = 0 \quad \forall i.$

b) $E[\Delta y_i^0 \, \Delta_j^0] = 0$ si $i \neq j$

c) $E[(\Delta y_i^0)^2] = \sigma^2$

Es decir, en este modelo se supone que cada entrada y_i^0 se desvia de forma que adquiere un nuevo valor $y_i^0 \pm \delta_i$, siendo δ_i una variable aleatoria con desviación típica igual a σ. El objetivo es calcular la sensibilidad estadística cuya expresión viene dada por:

$$S_i^m = \lim_{\sigma \to 0} \frac{\sqrt{var\ (\Delta y_i^m)}}{\sigma} \qquad i=1,...,N_m \ ; \quad m=1,...,M \tag{5.1}$$

donde N_m es el número de neuronas de la capa m, M es el número de capas de la red, y var (Δy_i^m) es la varianza de Δy_i^m, la cual viene dada por

$$var\ (\Delta y_i^m) = E\ [(\Delta y_i^m)^2\ ] - (E\ [\Delta y_i^m\ ])^2 \tag{5.2}$$

Para el desarrollo de la expresión (5.1) se asumirá que las desviaciones son pequeñas de forma que se cumple que

$$\Delta y_i^m = \partial f_i^m \sum_{j=1}^{N_{m-1}} (y_j^{m-1} \Delta w_{ij}^m + w_{ij}^m \Delta y_j^{m-1}) \tag{5.3}$$

siendo ∂f_i^m la derivada de y_i^m respecto de z_i^m (la suma ponderada de las entradas a la neurona). Si se considera que los pesos no sufren desviaciones, se obtiene a partir de (5.3) que

$$\Delta y_i^m = \partial f_i^m \sum_{j=1}^{N_{m-1}} w_{ij}^m \Delta y_j^{m-1} \tag{5.4}$$

Proposición 5.1: $E[\Delta y_i^m] = 0 \quad \forall i\ , \forall m.$

Demostración 5.1: Se va a realizar por inducción sobre m (número de capas).

- Para m=1: A partir de (5.4) se obtiene que:

$$E[\Delta y_i^1] = E[\partial f_i^1 \sum_{j=1}^{N_0} w_{ij}^1 \Delta y_j^0] = \partial f_i^1 \sum_{j=1}^{N_0} w_{ij}^1\ E[\Delta y_j^0] \tag{5.5}$$

donde si se tiene en cuenta que $E\ [\Delta y_j^0] = 0\ \ \forall j$ por a), se prueba que $E[\Delta y_i^1] = 0$.

- Para la capa m. Suponiendo que la $E[\Delta y_i^{m-1}] = 0\ \ \forall i$, se obtiene igualmente que:

$$E[\Delta y_i^{\,m}] \;=\; E[\partial f_i^{\,m} \sum_{j=1}^{N_{m-1}} w_{ij}^{\,m} \Delta y_j^{\,m-1}] \;=\; \partial f_i^{\,m} \sum_{j=1}^{N_{m-1}} w_{ij}^{\,m}\; E[\Delta y_j^{\,m-1}] \;=\; 0 \qquad (5.6)$$

de forma que la Proposición 5.1 queda así probada.

□

Proposición 5.2: Definiendo $C_{jk}^{\,m}$ como $C_{jk}^{\,m} \equiv \dfrac{E[\Delta y_j^{\,m} \Delta y_k^{\,m}]}{\sigma^2}$, el valor de este término $\forall m>0$ se puede expresar recursivamente como:

$$C_{jk}^{\,m} \;=\; \partial f_j^{\,m}\; \partial f_k^{\,m} \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} w_{jr}^{\,m} w_{ks}^{\,m} C_{rs}^{\,m-1} \qquad (5.7)$$

siendo las condiciones iniciales que $C_{jk}^{\,0} = 0$ si $j \neq k$, y $C_{jk}^{\,0} = 1$ en otro caso.

Demostración 5.2:

Para demostrar la Proposición 5.2 se va a obtener la expresión de $E[\Delta y_j^{\,m} \Delta y_k^{\,m}]$ $\forall j,k$. Desarrollando $E[\Delta y_j^{\,m} \Delta y_k^{\,m}]$ se obtiene que:

$$\begin{aligned} E[\Delta y_j^{\,m} \Delta y_k^{\,m}] \;&=\; E[(\partial f_j^{\,m} \sum_{r=1}^{N_{m-1}} w_{jr}^{\,m} \Delta y_r^{\,m-1})\; (\partial f_k^{\,m} \sum_{s=1}^{N_{m-1}} w_{ks}^{\,m} \Delta y_s^{\,m-1})] \\ &=\; E[\partial f_j^{\,m}\; \partial f_k^{\,m} \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} w_{jr}^{\,m} w_{ks}^{\,m}\; \Delta y_r^{\,m-1} \Delta y_s^{\,m-1}] \qquad (5.8) \\ &=\; \partial f_j^{\,m}\; \partial f_k^{\,m} \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} w_{jr}^{\,m} w_{ks}^{\,m}\; E[\Delta y_r^{\,m-1} \Delta y_s^{\,m-1}] \end{aligned}$$

De este modo hemos obtenido una relación recursiva de la que se conoce el caso inicial, dado por b) y c), es decir, $E[\Delta y_j^{\,0} \Delta y_k^{\,0}] = 0$ si $j \neq k$ y $E[(\Delta y_j^{\,0})^2] = \sigma^2$ en otro caso. De estas condiciones se deduce directamente que $C_{jk}^{\,0} = 0$ si $j \neq k$ y que $C_{jk}^{\,0} = 1$ si $j=k$.

Si se hace el cambio de variable indicado en la Proposición 5.2 en (5.8), se obtiene que,

$$E[\Delta y_j^m \ \Delta y_k^m] = \partial f_j^m \ \partial f_k^m \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} \sigma^2 \ w_{jr}^m w_{ks}^m \ C_{rs}^{m-1} \tag{5.9}$$

De esta forma, dividiendo ambos miembros de la expresión (5.9) por σ^2, obtenemos la demostración de la Proposición 5.2.

□

Obsérvese también que los términos C_{jk}^m son simétricos, es decir, $C_{jk}^m = C_{kj}^m \ \forall m$.

Corolario 5.3: La sensibilidad estadística a desviaciones aditivas en la entrada del MLP, para una neurona i perteneciente a la capa m, S_i^m se puede expresar como:

$$S_i^m = \partial f_i^m \sqrt{\sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} w_{ir}^m w_{is}^m \ C_{rs}^{m-1}} \tag{5.10}$$

Demostración 5.3: Por la Proposición 5.1 se sabe que $E[\Delta y_j^m] = 0$ y por lo tanto, la varianza Δy_j^m , var (Δy_j^m), es igual a $E[(\Delta y_j^m)^2]$.

La expresión de $E[(\Delta y_j^m)^2]$ se deduce de la expresión (5.9), particularizada para el caso en que j=k. De aquí, usando la expresión de la sensibilidad estadística (5.1), y sólo con cambiar el subíndice j por el i, se obtiene finalmente (5.10).

□

Corolario 5.4: La sensibilidad estadística a desviaciones aditivas en las entradas, para una neurona i de la primera capa oculta, S_i^1, se puede obtener como:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{r=1}^{N_0} (w_{ir}^1)^2} \tag{5.11}$$

Demostración 5.4: La demostración de (5.11) es directa. Basta aplicar la expresión (5.10)

a las neuronas de la primera capa oculta (m=1), y tener en cuenta las condiciones iniciales, es decir, $C_{rs}^{0} = \delta_{rs}$, siendo δ_{rs} la delta de Kronecker.

□

Corolario 5.5: Para las neuronas de la primera capa oculta (m=1) cuando j≠k, el valor de los términos cruzados C_{jk}^{1} se puede expresar como:

$$C_{jk}^{1} = \partial f_j^1 \ \partial f_k^1 \sum_{r=1}^{N_0} w_{jr}^1 w_{kr}^1 \tag{5.12}$$

Demostración 5.5: Teniendo en cuenta la expresión (5.7) para las neuronas de la primera capa oculta (m=1), y que $C_{rs}^{0} = \delta_{rs}$, siendo δ_{rs} la delta de Kronecker, se obtiene la demostración del Corolario 5.5.

□

Puede apreciarse que, a diferencia del Corolario 2.6 (Capítulo 2) referido a la sensibilidad frente a desviaciones aditivas en los pesos, los términos cruzados C_{jk}^{1} con j≠k no son nulos para las neuronas de la primera capa (m=1) por lo que la expresión de la sensibilidad estadística a ruido en las entradas para las neuronas de la capa de salida, en un MLP con una única capa oculta, es algo más complicada dada la influencia de tales términos. En el siguiente corolario se obtendrá dicha expresión.

Corolario 5.6: La sensibilidad estadística a desviaciones aditivas en las entradas, para una neurona i de la segunda capa, S_i^2, se puede obtener como:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} ((w_{ij}^2 \ S_j^1)^2 + 2 w_{ij}^2 \ \partial f_j^1 \sum_{k>j}^{N_1} w_{ik}^2 \ \partial f_k^1 \sum_{r=1}^{N_0} w_{jr}^1 w_{kr}^1)} \tag{5.13}$$

Demostración 5.6: A partir de (5.10), particularizada para m=2, se obtiene que

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1}\sum_{k=1}^{N_1} w_{ij}^2 w_{ik}^2 C_{jk}^1}$$
$$= \partial f_i^2 \sqrt{\sum_{j=1}^{N_1}((w_{ij}^2)^2 C_{jj}^1 + w_{ij}^2 \sum_{k=1, j\neq k}^{N_1} w_{ik}^2 C_{jk}^1)} \quad (5.14)$$

En la expresión (5.14), si se tiene en cuenta que de la definición de $C_{jj}^{\ 1}$ se deduce que $C_{jj}^{\ 1} = (S_j^{\ 1})^2$ y que los términos $C_{jk}^{\ m} = C_{kj}^{\ m}$ $\forall$m, tal como se observa a partir de (5.7). Haciendo los cambios oportunos se obtiene la siguiente expresión:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1}((w_{ij}^2 S_j^1)^2 + 2w_{ij}^2 \sum_{k>j}^{N_1} w_{ik}^2 C_{jk}^1)} \quad (5.15)$$

En esta última expresión (5.15), basta tener en cuenta el Corolario 5.5 para obtener (5.13) quedando demostrado de esta manera el Corolario 5.6.

□

Mientras que la expresión (5.10) permite obtener la sensibilidad estadística a ruido aditivo en las entradas de cualquier neurona en un MLP independientemente del número de capas que lo constituyan, las expresiones (5.11) y (5.15) nos permiten obtener la sensibilidad estadística para las neuronas de MLPs con una sola capa oculta, que es en los que vamos a centrar nuestro estudio.

En el siguiente apartado se obtendrán expresiones similares considerando un modelo de desviación multiplicativo.

5.2.2. Modelo multiplicativo

El modelo de ruido multiplicativo en las entradas asume que las desviaciones en dichas entradas son procesos de ruido blanco que satisfacen las siguientes condiciones:

a) $E[\Delta y_i^0] = 0 \quad \forall i$.

b) $E[\Delta y_i^0 \, \Delta y_j^0] = 0$ si $i \neq j$

c) $E[(\Delta y_i^0)^2] = (\sigma \, y_i^0)^2$

Es decir, en este modelo se supone que cada entrada y_i^0 se desvia de forma que adquiere un nuevo valor igual a $y_i^0 (1 \pm \delta_i)$, siendo δ_i una variable aleatoria con desviación típica igual a σ.

Proposición 5.7: $E[\Delta y_i^m] = 0 \quad \forall i \; \forall m$.

Demostración 5.7: La Proposición 5.7 se prueba de forma idéntica a como se hizo en la Proposición 5.1 por lo que no se repetirá aquí.

□

Proposición 5.8: Definiendo C_{jk}^m como $C_{jk}^m \equiv \dfrac{E[\Delta y_j^m \Delta y_k^m]}{\sigma^2}$, el valor de este término $\forall m>0$ se puede expresar recursivamente como:

$$C_{jk}^m = \partial f_j^m \; \partial f_k^m \sum_{r=1}^{N_{m-1}} \sum_{s=1}^{N_{m-1}} w_{jr}^m w_{ks}^m C_{rs}^{m-1} \tag{5.16}$$

siendo las condiciones iniciales que $C_{jk}^0 = 0$ si $j \neq k$, y $C_{jk}^0 = (y_j^0)^2$ en otro caso.

Demostración 5.8:

Para demostrar la Proposición 5.8 basta considerar la expresión de $E[\Delta y_j^m \Delta y_k^m] \; \forall j,k$ dada por (5.8) y dividir por σ^2.

De este modo hemos obtenido una relación recursiva de la que se conoce el caso inicial, dado por b) y c), es decir, $E[\Delta y_j^0 \Delta y_k^0] = 0$ si $j \neq k$, y $E[(\Delta y_j^0)^2] = (\sigma y_j^0)^2$ en otro caso. De estas condiciones se deduce directamente que $C_{jk}^0 = 0$ si $j \neq k$, y que $C_{jk}^0 = (y_j^0)^2$ si $j=k$.

De forma similar a como se hizo en la Demostración 5.2, basta realizar el cambio de variable $E[\Delta y_j^m \Delta y_k^m] = \sigma^2 C_{jk}^m$ en la expresión (5.8) y dividir ambos miembros por σ^2 para obtener definitivamente la demostración requerida.

□

Se puede observar, que los términos C_{jk}^m son simétricos, es decir, $C_{jk}^m = C_{kj}^m \ \forall m$, igual que sucedía en el caso de desviación aditivo. De hecho, las expresiones (5.7) y (5.16) son formalmente idénticas, variando únicamente en las condiciones iniciales derivadas del modelo de desviación considerado. Así, el conjunto de expresiones derivadas en los corolarios demostrados en el caso de ruido aditivo son muy similares a los que se obtendrán a continuación.

Corolario 5.9: La sensibilidad estadística a desviaciones multiplicativas en la entrada del MLP, para una neurona i perteneciente a la capa m, S_i^m, viene dada por la expresión (5.10).

Demostración 5.9: La demostración es formalmente idéntica a la realizada en el caso del Corolario 5.3, ya que la expresión de la sensibilidad estadística es similar en el caso de desviaciones aditivas y multiplicativas. La diferencia fundamental entre ambos casos viene determinada por las distintas condiciones iniciales aconsiderar en la recursión. Este hecho será puesto de manifiesto en los siguientes corolarios.

□

Corolario 5.10: La sensibilidad estadística a desviaciones multiplicativas en las entradas, para una neurona i de la primera capa oculta, S_i^1, se puede obtener como:

$$S_i^1 = \partial f_i^1 \sqrt{\sum_{r=1}^{N_0} (w_{ir}^1 \ y_r^0)^2} \qquad (5.17)$$

Demostración 5.10: Para obtener (5.11) basta aplicar la expresión (5.10) a las neuronas de la primera capa oculta (m=1), y tener en cuenta las condiciones iniciales, es decir, $C_{rs}^0 = 0$ si $r \neq s$, y $C_{rr}^0 = (y_r^0)^2$ tal como se deduce de la condición b) del modelo de desviación multiplicativo.

□

Corolario 5.11: Para las neuronas de la primera capa oculta (m=1) cuando j≠k, el valor de los términos cruzados C_{jk}^{1}, n el caso de desviaciones multiplicativas en las entradas, se puede expresar como:

$$C_{jk}^{1} = \partial f_j^1 \ \partial f_k^1 \sum_{r=1}^{N_0} w_{jr}^1 w_{kr}^1 \ (y_r^0)^2 \tag{5.18}$$

Demostración 5.11: Teniendo en cuenta la expresión (5.16) para las neuronas de la primera capa oculta (m=1), que $C_{rs}^{0} = 0$ si r≠s y que $C_{rr}^{0} = (y_r^0)^2$, se tiene la demostración del Corolario 5.11.

□

Corolario 5.12: La sensibilidad estadística a desviaciones multiplicativas en las entradas, para una neurona i de la segunda capa, S_i^2, se puede obtener como:

$$S_i^2 = \partial f_i^2 \sqrt{\sum_{j=1}^{N_1} ((w_{ij}^2 \ S_j^1)^2 + 2w_{ij}^2 \ \partial f_j^1 \sum_{k>j}^{N_1} w_{ik}^2 \ \partial f_k^1 \sum_{r=1}^{N_0} w_{jr}^1 w_{kr}^1 \ (y_r^0)^2)} \tag{5.19}$$

Demostración 5.12: Haciendo las mismas consideraciones que en la Demostración 5.6 y teniendo en cuenta la expresión (5.18) para los términos cruzados C_{jk}^{1} obtenida por el Corolario 5.11 se deduce la expresión (5.19).

□

Así pues, se han determinado las expresiones de la sensibilidad estadística a ruido en las entradas para los dos modelos de desviación considerados, el aditivo y el multiplicativo. Las expresiones respectivas son muy similares formalmente, diferenciándose en las condiciones iniciales. De forma similar a lo indicado para la expresión de la sensibilidad estadística a desviaciones en los pesos (2.17) se observa que en el caso de ruido en las entradas, el valor de la sensibilidad dado por la expresión (5.10) es menor bien cuando el valor de la derivada ∂f_i^m es pequeño (es cero si la señal está saturada), o bien cuando las magnitudes de los pesos son pequeñas. No obstante, como los términos cruzados no son nulos, aún en MLPs de una sola capa oculta, puede apreciarse a partir de (5.13) y (5.19) que no es suficiente que las magnitudes de los pesos sean pequeñas, sino que además la interrelación entre dichos términos sea lo menor posible. Por eso es preciso que la distribución del aprendizaje se haya realizado de manera que se minimice la sensibilidad

estadística.

Otro aspecto no comentado aún se refiere al hecho de que las neuronas presenten entrada de sesgo. En la obtención de las expresiones presentadas hasta ahora no se ha considerado la entrada de sesgo, por lo que en principio sólo son válidas para neuronas que carecen de la misma. En el caso de usar dichas entradas, existen varias posibilidades. En la primera de ellas, si se considera que la entrada de sesgo está libre de errores, entonces la expresión obtenida (5.10) sigue siendo válida puesto la entrada no influye en la sensibilidad estadística al no sufrir perturbaciones y los correspondientes términos C_{0k}^{m} son nulos. Si las entradas de sesgo están sujetas a perturbaciones es necesario tener en cuenta sus efectos. Para ello, dichas entradas deben tratarse como entradas primarias (recordemos que son entradas ficticias a las neuronas, es decir, no interconectan neuronas) y de esta forma deben añadirse en las expresiones correspondientes como un término más, incluyendo en la sumatoria de (5.10) los C_{0k}^{m} (indicando el subíndice 0 la entrada de sesgo) calculados a partir del modelo de desviación considerado. Si hay entradas de sesgo, las expresiones también difieren según se tenga que dichas entradas son compartidas o independientes para cada neurona. En los ejemplos utilizados para validar las expresiones de la sensibilidad a ruido en las entradas, se va a considerar que las neuronas poseen entrada de sesgo pero que dichas entradas no sufren perturbaciones.

En la siguiente sección se van a mostrar resultados que validan las expresiones obtenidas anteriormente, mientras que en la Sección 5.4 se derivarán las expresiones del gradiente de la sensibilidad para, de manera similar a como se hizo en el Capítulo 4, disponer de una nueva regla de aprendizaje que minimice la sensibilidad al ruido en las entradas, manteniendo las prestaciones del algoritmo clásico respecto al aprendizaje. A esta nueva regla la llamaremos ARRN.

5.3. VALIDACIÓN DE LAS EXPRESIONES DE LA SENSIBILIDAD ESTADÍSTICA A RUIDO EN LAS ENTRADAS

Hemos obtenido las expresiones de la sensibilidad estadística a ruido en las entradas de un MLP considerando el modelo aditivo (5.13) y el multiplicativo (5.19). Ahora se validarán estas expresiones mediante resultados experimentales obtenidos a partir del cómputo directo realizado tras perturbar los patrones de entrada conforme a los modelos antedichos.

Para realizar dicha validación se usa la expresión (4.6). Dicha expresión relaciona el valor esperado del error cuadrático medio (ECM) de un MLP sujeto a perturbaciones, con la sensibilidad cuadrática media (SCM). Esta expresión se obtuvo para el caso de perturbaciones en los pesos, pero puede apreciarse en el desarrollo realizado en la Sección 4.1 que la expresión (4.6) es también válida si se considera que las perturbaciones se producen en las entradas. En este último caso, sólo hay que tener en cuenta que la SCM se debe calcular a partir de las expresiones (5.13) o (5.19) según se considere el modelo aditivo o multiplicativo, respectivamente. Para no generar confusión, a partir de ahora se distinguirá entre la SCM frente a perturbaciones en los pesos (SCMW) y la SCM frente a ruido en las entradas (SCMN).

La expresión (4.6) permite predecir el comportamiento de un MLP ya entrenado, con un determinado ECM nominal (ε_0), cuando sus entradas se perturban en una cierta magnitud con desviación típica σ. Para ello basta calcular la sensibilidad cuadrática media utilizando la expresión (4.9). Si se tienen en cuenta las expresiones para la sensibilidad estadística ruido en las entradas obtenidas en los apartados 5.2.1 y 5.2.2, entonces el valor predicho mediante la expresión (4.6) debe ajustarse al obtenido mediante el cálculo directo del valor medio del ECM ($E[\varepsilon']$) cuando se perturban los patrones de entrada.

De esta forma, para validar las expresiones obtenidas vamos a usar MLPs previamente entrenados para los que se calcularán el ECM nominal (ε_0) y la SCMN. Para una determinada perturbación aleatoria de los valores de los patrones de entrada con desviación típica σ, y acorde con el modelo de desviación aditivo o multiplicativo, se calculará el valor predicho mediante la expresión (4.6). Este valor se comparará con el obtenido a partir de la evaluación de $E[\varepsilon']$ para un determinado número de simulaciones. Cada simulación consistirá en la perturbación aleatoria de todos los patrones de entrada dentro del rango establecido.

Obsérvese también que la SCMW puede sumarse a la SCMN para considerar el efecto conjunto de perturbaciones en los pesos y en las entradas siendo válida la expresión (4.6) para predecir los efectos de tales desviaciones en las prestaciones del MLP. Incluso pueden considerarse distintos rangos para unas desviaciones y otras.

Se han utilizado MLPs que aproximan la función seno y que predicen la serie temporal, cuyas estructuras han sido descritas en capítulos anteriores. Las neuronas poseen una entrada de sesgo, si bien dicha entrada se ha supuesto libre de error. Para cada valor de σ considerado, se han aplicado 100 perturbaciones aleatorias de cada uno de los patrones de entrada con objeto de calcular experimentalmente el valor esperado del ECM.

En primer lugar se van a mostrar resultados usando el modelo de desviación aditivo en 5.3.1 y, a continuación, los obtenidos usando el modelo multiplicativo en 5.3.2.

5.3.1 Uso del modelo de desviación aditivo

Para validar la expresión (5.13) de la SCMN frente a ruido aditivo, se han perturbado las entradas primarias y_i^0 con objeto de comparar el valor obtenido para el valor esperado del ECM mediante simulación con el obtenido a partir de la expresión (4.6). En cada test, todas las entradas del MLP se han perturbado de forma que toman un valor $y_i^0 = p_i \pm \delta_i$, siendo p_i la componente i del patrón de entrada p, y δ_i una variable aleatoria que sigue una distribución normal de media 0 y varianza σ^2. El valor esperado del error cuadrático medio de la red para cada valor de σ se calcula como la media de este error sobre 100 tests con el conjunto de patrones de entrada considerados.

La Tabla 5.1 muestra los valores obtenidos para el ECM nominal (ε_0) y la SCMN. Ambas magnitudes se han calculado sobre el conjunto de patrones de test una vez finalizado el entrenamiento (los patrones de test son distintos de los usados durante el entrenamiento).

Tabla 5.1. Error cuadrático medio y sensibilidad cuadrática media nominales.

	Aproximador	Predictor
ε_0	0.0011	0.0001
SCMN	2.125	0.8478

En la Tabla 5.2 se presentan los valores obtenidos mediante el cálculo directo de $E[\varepsilon']$ con un intervalo de confianza al 95% , y los valores predichos mediante la expresión (4.6) para distintos valores de σ. Se aprecia que el valor predicho se aproxima fielmente al valor experimental obtenido mediante el cálculo directo, tanto más cuanto menor sea el valor de σ como era de esperar por la aproximación realizada al obtener las expresiones.

Tabla 5.2. Comparación entre E[ε'] predicho y experimental.

	Aproximador		Predictor	
σ	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)
0.01	1.357	1.410 ± 0.053	0.188	0.192 ± 0.004
0.02	1.995	1.952 ± 0.113	0.442	0.449 ± 0.010
0.03	3.058	2.938 ± 0.206	0.866	0.891 ± 0.023
0.04	4.545	4.470 ± 0.413	1.46	1.462 ± 0.039
0.05	6.458	5.881 ± 0.521	2.223	2.194 ± 0.064
0.06	8.795	8.558 ± 0.784	3.155	3.117 ± 0.088
0.07	11.56	10.49 ± 0.829	4.257	4.202 ± 0.113
0.08	14.75	12.28 ± 1.03	5.529	5.378 ± 0.175

En las figuras 5.1 y 5.2 se representan los valores predichos y calculados mediante simulación para distintos valores de σ. Los valores calculados experimentalmente se muestran con el intervalo de confianza al 95% obtenido sobre 100 muestras. La Figura 5.1 se refiere al aproximador de la función seno mientras que la Figura 5.2 se refiere al predictor de la serie temporal. Puede apreciarse que la curva obtenida mediante la expresión (4.6) se acerca bastante a la experimental en ambas figuras, siendo lógicamente mejor el acoplamiento cuanto menor sea el ruido.

En ambos casos se pone de manifiesto la validez de la expresión (4.6) para el caso de ruido aditivo, al menos hasta un punto crítico en el que las prestaciones de la red se han degradado considerablemente respecto de las obtenidas tras el entrenamiento. En cualquier caso, incluso cuando la aproximación se aleja algo del valor real, el valor predicho constituye una cota superior por lo que permite saber que el ECM no será superior a tal valor y, por tanto, la SCMN constituye una magnitud apropiada para evaluar la inmunidad al ruido de los MLPs.

Las desviaciones aditivas en las entradas, junto con las desviaciones aditivas en los pesos permiten estudiar el comportamiento del ECM debido a los efectos de cuantización, es decir, cuando se cambia la resolución utilizada para almacenar los valores de estas magnitudes. De esta forma, la expresión (4.6), donde la SCM se calcule considerando perturbaciones aditivas tanto en las entradas como en los pesos, puede usarse para estudiar la degradación del ECM al cambiar el número de bits para almacenar los patrones de entrada o los valores de los pesos. El modelo

aditivo de desviación en los pesos también puede usarse estudiar los efectos del 'offset' debido a los transistores en una implementación analógica.

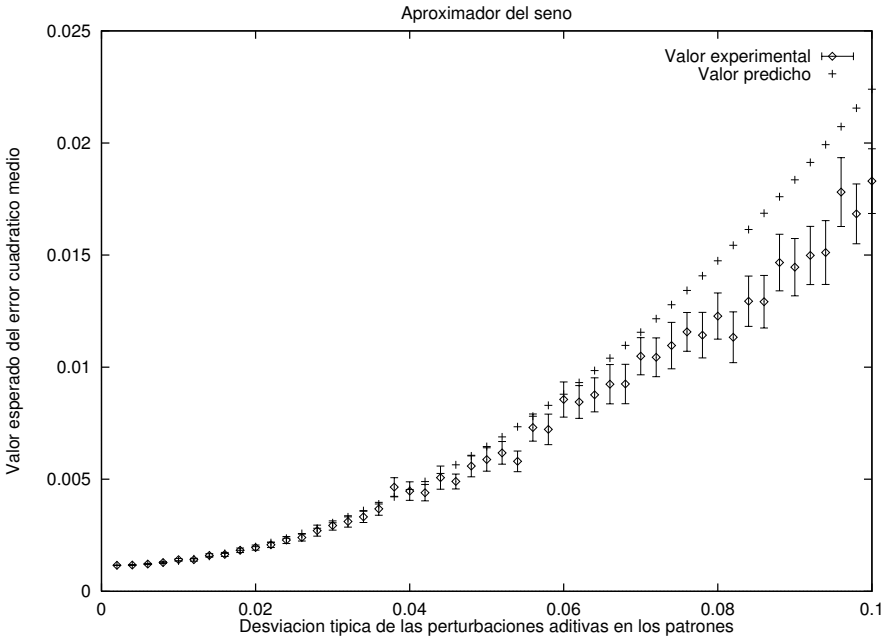

Figura 5.1. Error cuadrático medio experimental y predicho frente a σ.

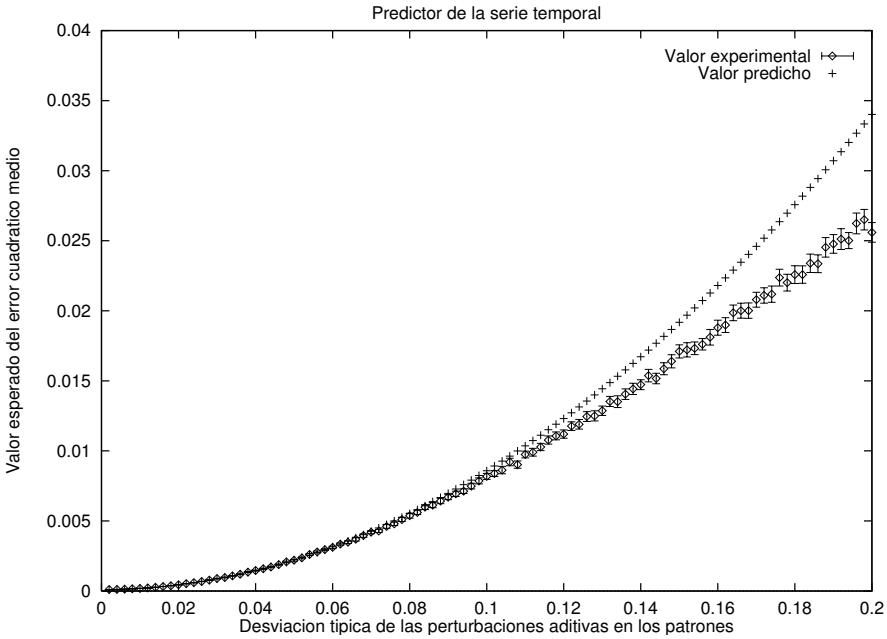

Figura 5.2. Error cuadrático medio experimental y predicho frente a σ.

5.3.2. Uso del modelo de desviación multiplicativo

En el caso de desviaciones multiplicativas en los entradas, en cada test todas las entradas y_i^0 se han modificado para tomar un valor $y_i^0 = p_i (1+\delta_i)$ siendo p_i la componente i del patrón de entrada p, y δ_i una variable aleatoria que sigue una distribución normal de media 0 y desviación típica σ. El valor esperado del error cuadrático medio de la red para cada valor de σ se calcula como la media de este error para los patrones de entrada considerados sobre 100 tests, donde cada test implica una perturbación aleatoria de los pesos según el modelo multiplicativo. Usando este modelo de desviación, el valor de σ representa el valor de la tolerancia en tanto por 1, por lo que cuando las entradas se obtienen a partir de sensores analógicos con un cierto de margen de tolerancia, la expresión (4.6) permite predecir cuales serán las prestaciones de dicha implementación.

La Tabla 5.3 muestra los valores obtenidos para el error cuadrático medio nominal ε_0 y la sensibilidad cuadrática media a ruido en las entradas (SCMN) tras el entrenamiento. Se puede observar que el valor de la SCMN es de 0.7731 en el caso del aproximador del seno lo que va a implicar una rápida degradación de las prestaciones al crecer el valor de las perturbaciones en las entradas. En cambio, para el predictor la SCMN toma un valor de 0.3202 por lo que sus prestaciones se degradarán en menor cuantía para ruido de una misma magnitud.

Tabla 5.3. Error cuadrático medio y sensibilidad cuadrática media nominales.

	Aproximador	Predictor
ε_0	0.0004	0.0001
SCMN	0.7731	0.3202

La Tabla 5.4 muestra algunos ejemplos de valores predichos y simulados para distintos valores de σ. Como se ha mencionado antes, σ se puede identificar con la tolerancia analógica y por ello, en la Tabla 5.4 se muestra el valor de la misma expresada en %. El valor esperado del ECM de la red perturbada, E[ε'], se presenta con un intervalo de confianza al 95%. Este valor se ha calculado sobre 100 muestras. Puede observarse que los valores predichos y simulados son bastante semejantes, estando dentro del mismo margen en la mayoría de los casos. La diferencia entre el valor predicho y el simulado aumenta al crecer la magnitud de las desviaciones, aunque al ser superior siempre el valor predicho, permite disponer de una cota superior de manera que se puede conocer el valor máximo de degradación para el MLP en cuestión.

Tabla 5.4. Comparación entre E[ε'] predicho y calculado.

	Aproximador		Predictor	
Tolerancia (%)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)	E[ε'] predicho (x 1e-3)	E[ε'] experimental (x 1e-3)
0.6	0.494	0.495 ± 0.005	0.114	0.114 ± 0.001
1.2	0.578	0.576 ± 0.014	0.149	0.149 ± 0.002
1.8	0.717	0.696 ± 0.028	0.206	0.208 ± 0.004
2.4	0.912	0.863 ± 0.045	0.287	0.287 ± 0.006
3	1.162	1.097 ± 0.065	0.391	0.397 ± 0.012
3.6	1.468	1.346 ± 0.075	0.518	0.535 ± 0.016
4.2	1.83	1.574 ± 0.116	0.667	0.690 ± 0.020
4.8	2.247	1.957 ± 0.159	0.84	0.882 ± 0.028

En las Figuras 5.3 y 5.4 se representan los valores predichos y calculados mediante simulación para distintos valores de σ para el aproximador de la función seno y el predictor de la serie temporal, respectivamente. Cada punto experimental se muestra con su correspondiente intervalo de confianza al 95%.

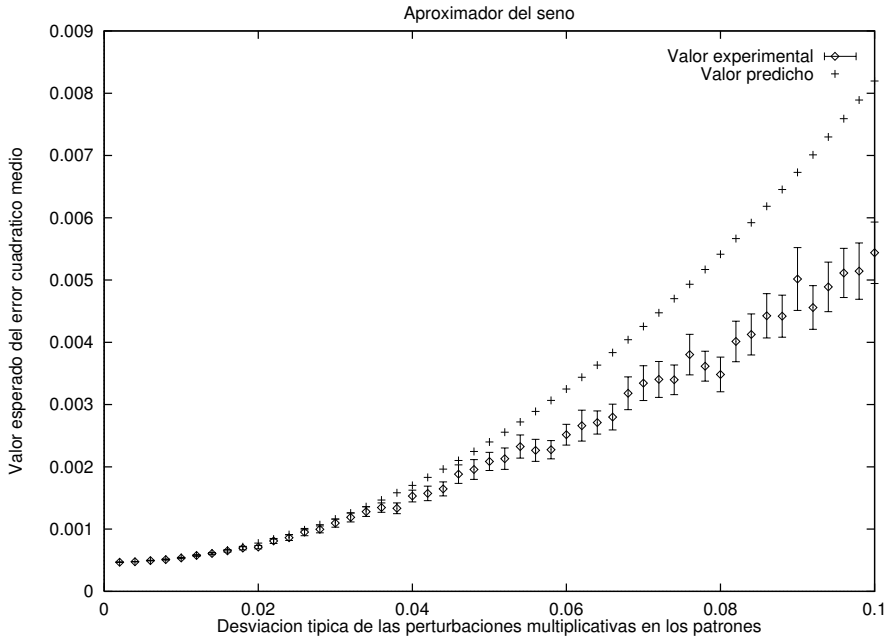

Figura 5.3. Error cuadrático medio simulado y predicho frente a σ.

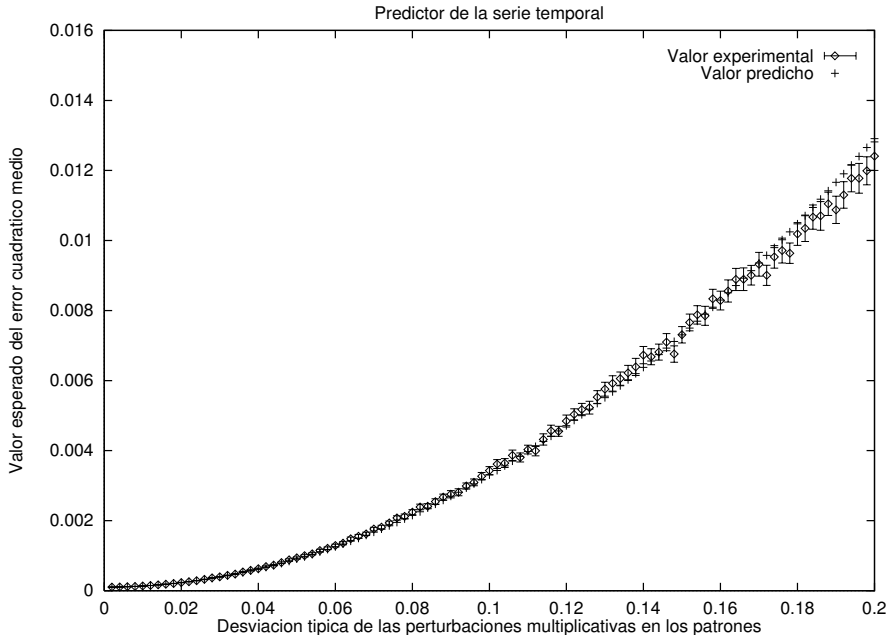

Figura 5.4. Error cuadrático medio simulado y predicho frente a σ.

En ambas figuras se aprecia que el valor predicho ajusta correctamente al valor experimental, al menos en el margen considerado. Al crecer σ este ajuste es peor, debido a que para el cálculo de la sensibilidad estadística se han supuesto pequeñas desviaciones.

5.4. LA SCM A RUIDO EN LAS ENTRADAS COMO MEDIDA DE LA CAPACIDAD DE GENERALIZACIÓN

La perturbación de las entradas durante el entrenamiento para aumentar la capacidad de generalización de un MLP ha sido propuesta en [MAO93,EDW97]. El objetivo es conseguir, de esta manera, una suavización de la superficie del error cuadrático respecto de las entradas.

En efecto, si se realiza un desarrollo de Taylor del error cuadrático medio cuando se las entradas del MLP se perturban:

$$\varepsilon' = \varepsilon_0 + \frac{1}{2N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\frac{\partial\varepsilon(p)}{\partial y_i^0}\,\Delta y_i^0 + \frac{1}{4N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\sum_{j=1}^{N_0}\frac{\partial^2\varepsilon(p)}{\partial y_i^0\partial y_j^0}\,\Delta y_i^0\,\Delta y_j^0 + \cdots \quad (5.20)$$

siendo y_i^0 la entrada i, N_0 el número de entradas del MLP, ε(p) el error cuadrático para el patrón de entrada p, y ε_0 el ECM nominal. Al calcular el valor esperado de la expresión (5.20) se tiene que:

$$E[\varepsilon'] = \varepsilon_0 + \frac{1}{2N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\frac{\partial\varepsilon(p)}{\partial y_i^0}E[\Delta y_i^0] + \\ + \frac{1}{4N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\sum_{j=1}^{N_0}\frac{\partial^2\varepsilon(p)}{\partial y_i^0\partial y_j^0}E[\Delta y_i^0\ \Delta y_j^0] \tag{5.21}$$

Teniendo en cuanta que en los modelos de desviación considerados $E[\Delta y_i^0] = 0$ y que $E[\Delta y_i^0\ \Delta y_j^0] = 0$ cuando $i \neq j$, se obtiene:

$$E[\varepsilon'] = \varepsilon_0 + \frac{1}{4N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\frac{\partial^2\varepsilon(p)}{\partial (y_i^0)^2}E[(\Delta y_i^0)^2] \tag{5.22}$$

De esta manera, como en el modelo aditivo $E[(\Delta y_i^0)^2] = \sigma^2$ y en el multiplicativo se tiene que $E[(\Delta y_i^0)^2] = (\sigma y_i^0)^2$, sustituyendo en (5.22) se tiene finalmente:

$$E[\varepsilon'] = \varepsilon_0 + \frac{\sigma^2}{4N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}\frac{\partial^2\varepsilon(p)}{\partial (y_i^0)^2} \tag{5.23}$$

para perturbaciones aditivas y,

$$E[\varepsilon'] = \varepsilon_0 + \frac{\sigma^2}{4N_p}\sum_{p=1}^{N_p}\sum_{i=1}^{N_0}(y_i^0)^2\frac{\partial^2\varepsilon(p)}{\partial (y_i^0)^2} \tag{5.24}$$

para perturbaciones multiplicativas.

En las expresiones (5.23) y (5.24) se aprecia claramente como el efecto de perturbar las entradas durante el entrenamiento se manifiesta en una perturbación del error cuadrático y por tanto, es éste error perturbado el que se minimiza. Esta minimización implica una suavización de la curvatura de la superficie de error respecto de las entradas, ya que se ha obtenido una relación entre el error cuadrático perturbado por ruido y la matriz hessiana de dicho error respecto de las entradas. Este hecho es lógico, pues si la curva del error cuadrático en función de las entradas es plana, el error no se modifica cuando se produce un cambio en las entradas.

Es importante notar que precisamente este efecto de suavizado tiene efectos positivos sobre la capacidad de generalización. Cuando se produce sobreentrenamiento (“overfitting”) se observan

cambios bruscos de pendiente en la superficie de error respecto de las entradas. De esta forma, una superficie más suave suele implicar una mayor capacidad de generalización, puesto que de un punto a otro se pasa de manera uniforme, permitiendo que el MLP interpole correctamente los puntos intermedios.

Por este motivo, la introducción de ruido durante el entrenamiento actúa como una técnica de regularización *inherente al algoritmo* [MAO93]. En [EDW97B] se consideran los efectos mencionados sobre la capacidad de generalización.

En el capítulo anterior obtuvimos la relación explícita entre el error cuadrático medio sujeto a perturbaciones y la sensibilidad cuadrática media. Hemos visto que esta relación sigue siendo correcta en el caso de considerar que las perturbaciones se deben a ruido en las entradas. Así, si igualamos las expresiones (4.6) y (5.23) se obtiene:

$$SCMN = \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_0} \frac{\partial^2 \varepsilon(p)}{\partial (y_i^0)^2} \tag{5.25}$$

De idéntica manera, al igualar (4.6) y (5.24) obtenemos:

$$SCMN = \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{N_0} (y_i^0)^2 \frac{\partial^2 \varepsilon(p)}{\partial (y_i^0)^2} \tag{5.26}$$

Es decir la SCM a ruido en las entradas (SCMN) es una medida de la suavidad de las curvas de error cuadrático respecto de las entradas y por tanto, apropiada para medir la capacidad de generalización para un error cuadrático medio dado. Así, si se tienen dos MLPs que presentan el mismo ECM, generalizará mejor aquél que presente un menor valor de la SCMN.

Para visualizar este resultado hemos entrenado el MLP que implementa al aproximador del seno y calculado la SCMN considerando el modelo de ruido aditivo. En este experimento se ha tratado de forzar el sobreentrenamiento, por lo que sólo se han utilizado 13 patrones de entrenamiento y se han considerado 4000 épocas. En la Figura 5.5 se representa el error cuadrático en función de la entrada para dos configuraciones de pesos distintas que presentaban aproximadamente el mismo ECM (0.011) sobre el conjunto de test (distinto del conjunto de entrenamiento) y, sin embargo distinto valor de la SCMN. Puede observarse que el MLP con menor SCMN presenta pendientes menores. Este hecho indica que el error cometido se distribuye uniformemente entre todas las posibles entradas, es decir, se comete un error similar con cualquier entrada. En cambio el MLP con mayor SCMN proporciona un error muy pequeño para algunas entradas y muy

grande para otras, o sea, se ha especializado en ciertos patrones próximos a los utilizados para el entrenamiento.

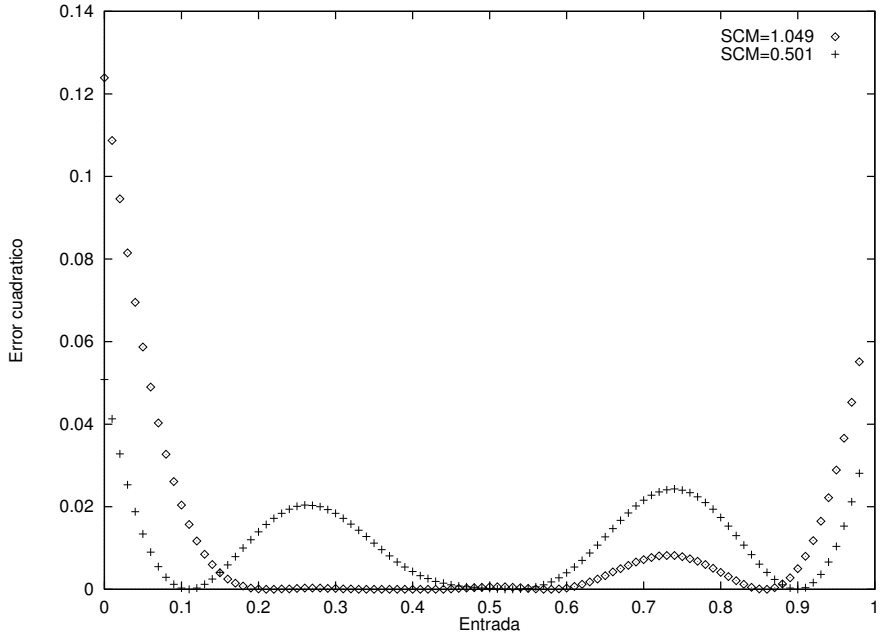

Figura 5.5. Error cuadrático en función de la entrada en MLPs con distinta SCM a ruido en las entradas.

En la Figura 5.6 se presenta la salida proporcionada por ambos MLPs junto a la salida deseada para ambos. Se observa que el MLP con menor SCMN comete un error más uniforme para cada patrón de entrada que aquél que presenta una SCMN mayor.

A partir de estas figuras se deduce que la expresión de la SCM a ruido en las entradas es apropiada para medir la capacidad de generalización de los MLPs. En MLPs que presenten un ECM similar, cuanto menor sea la SCMN mayor es la uniformidad del error cuadrático respecto de las posibles entradas, es decir, todos los patrones de entrada se reconocen en igual medida, por lo que el riesgo de sobreentrenamiento es menor.

Por tanto, un algoritmo que minimice la SCM a ruido en las entradas permitirá obtener redes más inmunes a tales perturbaciones y con una mayor capacidad de generalización. Nótese que en el caso de perturbaciones aditivas esta minimización se traducirá en un suavizado de la curva de error. En cambio, en el caso de perturbaciones multiplicativas este suavizado se hará de forma proporcional a la magnitud de las entradas, es decir, la curva será más suave cuanto mayor sea la magnitud de la entrada.

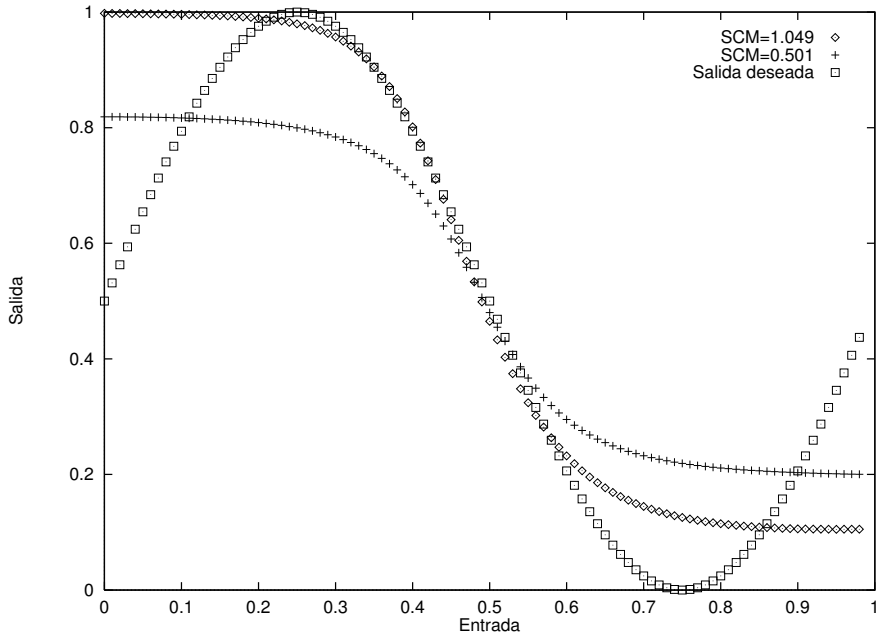

Figura 5.6. Salidas proporcionadas por MLPs con distinta SCM a ruido en las entradas.

5.5. ALGORITMO DE RETROPROPAGACIÓN ROBUSTO PARA PERTURBACIONES EN LAS ENTRADAS. REGLA ARRN.

En las secciones anteriores hemos obtenido y validado expresiones para evaluar la sensibilidad estadística a desviaciones en las entradas. También se ha comprobado que un menor valor de la SCMN implica una menor degradación del ECM a partir de la expresión (4.6), por lo que un MLP que presente un pequeño valor de la SCMN será más inmune a dicho ruido. Cuando el MLP se entrena con el algoritmo clásico (ARC) se minimiza el ECM pero, en general, se obtienen configuraciones de pesos que presentarán distintos valores de la SCMN con lo que unas serán más inmunes al ruido que otras. También se ha encontrado una relación entre la capacidad de generalización y la SCMN, de forma que un menor valor de ésta indica mejores prestaciones frente a aquella. Usando la misma filosofía que en capítulos anteriores se va a introducir la SCMN frente a perturbaciones en las entradas dentro de la regla de aprendizaje como otro término a minimizar conjuntamente con el ECM, lo cual equivale a minimizar el ECM sujeto a perturbaciones ε' tal como aparece en la expresión (4.8).

Para obtener esta nueva regla, a la que llamaremos ARRN, es preciso determinar el gradiente de la sensibilidad cuadrática a ruido en las entradas respecto de los pesos. Como las expresiones de la sensibilidad estadística son formalmente idénticas para el caso de desviaciones aditivas y multiplicativas, salvo en las condiciones iniciales, se va a hacer uso de la expresión (5.10) para

obtener el gradiente respecto de los pesos de las neuronas de la capa de salida y de los pesos de las neuronas de la capa oculta, y posteriormente se particularizarán las expresiones obtenidas para los dos modelos de desviación considerados

5.5.1. Cálculo del gradiente de la sensibilidad cuadrática a ruido en las entradas

Como sólo se van a considerar perceptrones con una capa oculta, primero se obtendrá el gradiente de la sensibilidad cuadrática respecto a un peso de la capa de salida y luego el gradiente respecto de pesos de la capa oculta. La sensibilidad cuadrática a ruido en las entradas para un patrón de entrada, $S_{c,}$ y viene dada por la siguiente expresión, calculada a partir de (5.10):

$$\begin{aligned} S_c &= \frac{1}{2} \sum_{r=1}^{N_2} (S_r^2)^2 \\ &= \frac{1}{2} \sum_{r=1}^{N_2} (\partial f_r^2)^2 \sum_{u=1}^{N_1} \sum_{v=1}^{N_1} w_{ru}^2 \; w_{rv}^2 \; C_{uv}^1 \end{aligned} \tag{5.27}$$

5.5.1.1. Neuronas de salida

Considérese un peso w_{ik}^2 de la capa de salida (m=2) que conecta a la neurona i con la neurona k de la capa oculta. Se debe calcular el gradiente de S_c respecto de dicho peso, en dicho cálculo se consideran constantes las derivadas de las funciones de activación.

$$
\begin{aligned}
(\nabla S_c)^2_{ik} &= \frac{1}{2}\,\frac{d}{dw^2_{ik}}\sum_{r=1}^{N_2}(S^2_r)^2 = \frac{1}{2}\,\frac{d(S^2_i)^2}{dw^2_{ik}} \\
&= \frac{1}{2}\,(\partial f^2_i)^2\,\frac{d}{dw^2_{ik}}\,(\sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w^2_{iu}\; w^2_{iv}\; C^1_{uv}) \\
&= \frac{1}{2}\,(\partial f^2_i)^2\,\sum_{u=1}^{N_1}\frac{d}{dw^2_{ik}}\,(w^2_{iu}\sum_{v=1}^{N_1} w^2_{iv}\; C^1_{uv}) \\
&= \frac{1}{2}\,(\partial f^2_i)^2\,(\sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w^2_{iv}\; C^1_{uv}\,\frac{dw^2_{iu}}{dw^2_{ik}} + \sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w^2_{iu}\; C^1_{uv}\,\frac{dw^2_{iv}}{dw^2_{ik}}) \\
&= \frac{1}{2}\,(\partial f^2_i)^2\,(\sum_{v=1}^{N_1} w^2_{iv}\; C^1_{kv} + \sum_{u=1}^{N_1} w^2_{iu}\; C^1_{uk}) \\
&= \frac{1}{2}\,(\partial f^2_i)^2\;2\sum_{u=1}^{N_1} w^2_{iu}\; C^1_{uk} = (\partial f^2_i)^2\sum_{u=1}^{N_1} w^2_{iu}\; C^1_{uk}
\end{aligned}
\tag{5.28}
$$

En el último paso se ha hecho uso de que los términos $C_{uk}^{\ 1}$ son simétricos, es decir, $C_{uk}^{\ 1} = C_{ku}^{\ 1}$ y que en las sumatorias los índices u y v recorren los mismos valores.

5.5.1.2. Neuronas de la capa oculta

Considérese un peso $w_{ik}^{\ 1}$ que conecta a una neurona i de la capa oculta (m=1) con una entrada primaria k. Se trata de obtener el gradiente de la expresión (5.20) respecto de dicho peso. Como en esta expresión, los términos dependientes de $w_{ik}^{\ 1}$ son los $C_{uv}^{\ 1}$, ya que el resto son constantes respecto de dicho peso, se va a proceder en primer lugar a la obtención de la derivada de $C_{uv}^{\ 1}$ respecto del peso $w_{ik}^{\ 1}$. En el desarrollo de dicha expresión se tiene en cuenta que los términos $C_{rs}^{\ 0}$ con $r \neq s$ son nulos, tanto en el modelo de desviación aditivo como en el multiplicativo.

$$
\begin{aligned}
\frac{dC_{uv}^{1}}{dw_{ik}^{1}} &= \frac{d}{dw_{ik}^{1}}\ \partial f_u^1\ \partial f_v^1\ \sum_{s=1}^{N_0} w_{us}^1\ w_{vs}^1\ C_{ss}^0 \\
&= \partial f_u^1\ \partial f_v^1\ \frac{d}{dw_{ik}^{1}}\ \sum_{s=1}^{N_0} w_{us}^1\ w_{vs}^1\ C_{ss}^0 \\
&= \partial f_u^1\ \partial f_v^1\ \sum_{s=1}^{N_0} \left(\frac{dw_{us}^{1}}{dw_{ik}^{1}}\ w_{vs}^1 + \frac{dw_{vs}^{1}}{dw_{ik}^{1}}\ w_{us}^1\right)\ C_{ss}^0 \\
&= (\partial f_i^1\ \partial f_v^1\ w_{vk}^1\ \delta_{iu} + \partial f_u^1\ \partial f_i^1\ w_{uk}^1\ \delta_{iv})\ C_{kk}^0 \\
&= (\partial f_v^1\ w_{vk}^1\ \delta_{iu} + \partial f_u^1\ w_{uk}^1\ \delta_{iv})\ \partial f_i^1\ C_{kk}^0
\end{aligned}
\tag{5.29}
$$

En la expresión anterior δ_{iu} y δ_{iv} representan la delta de Kronecker. Una vez obtenido el resultado (5.29) es inmediato calcular el gradiente de S_c respecto de w_{ik}^{1}.

$$
\begin{aligned}
(\nabla S_c)_{ik}^{1} &= \frac{1}{2}\ \frac{d}{dw_{ik}^{1}}\ \sum_{r=1}^{N_2} (S_r^2)^2 \\
&= \frac{1}{2}\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ \frac{d}{dw_{ik}^{1}}\ \left(\sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w_{ru}^2\ w_{rv}^2\ C_{uv}^1\right) \\
&= \frac{1}{2}\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ \sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w_{ru}^2\ w_{rv}^2\ \frac{dC_{uv}^{1}}{dw_{ik}^{1}} \\
&= \frac{1}{2}\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ \sum_{u=1}^{N_1}\sum_{v=1}^{N_1} w_{ru}^2\ w_{rv}^2\ (\partial f_v^1\ w_{vk}^1\ \delta_{iu} + \partial f_u^1\ w_{uk}^1\ \delta_{iv})\ \partial f_i^1\ C_{kk}^0 \\
&= \frac{1}{2}\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ \left(\sum_{v=1}^{N_1} w_{ri}^2\ w_{rv}^2\ \partial f_v^1\ w_{vk}^1 + \sum_{u=1}^{N_1} w_{ru}^2\ w_{ri}^2\ \partial f_u^1\ w_{uk}^1\right)\ \partial f_i^1\ C_{kk}^0 \\
&= \frac{1}{2}\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ 2\ \sum_{v=1}^{N_1} w_{ri}^2\ w_{rv}^2\ \partial f_v^1\ w_{vk}^1\ \partial f_i^1\ C_{kk}^0 \\
&= \partial f_i^1\ C_{kk}^0\ \sum_{r=1}^{N_2} (\partial f_r^2)^2\ w_{ri}^2\ \sum_{v=1}^{N_1} w_{rv}^2\ \partial f_v^1\ w_{vk}^1
\end{aligned}
\tag{5.30}
$$

A partir de las expresiones (5.28) y (5.30) se puede calcular el valor del gradiente de la sensibilidad cuadrática a desviaciones en las entradas del MLP, respecto de pesos pertenecientes a la capa de salida o a la capa oculta, respectivamente. A continuación se van a particularizar las expresiones para el modelo aditivo y el multiplicativo, sustituyendo los valores concretos de los términos C_{jk}^{m} para el modelo considerado.

5.5.2. Expresión del gradiente para el modelo de desviación aditivo

Los pesos de la neuronas de salida se deben actualizar según (5.28). En esa expresión, sustituyendo el valor de C_{uk}^{1} dada por el Corolario 5.5 se obtiene que:

$$\begin{aligned}(\nabla S_c)_{ik}^{2} &= (\partial f_i^2)^2 \sum_{u=1}^{N_1} w_{iu}^{2}\ \partial f_u^1\ \partial f_k^1 \sum_{v=1}^{N_0} w_{uv}^{1}\ w_{kv}^{1} \\ &= (\partial f_i^2)^2\ \partial f_k^1 \sum_{u=1}^{N_1} w_{iu}^{2}\ \partial f_u^1 \sum_{v=1}^{N_0} w_{uv}^{1}\ w_{kv}^{1}\end{aligned} \tag{5.31}$$

Para los pesos de la capa oculta deben utilizarse las condiciones iniciales correspondientes al modelo de desviación aditivo en la expresión (5.30) y sustituir los términos C_{kk}^{0}. Estos términos son iguales a 1, así que:

$$(\nabla S_c)_{ik}^{1} = \partial f_i^1 \sum_{r=1}^{N_2} (\partial f_r^2)^2\ w_{ri}^{2} \sum_{v=1}^{N_1} w_{rv}^{2}\ \partial f_v^1\ w_{vk}^{1} \tag{5.32}$$

5.5.3. Expresión del gradiente para el modelo de desviación multiplicativo

En este modelo, el valor de los C_{uk}^{1} necesario para particularizar la expresión (5.28) viene dada por el Corolario (5.11) y, haciendo uso de las condiciones iniciales, el valor de C_{kk}^{0} se expresa como $(y_k^0)^2$. De esta forma se obtienen las expresiones, (5.33) y (5.34) para el gradiente de los pesos de las neuronas de salida y de la capa oculta, respectivamente:

$$(\nabla S_c)^2_{ik} = (\partial f_i^2)^2 \; \partial f_k^1 \sum_{u=1}^{N_1} w_{iu}^2 \; \partial f_u^1 \sum_{v=1}^{N_0} w_{uv}^1 \; w_{kv}^1 \; (y_v^0)^2 \tag{5.33}$$

$$(\nabla S_c)^1_{ik} = \partial f_i^1 \; (y_k^0)^2 \sum_{r=1}^{N_2} (\partial f_r^2)^2 \; w_{ri}^2 \sum_{v=1}^{N_1} w_{rv}^2 \; \partial f_v^1 \; w_{vk}^1 \tag{5.34}$$

Por tanto, las expresiones (5.24) a (5.27) pueden utilizarse junto con el gradiente del ECM durante el entrenamiento para ajustar el valor de los pesos. El gradiente de la sensibilidad cuadrática, no obstante, debe multiplicarse por el factor de robustez γ para mantener la jerarquía de objetivos a minimizar, y permitir que la red converja.

En la siguiente sección se presentan los resultados obtenidos usando la regla ARRN en comparación con los que se obtienen cuando se usa ARC.

5.6. VALIDACIÓN DEL ALGORITMO ARRN

El algoritmo ARRN permite obtener configuraciones de pesos más inmunes al ruido de entrada que las que se obtienen con el algoritmo clásico. También se ha visto que minimizar la SCM a ruido en las entradas es beneficioso de cara a obtener un error cuadrático uniforme para cada patrón de entrada y así incrementar la capacidad de generalización. Hay que tener cuidado con este último aspecto, si en el aprendizaje se prima demasiado la SCMN puede pasar que las curvas de error cuadrático sean demasiado planas para el conjunto de entrenamiento, es decir, cada vector de entrada y los puntos próximos a él se reconocen como un mismo vector de salida, para proporcionar una mayor inmunidad al ruido, de forma que el ECM sobre el conjunto de test va a ser grande.

Así, el algoritmo ARRN muestra claramente su utilidad para la inmunidad al ruido en el caso de que el MLP implemente una función cuyos valores de entrada son discretos, de manera que cada patrón de entrada será reconocido como un mismo valor aunque el patrón haya sido modificado dentro de un cierto margen.

Para comparar el algoritmo propuesto ARRN con ARC se ha utilizado como ejemplo un MLP de tres capas que sintetiza la función XOR para cuatro variables. EL MLP considerado consta de 4 neuronas de entrada, 9 neuronas en la capa oculta y 1 neurona de salida. Se ha utilizado un

conjunto de 16 patrones de entrenamiento (cada una de las posibles salidas de la función) y se han realizado 5000 épocas durante el entrenamiento (en cada época se ha presentado el conjunto completo de patrones de entrenamiento).

Las neuronas del MLP se han tomado sin entrada de sesgo, el parámetro de aprendizaje α y el de momento β se han fijado ambos a 0.9. Se debe hacer notar que en todos los casos se ha obtenido un error de clasificación del 0%, por lo que se mostrará el ECM en las tablas correspondientes. Otro aspecto que merece la pena mencionar es que en la ejecución de ARRN, el factor de robustez se ha hecho variar dinámicamente durante el entrenamiento de forma distinta a como se hizo con los algoritmos ARR y ARRW , donde se usó la expresión (3.5). Con ARRN el factor de robustez γ se ha hecho decrecer lentamente durante el entrenamiento partiendo de un determinado valor inicial hasta un valor final pequeño, con objeto de permitir la convergencia del algoritmo. Esta estrategia es similar a la empleada con técnicas de enfriamiento simulado [BER95B, BER96] y, en este caso, el factor γ podría ser identificado con la temperatura. De este modo, cuando γ tiene un valor relativamente alto se permiten actualizaciones de los pesos que priman la resistencia al ruido frente al aprendizaje pero, conforme se va bajando el valor de este parámetro, el aprendizaje prima sobre la inmunidad al ruido, de forma que finalmente el algoritmo converge a una solución que da similares resultados en cuanto a ECM respecto de ARC, y que a su vez presenta un valor más pequeño de la SCMN.

A continuación, en los apartados 5.5.1 y 5.5.2 se muestran los resultados obtenidos con ruido aditivo y ruido multiplicativo, respectivamente. El Apartado 5.5.3 proporciona los resultados relacionados con el incremento de la capacidad de generalización por el uso de ARRN.

5.6.1. Algoritmo ARRN con ruido aditivo en las entradas

Como se ha mencionado antes, se va a utilizar un MLP que sintetiza la función XOR para 4 variables de entrada. El factor γ se ha disminuido desde un valor inicial hasta un valor pequeño final cuando se usa ARRN. Para cada valor inicial de γ se han realizado 40 pruebas. La Tabla 5.5 recoge los resultados obtenidos para el ECM y la SCMN junto con las respectivas desviaciones típicas muestrales para los distintos valores de γ considerados. La primera fila (γ=0) se refiere al algoritmo clásico ARC.

Puede apreciarse que la utilización de ARRN repercute en una disminución drástica de la SCM a ruido en las entradas, si bien el ECM es similar. Comparando los resultados obtenidos con ARC (γ=0) y los correspondientes a ARRN cuando el valor inicial de γ es 1.0, se puede ver que aunque el ECM obtenido con este último es ligeramente superior (1.12 veces mayor), sin embargo, la SCMN obtenida mediante ARRN es 18.29 veces inferior. Esto implica que, en presencia de

ruido aditivo en las entradas, el ECM de MLPs entrenados con ARC se degrada unas 18 veces más que los entrenados con ARRN.

Aunque, al tratarse de un "clasificador" el ECM no sea la medida más apropiada de las prestaciones respecto al aprendizaje, se utiliza como figura de error porque, como se ha comprobado en capítulos anteriores, una menor degradación del ECM suele implicar una menor degradación del error de clasificación, a pesar de no estar relacionados linealmente. También, debe tenerse en cuenta que el error de clasificación obtenido en todos los casos fue del 0%.

Tabla 5.5. Resultados de ARRN con ruido aditivo.

	ECM		SCMN	
γ	Media	Desv.Típ	Media	Desv Típ.
0	0.0687	0.0006	0.512	0.067
0.2	0.0746	0.0004	0.080	0.011
0.4	0.0754	0.0004	0.055	0.009
0.6	0.0760	0.0004	0.041	0.008
0.8	0.0763	0.0003	0.034	0.007
1	0.0766	0.0003	0.028	0.005

5.6.2. Algoritmo ARRN con ruido multiplicativo en las entradas

De forma similar al apartado anterior, se han considerado distintos valores iniciales para γ. La Tabla 5.6 muestra los resultados obtenidos con ARC (γ=0) y con ARRN. Para cada valor de γ se han realizado 40 pruebas.

En este caso, los resultados muestran que el ECM obtenido con ambos algoritmos es similar y la SCMN obtenida con ARRN para γ=1.0 es menor unas 6.74 veces en media que la obtenida con ARC.

Tabla 5.6. Resultados de ARRN con ruido multiplicativo.

	ECM		SCMN	
γ	Media	Desv.Típ	Media	Desv.Típ
0	0.0687	0.0005	0.0479	0.0014
0.2	0.0582	0.0104	0.0150	0.0171
0.4	0.0694	0.0006	0.0085	0.0013
0.6	0.0696	0.0005	0.0081	0.0011
0.8	0.0698	0.0004	0.0078	0.0009
1	0.0701	0.0004	0.0071	0.0009

5.6.3. Incremento de la capacidad de generalización usando ARRN

Se ha probado anteriormente que a igualdad de prestaciones respecto del ECM, cuanto menor sea la SCM a ruido en las entradas, menor es la pendiente de las curvas de error respecto de las entradas al MLP. Este hecho se traduce en una mayor uniformidad del aprendizaje para cada vector de entrada, de forma que aunque el ECM de dos MLPs sea similar, si uno de ellos presenta un valor de la SCMN grande, el error cuadrático cometido con cada patrón de entrada puede ser muy diferente, es decir, unos patrones los clasifica muy bien y otros, en cambio, muy mal. Es decir, el uso de ARRN evita la especialización del MLP en determinados patrones y, por tanto, incrementa la capacidad de generalización.

Este hecho puede ser así aprovechado durante el entrenamiento para obtener configuraciones de pesos con mayor capacidad de generalización. Para probarlo se ha utilizado el MLP que aproxima la función seno. Para el entrenamiento se han utilizado únicamente 13 patrones con salida conocida y se han invertido 4000 épocas, considerando perturbaciones aditivas en las entradas y presentando entradas de sesgo todas las neuronas. Se ha experimentado con distintos valores iniciales del factor de robustez, y para cada experimento se han realizado 40 entrenamientos. La Tabla 5.7 resume los resultados mostrando el valor inicial de γ utilizado, el ECM sobre un conjunto de test de 100 vectores y la SCMN de dicho conjunto.

En la Tabla 5.7 se aprecia el aspecto anteriormente comentado. La minimización de la SCMN es beneficiosa para incrementar la capacidad de generalización y de hecho mejora las prestaciones de aprendizaje. Con ARC se obtiene un ECM 2.38 veces mayor que con ARRN cuando γ=0.02

(0.0112 frente 0.0047). Se aprecia que al hacer mayor el valor de γ, el ECM es menor aunque la SCMN disminuye. Ello se debe a que al primar más la SCMN se degrada el ECM aunque éste error se hace más uniforme para los distintos vectores de entrada, es decir, el error de salida (respecto de la salida deseada) es similar para cualquier patrón de entrada.

Tabla 5.7. Estudio de la capacidad de generalización.

	ECM		SCMN	
γ	Media	Desv.Típ	Media	Desv.Típ
0	0.0112	0.0027	1.0663	0.1633
0.02	0.0047	0.0001	1.3723	0.0184
0.04	0.0058	0.0001	1.0817	0.0157
0.06	0.0067	0.0003	0.8992	0.0270
0.08	0.0076	0.0006	0.7710	0.0322
0.10	0.0084	0.0006	0.6824	0.0248

La Figura 5.7 representa la salida proporcionada por un MLP entrenado con el algoritmo clásico y por otro entrenado con ARRN. También se muestra la salida deseada para cada vector de entrada. Se aprecia claramente como aquél que ha sido entrenado con ARRN presenta una mayor capacidad de generalización.

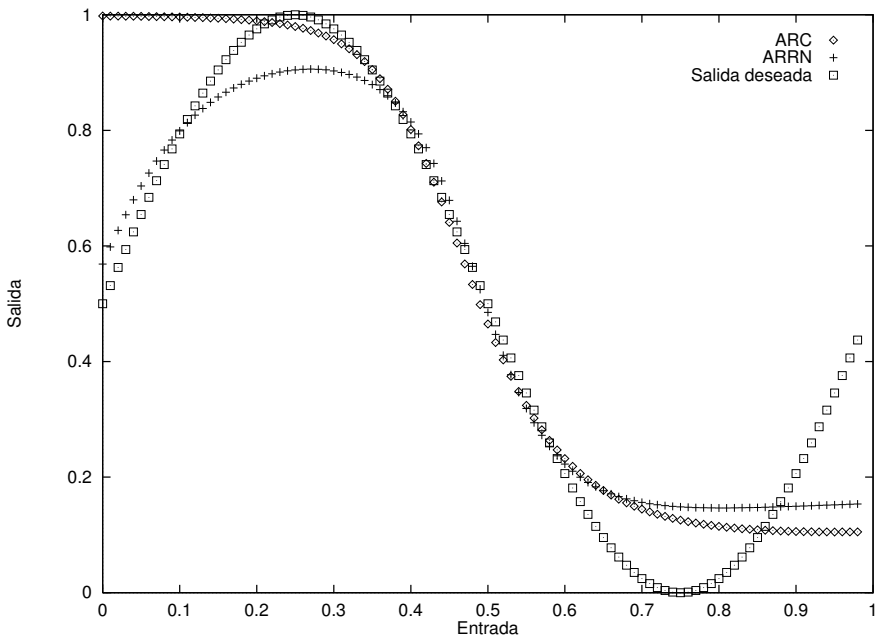

Figura 5.7. Salidas proporcionadas por MLPs entrenados con ARC y ARRN.

Capítulo 5

Estudio de las Perturbaciones en las Entradas y de la Capacidad de Generalización: Regla ARRN

5.7. CONCLUSIONES

En este capítulo se ha estudiado la influencia del ruido en las entradas de los MLPs. Se han obtenido expresiones para evaluar la inmunidad que presenta un MLP a dicho ruido, considerando perturbaciones en las entradas de tipo aditivo y multiplicativo. Mediante la sensibilidad cuadrática media a ruido en las entradas (SCMN) es posible predecir cómo se degradará el error cuadrático medio (ECM) que presenta el MLP tras el entrenamiento cuando las entradas del mismo son perturbadas.

Se han validado las expresiones de la SCMN para perturbaciones aditivas (5.15) y multiplicativas (5.19) mediante distintos experimentos. Se ha podido comprobar la bondad de la predicción de la degradación del ECM obtenida mediante (4.6) comparada con la calculada experimentalmente. De esta forma, la expresión de la SCMN (6) puntualizada para el caso de ruido en las entradas puede ser utilizada como figura de mérito para medir la inmunidad al ruido de un MLP concreto.

También se ha identificado la SCMN con la curvatura de las curvas de error cuadrático respecto de los vectores de entrada y pondrá en con la capacidad de generalización, ya que para MLPs con un valor similar del ECM, aquél con menor valor de la SCMN presenta un reparto más uniforme del aprendizaje respecto de los patrones de entrada, o sea, el error respecto de la salida deseada cometido para cada patrón de entrada es similar, de forma que la especialización en ciertos patrones es menor.

Por otra parte, se ha derivado un algoritmo de aprendizaje llamado ARRN que, en la misma línea que ARC y ARRW, permite obtener configuraciones de pesos más inmunes al ruido que las que se obtienen mediante un algoritmo clásico de retropropagación, y con mayor capacidad de generalización. Para ello se han obtenido las expresiones del gradiente de la SCMN respecto del valor de los pesos para los dos modelos de ruido considerados.

Hasta ahora, se ha considerado la sensibilidad estadística a desviaciones en los pesos y de hecho se ha supuesto que las entradas al MLP no presentaban errores. En este capítulo, sin embargo, se va a considerar el caso en que las entradas aplicadas al MLP puedan presentar perturbaciones y siguiendo la filosofía de capítulos anteriores, se van a obtener las expresiones correspondientes a la sensibilidad estadística a ruido en las entradas (Sección 5.2), que serán validadas en la siguiente sección (Sección 5.3). En la Sección 5.4 se muestra la relación existente entre la capacidad de generalización y la SCMN a ruido en las entradas, proponiéndose el uso de ésta como medida cuantitativa de aquélla. Se presenta un algoritmo similar a ARRW para maximizar la ECM frente a cambios en las entradas al que llamamos ARRN (Sección 5.5). La Sección 5.6 muestra los resultados obtenidos con ARRN, comparándolos a los que se obtienen con el algoritmo clásico ARC y en la Sección 5.7 se resumen las conclusiones.

También puede usarse ARRN para aumentar la capacidad de generalización y las prestaciones de aprendizaje, ya que al suavizar las curvas de error se evitan pendientes pronunciadas, de forma que el error cuadrático obtenido es similar para cualquier vector de entrada, evitándose así la especialización del MLP en determinados patrones de entrada.

Nótese que, como se ha mencionado previamente, la perturbación de los patrones de entrenamiento es un procedimiento considerado efectivo para incrementar la capacidad de generalización e incluso la tolerancia a fallos. Ello se debe, a que la introducción de ruido durante el entrenamiento se traduce en una perturbación implícita del ECM [EDW96], de forma que al modificar los patrones de entrada en realidad lo que se minimiza es el ECM sujeto a perturbaciones. Con el algoritmo propuesto, ARRN, se realiza también esta minimación pero siendo en este caso la perturbación del ECM explícita. Es decir, ambos procedimientos son equivalentes, si bien con ARRN la perturbación del ECM es modulada por la SCMN, en vez de ser aleatoria. De esta forma la perturbación de las entradas durante el entrenamiento puede considerarse como un caso particular de ARRN, donde la SCMN se aproxima aleatoriamente.

La minimización del ECM sujeto a perturbaciones tiene como consecuencia secundaria el incrementar la tolerancia a fallos. Si se observan las expresiones de la SCM a desviaciones en los pesos (SCMW) y se comparan con las correspondientes a la SCM a ruido en las entradas (SCMN) puede observarse su similitud.

Por ejemplo, las expresiones de la sensibilidad estadística para el modelo de desviación multiplicativo en las neuronas de la capa oculta son idénticas en el caso de perturbaciones en los pesos y perturbaciones en las entradas, de forma que el MLP no distingue una perturbación multiplicativa en las entradas de una desviación en los pesos de la capa oculta. Así, la minimización de la inmunidad al ruido multiplicativo en las neuronas de la capa oculta implica una minimización de su tolerancia a fallos y viceversa. En los demás casos no existe una relación tan obvia, pero se observan claramente ciertas similitudes en las respectivas expresiones para la SCM.

Por ello, aunque con ARRW pueda obtenerse como efecto secundario un beneficio característico de ARRN (mayor capacidad de generalización, mayores prestaciones de aprendizaje e inmunidad al ruido) y viceversa (respecto de tolerancia a fallos), debe utilizarse el algoritmo adecuado según el propósito deseado, o sea, ARRW para incrementar la tolerancia a fallos y ARRN para incrementar la inmunidad al ruido, o la combinación de ambos si se requieren ambas cosas.

Capítulo 6

Comparación de ARRW con otros Procedimientos

En este capítulo se va a comparar el procedimiento ARRW, propuesto en el Capítulo 4 con otro procedimiento [MUR94] que perturba los pesos mediante ruido durante el entrenamiento a fin de obtener una mayor tolerancia. Dicho procedimiento ha sido ampliamente estudiado y comparado a su vez con otros algoritmos [EDW95] mostrándose claramente superior. En la Sección 6.1 se realiza una introducción, presentándose a continuación en la Sección 6.2 el algoritmo que se va a utilizar para comparar con ARRW. A continuación, en la Sección 6.3 se presentan los resultados de ECM y SCM obtenidos con estos métodos y con el algoritmo ARC. En la Sección 6.4 se estudia otro aspecto derivado del uso de ARRW como es el incremento de la capacidad de generalización y, en la Sección 6.5 se muestran resultados de aplicar ARRW a problemas reales normalizados usados como benchmarks. Finalmente, en la Sección 6.6 se muestran las conclusiones.

6.1. INTRODUCCIÓN.

El incremento de la tolerancia a fallos de los MLPs ha sido estudiado por diversos autores [PHA95,CHI94,MUR93,EDW95] y se han propuesto distintas alternativas para incrementar dicha tolerancia. Así, en [PHA95] se propone incrementar la redundancia de la red teniendo en cuenta qué elementos son los que producen mayor degradación de las prestaciones con objeto de replicarlos.

En [CHI94] se propone un procedimiento en el que los pesos mantienen valores pequeños en magnitud. No obstante, esta alternativa, como los propios autores indican, implica una reducción de las prestaciones en cuanto a aprendizaje por lo que debe ir acompañada de un incremento del número de neuronas de la capa oculta. La razón por la que una menor magnitud de los pesos implica una mayor robustez se ha justificado en el Capítulo 2 a partir de la expresión de la sensibilidad estadística, al comprobar que ésta disminuye con la magnitud de los pesos.

El objetivo principal en un procedimiento para aumentar la robustez de un MLP debe ser repartir equitativamente el aprendizaje entre sus neuronas constituyentes porque así, al intervenir los distintos pesos de forma similar en la función realizada por el MLP, el perjuicio es menor si falla uno de sus elementos [EDW98C]. Lo que está claro es que esta distribución del aprendizaje debe realizarse mediante alguna modificación de dicho proceso de aprendizaje ya que el algoritmo clásico de entrenamiento del MLP no asegura que dicha distribución sea óptima en tal sentido [EDW97].

También se ha mostrado que el simple aumento del número de neuronas en la capa oculta, del número de capas, o ambos no implica en principio un incremento de la tolerancia a faltas. Este hecho ha sido constatado por algunos autores. En [STE90] se concluye que la probabilidad de error en MLPs con función de activación tipo escalón no depende del número de neuronas en la capa oculta; y en [EDW98B] los autores aseguran que el incremento de neuronas en MLPs, no sólo no implica un aumento de la tolerancia, sino que incluso puede degradarla debido a que existen más elementos susceptibles de fallar. Lo que sí se obtiene con redes mayores es una mayor *tolerancia potencial* ya que al aumentar los grados de libertad, de un reparto óptimo del aprendizaje entre un mayor número de neuronas se obtendría una mayor robustez [EDW98C].

De esta manera, en [EDW95] se utiliza un método que tiene por objeto disminuir la sensibilidad de salida. La sensibilidad de salida, definida como la derivada de la salida del perceptrón respecto del valor de un determinado peso, mide la dependencia de la salida del MLP respecto del valor de dicho peso. Este procedimiento, que usa una filosofía similar a la usada en los algoritmos ARR y ARRW, pretende aumentar la tolerancia del MLP forzando una mejor distribución del

aprendizaje, ya que al penalizar el aprendizaje de los pesos más influyentes para un patrón dado, el resto de los pesos deben asimilar dicho aprendizaje. No obstante, en este procedimiento las prestaciones de aprendizaje se degradan en cierta medida y, por otro lado, la ganancia en cuanto a tolerancia a fallos no es muy acusada.

En [EDW95] se propone utilizar la saliencia (*saliency*) como medida de la tolerancia a fallos. La saliencia de un peso, definida como la segunda derivada del error cuadrático respecto del valor de un peso, es una medida de la curvatura de la superficie de error. Esta medida se basa en la relación probada experimentalmente entre la capacidad de generalización y la tolerancia a fallos de los MLPs. Así, en [EDW95, EDW97] como medida de la robustez de un MLP se utiliza la media de la suma de los elementos de la diagonal de la matriz hessiana sobre todos los patrones de entrada. Cabe esperar que, si las pendientes de la superficie de error son pequeñas, las perturbaciones en los pesos apenas modificarán el error cuadrático de salida. La validez de la saliencia como medida cuantitava de la tolerancia a fallos se deduce a continuación:

Si se hace un desarrollo de Taylor del error cuadrático medio respecto de las perturbaciones en los pesos se obtiene:

$$\begin{aligned}\varepsilon' &= \varepsilon_0 + \frac{1}{2N_p}\sum_{p=1}^{N_p}\sum_{m=1}^{M}\sum_{i=1}^{N_m}\sum_{j=1}^{N_{m-1}}\frac{\partial\varepsilon(p)}{\partial w_{ij}^{m}}\,\Delta w_{ij}^{m} + \\ &+ \frac{1}{4N_p}\sum_{p=1}^{N_p}\sum_{m=1}^{M}\sum_{i=1}^{N_m}\sum_{j=1}^{N_{m-1}}\sum_{m'=1}^{M}\sum_{i'=1}^{N_{m'}}\sum_{j'=1}^{N_{m'-1}}\frac{\partial^2\varepsilon(p)}{\partial w_{ij}^{m}\partial w_{i'j'}^{m'}}\,\Delta w_{ij}^{m}\,\Delta w_{i'j'}^{m'} + \ldots\end{aligned} \tag{6.1}$$

siendo M el número de capas y N_m el número de neuronas de la capa m. ε_0 es el ECM nominal (cuando los pesos no están perturbados) y $\varepsilon(p)$ es el error cuadrático que presenta el MLP para un patrón de entrada p.

Si ahora se toman los valores esperados y se hace uso de que en los modelos de desviación considerados $E[\Delta w_{ij}^{m}] = 0$, y $E[\Delta w_{ij}^{m}\,\Delta w_{i'j'}^{m'}] = 0$ si los pesos son distintos, se verifica que:

$$E[\varepsilon'] = \varepsilon_0 + \frac{1}{4N_p}\sum_{p=1}^{N_p}\sum_{m=1}^{M}\sum_{i=1}^{N_m}\sum_{j=1}^{N_{m-1}}\frac{\partial^2\varepsilon(p)}{\partial (w_{ij}^{m})^2}\,E[(\Delta w_{ij}^{m})^2] \tag{6.2}$$

Considerando un modelo de desviación aditivo donde $E[(\Delta w_{ij}^{m})^2] = \sigma^2$ y sustituyendo en (6.2) se obtiene:

$$E[\varepsilon'] = \varepsilon_0 + \frac{\sigma^2}{4N_p} \sum_{p=1}^{N_p} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \sum_{j=1}^{N_{m-1}} \frac{\partial^2 \varepsilon(p)}{\partial (w_{ij}^m)^2} \tag{6.3}$$

donde el segundo sumando de la parte derecha es precisamente la suma de los elementos de la diagonal de la matriz hessiana, justificando por consiguiente que la medida de la saliencia es adecuada para la tolerancia a faltas de desviación aditivas.

Por otro lado, si la expresión (6.3) se iguala a (4.4), donde establecimos la relación entre el valor esperado del error cuadrático medio y la sensibilidad cuadrática media a perturbaciones en los pesos (SCMW) se verificará que:

$$\frac{1}{2} \sum_{p=1}^{N_p} \sum_{i=1}^{N_M} (S_i^M)^2 = \frac{1}{4} \sum_{p=1}^{N_p} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \sum_{j=1}^{N_{m-1}} \frac{\partial^2 \varepsilon(p)}{\partial (w_{ij}^m)^2} \tag{6.4}$$

es decir,

$$SCMW = \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \sum_{j=1}^{N_{m-1}} \frac{\partial^2 \varepsilon(p)}{\partial (w_{ij}^m)^2} \tag{6.5}$$

la SCMW puede identificarse con la saliencia cuando las perturbaciones son aditivas, ya que precisamente permite evaluar la curvatura de la superficie de error respecto de los pesos. De esta manera una menor SCMW implica pendientes más suaves, es decir, mínimos anchos. Así, en el supuesto de desviaciones aditivas ambas medidas son equivalentes.

Sin embargo, al considerar el modelo de desviación multiplicativo se tiene que $E[(\Delta w_{ij}^m)^2] = (\sigma w_{ij}^m)^2$, por lo que sustituyendo en (6.2):

$$E[\varepsilon'] = \varepsilon_0 + \frac{\sigma^2}{4N_p} \sum_{p=1}^{N_p} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \sum_{j=1}^{N_{m-1}} (w_{ij}^m)^2 \frac{\partial^2 \varepsilon(p)}{\partial (w_{ij}^m)^2} \tag{6.6}$$

En este caso, el uso de la matriz hessiana como medida de la tolerancia a fallos no es una medida totalmente adecuada, puesto que cada sumando está ponderado por $(w_{ij}^m)^2$, cuestión que no se tiene en cuenta en otros trabajos previos. De hecho, en [EDW97] se cuestiona que la saliencia sea proporcional a la magnitud de los pesos y se considera falso, a pesar de usar un modelo de perturbación multiplicativo en su procedimiento.

En cualquier caso, la expresión (6.6) también se puede relacionar con la SCMW para desviaciones multiplicativas, obteniendo que:

$$SCMW = \frac{1}{4N_p} \sum_{p=1}^{N_p} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \sum_{j=1}^{N_{m-1}} (w_{ij}^{\ m})^2 \frac{\partial^2 \varepsilon(p)}{\partial (w_{ij}^{\ m})^2} \tag{6.7}$$

Es decir, la SCMW también es una medida apropiada para medir el efecto de las perturbaciones multiplicativas puesto que considera todas las contribuciones.

A partir de las relaciones encontradas entre saliencia y SCMW se concluye que:

- La SCMW coincide con la saliencia en el caso de perturbaciones aditivas.
- La SCMW es una medida más precisa de la tolerancia a fallos multiplicativos que la saliencia.
- La SCMW está relacionada con la matriz hessiana del error cuadrático respecto del valor de los pesos, por lo que un menor valor de aquella implica una superficie de error más plana.
- La minimización de la SCMW es equivalente a encontrar un mínimo en el cual la curvatura de la superficie de error respecto del valor de los pesos es más suave. En el caso de desviaciones multiplicativas, esta suavización es además proporcional a la magnitud de los pesos.

Como se ha indicado en [EDW95, EDW96], perturbar los pesos durante el entrenamiento equivale a minimizar la saliencia (la SCMW en realidad, según se ha probado), por lo que este procedimiento puede considerarse como una técnica de regularización. A él nos referiremos como "Algoritmo de Retropropagación con RUido" (ARRU) y ha sido ampliamente estudiado en diversas publicaciones [MUR93, MUR94, EDW95, EDW96], mostrándose claramente superior a otros procedimientos propuestos. ARRU ha mostrado ser eficaz para aumentar la tolerancia a fallos de los MLPs y además presenta algunas ventajas adicionales como es una aceleración del aprendizaje y una mayor capacidad de generalización.

Debido a que dicho procedimiento ya ha sido comparado con otros métodos y a que se trata de un algoritmo bastante conocido, se ha implementado y utilizado para comparar los resultados con ARRW. En el siguiente apartado se va a describir el algoritmo ARRU. Posteriormente, en la Sección 6.3 se mostrarán los resultados de diversos experimentos que comparan ARC con ARRU y ARRW.

6.2. ALGORITMO DE RETROPROPAGACIÓN CON RUIDO EN LOS PESOS

Los pasos que sigue el algoritmo ARRU son básicamente los siguientes:

a) Fijar un valor inicial para el ruido r, normalmente se hace r_0 =0.4 [EDW95].

b) En cada época:

b.1) Fijar una perturbación para cada peso w_{ij}^{m} dentro de un margen ± r $w_{ij}^{m.}$

b.2) Para cada patrón de entrada:

b.2.1) Calcular el error cuadrático de salida considerando los pesos perturbados.

b.2.2) Restaurar los pesos a sus valores nominales (w_{ij}^{m}).

b.2.3) Modificar los pesos según el algoritmo de retropropagación clásico.

b.3) Decrementar el valor del ruido r.

Se puede observar que en este algoritmo se realiza una perturbación 'virtual' del valor de los pesos con objeto de perturbar la función de error cuadrático de salida. El valor de esta perturbación depende del ruido, el cual se va decrementando paulatinamente al final de cada época para asegurar la convergencia del entrenamiento. Normalmente la disminución del ruido es tal que al final del entrenamiento este disminuye hasta un valor de 0.01 [EDW95].

Usando este procedimiento se han realizado pruebas bastante exhaustivas en [EDW94,EDW95] observándose que es claramente superior en cuanto a tolerancia a fallos que otros procedimientos tales como el algoritmo clásico o la minimización de la sensibilidad de salida (recordemos que como se citó en el Apartado 4.1 no debe confundirse la sensibilidad de salida con la sensibilidad estadística). Se prueba en este trabajo que usando ARRU:

- Se obtienen configuraciones de pesos más tolerantes a fallos.
- Se acelera del aprendizaje.
- Se obtiene una distribución más equitativa del aprendizaje entre los pesos.
- Se obtiene una mayor capacidad de generalización.

Además, en [EDW98B] se realiza la prueba de trasladar configuraciones de pesos determinadas mediante simulación usando ARRU a configuraciones físicas concretas, demostrando que las prestaciones apenas se degradan, por lo que no es necesario realizar un nuevas iteraciones de entranemiento en la implementación física con objeto de ajustar las diferencias respecto de los valores obtenidos tras el entrenamiento.

Nótese que los resultados apuntados por los autores son semejantes a las conclusiones que se han presentado en los capítulos anteriores de la presente memoria. El objetivo ahora es comparar el algoritmo ARRU con el que hemos propuesto en el Capítulo 4 (ARRW).

Por otro lado, se pueden apreciar semejanzas entre ARRU y ARRW (o ARR). En el apartado anterior hemos comprobado la relación existente entre la SCMW y la saliencia, mostrando que la SCMW es más adecuada como medida de la tolerancia a fallos. Los autores de [EDW96] prueban que la perturbación en los pesos equivale a perturbar la función de error, y presentan los términos correspondientes a dicha perturbación. Así, durante el aprendizaje, el algoritmo de retropropagación minimiza implícitamente dichos términos. Estos términos podrían ser identificados con los correspondientes al gradiente de la SCMW. El factor de robustez se puede relacionar con la magnitud del ruido y éste, al igual que en ARRU, va disminuyendo en el transcurso del aprendizaje.

Nótese que el desarrollo de ARRW se ha realizado a partir de las expresiones de la sensibilidad estadística y de su relación con la degradación del ECM cuando los pesos están sujetos a desviaciones. En cambio, en el desarrollo de ARRU se propone un procedimiento ad hoc y en vista de sus resultados, se ha buscado con posterioridad la base matemática que lo explique. En particular, en [EDW96] se prueba que la adición de ruido sináptico equivale a perturbar la función de error, y se argumenta que el procedimiento ARRU equivale a minimizar la matriz hessiana del error cuadrático respecto del valor de los pesos. Como hemos probado, este hecho no es totalmente cierto en el caso de desviaciones multiplicativas que, por cierto, son las que se utilizan para perturbar los pesos durante el entrenamiento en ARRU.

Las principales diferencias entre ARRU y ARRW son las siguientes:

- ARRU se basa en un procedimiento aleatorio puesto que así lo son las perturbaciones introducidas, aunque éstas están dentro de un cierto margen. Sien embargo, ARRW es un procedimiento determinista una vez fijados los parámetros del algoritmo de aprendizaje y las condiciones iniciales.

- Existe una relación implícita entre ARRU y un procedimiento que minimiza las perturbaciones del ECM tal y como se prueba a posteriori. En cambio, ARRW se ha

obtenido a partir de una relación explícita entre las perturbaciones del ECM y la SCMW.

En concreto, ARRU puede considerarse como un caso particular de ARRW, puesto que ARRW puede verse es un procedimiento donde los pesos se consideran perturbados con un cierto ruido, pero donde esta perturbación, a diferencia de ARRU, *se contabiliza a través de la sensibilidad estadística* a perturbaciones en los pesos, en lugar de tener que inyectarse aleatoriamente en varias iteraciones. En otros términos, ARRU es una aproximación de ARRW.

El término añadido a la función de error en ARRW en realidad puede contemplarse como un término de regularización, siendo posible así justificar la mejora respecto a tolerancia a faltas obtenida como consecuencia del suavizado que se provoca en la función de salida de la red, como en otros trabajos previos se indica [MUR94, EDW95, EDW98A]. De acuerdo a la clasificación de técnicas de regularización presentada en [MAO93], ARRW puede considerarse como un procedimiento de regularización explícitamente especificado (Tipo III) aunque el estabilizador ha sido incluido pensando en un incremento de la tolerancia a faltas, más que en incrementar la capacidad de generalización. En este sentido, ARRW presenta la misma filosofía que se utiliza en [EDW98A] para la hessiana del error cuadrático, pero difiere de ARRU, ya que éste puede clasificarse como una técnica de regularización inherente al algoritmo (Tipo II).

La perturbación del ECM durante el entrenamiento también se puede obtener implícitamente perturbando los patrones de entrada tal como se ha vistpo en el capítulo anterior, y ésta es la razón por la que la adición de ruido en las entradas durante el aprendizaje repercuta en un cierto incremento de la tolerancia a fallos.

Como en ARRU, se usa un modelo de desviación multiplicativo (paso b.1 del algoritmo) para comparar sus resultados con ARRW, y se utilizarán las expresiones de sensibilidad estadística correspondientes a dicho modelo. En la siguiente sección se compararán los resultados de SCMW y ECM obtenidos tras el entrenamiento con ambos procedimientos.

6.3. COMPARACIÓN DE ARRW CON ARRU

Para la comparación se han utilizado los tres MLPs descritos en la Tabla 4.5: el aproximador del seno, el predictor de la serie temporal y el clasificador del problema de Hart.

En el caso de ARRU, aunque en [EDW95] se aconseja un valor inicial para el ruido, r_0, de 0.4, hemos considerado diversos valores para r_0 con objeto de realizar un análisis más exhaustivo. Este nivel de ruido se ha hecho decrecer durante el entrenamiento hasta alcanzar finalmente un

valor aproximadamente igual a 0.01. Igualmente, para ARRW hemos considerado distintos valores de la constante K del factor de robustez. Para cada valor de r_0 y para cada valor de K se han realizado 40 entrenamientos partiendo de distintos valores iniciales de los pesos con ARRU y ARRW, respectivamente.

Como ARRU usa un modelo de desviación multiplicativo, ARRW se ha particularizado para tal modelo.

6.3.1. Resultados para el aproximador del seno.

La Tabla 6.1 muestra los resultados obtenidos con ARC, la Tabla 6.2 los obtenidos con ARRU para los distintos valores de ruido inicial considerados y la Tabla 6.3, los de ARRW para distintos valores de la constante K. Estas tablas muestran el error cuadrático medio (ECM) y la sensibilidad cuadrática media a desviaciones en los pesos (SCMW) medidos sobre el conjunto de patrones de test tras el entrenamiento. También se muestran las desviaciones típicas muestrales de cada variable.

Tabla 6.1. Resultados con el algoritmo clásico.

ECM	SCMW
0.000818 ± 0.000307	3.1226 ± 0.7144

Tabla 6.2. Resultados del algoritmo con ruido (ARRU).

r_o	ECM	SCMW
0.1	0.000712 ± 0.000369	3.0997 ± 0.5996
0.2	0.000707 ± 0.000393	2.4739 ± 0.5667
0.3	0.000495 ± 0.000449	2.3768 ± 0.6098
0.4	0.000575 ± 0.001263	2.0372 ± 0.4264
0.5	0.000704 ± 0.001315	1.7465 ± 0.4308
0.6	0.001514 ± 0.002310	1.3950 ± 0.2947
0.7	0.002930 ± 0.003378	1.0323 ± 0.2194
0.8	0.005598 ± 0.003637	0.8033 ± 0.2777
0.9	0.005788 ± 0.003692	0.7638 ± 0.2349
1	0.003930 ± 0.003911	0.7191 ± 0.2532

Tabla 6.3. Resultados del algoritmo robusto (ARRW).

K	ECM	SCMW
0.1	0.000680 ± 0.000295	2.7091 ± 0.6272
0.2	0.000550 ± 0.000318	2.5964 ± 0.6138
0.3	0.000553 ± 0.000241	2.1607 ± 0.5295
0.4	0.000620 ± 0.000255	1.8088 ± 0.3383
0.5	0.000658 ± 0.000304	1.4953 ± 0.3140
0.6	0.000739 ± 0.000290	1.1979 ± 0.2507
0.7	0.000809 ± 0.000311	0.9651 ± 0.1708
0.8	0.001139 ± 0.001420	0.8251 ± 0.1944
0.9	0.000975 ± 0.000383	0.7039 ± 0.1286
1	0.000910 ± 0.000173	0.6580 ± 0.0957

Para comparar los resultados vamos a fijarnos en aquellas filas que proporcionen un valor para el ECM similar al obtenido con ARC (0.000810): con ARRU se obtiene un ECM igual a 0.000704 y la SCMW 1.7465 cuando el nivel inicial de ruido, r_0, se hace igual a 0.5. Usando ARRW, en la fila correspondiente a un valor de K igual a 0.7 se obtiene un ECM de 0.000809 y la SCMW vale 0.9651, un valor claramente inferior al obtenido con ARRU (1.8 veces menor).

En cualquier caso se observa que la SCMW es menor que la obtenida con ARC, y que además disminuye al aumentar r_0 o K, según se trate de ARRU o ARRW, respectivamente. Para valores inferiores a un cierto r_0 o K se obtienen también mejores prestaciones respecto a aprendizaje que con el algoritmo clásico.

La Figura 6.1 representa el ECM frente a la SCMW obtenidos con los tres algoritmos (ARC, ARRU y ARRW). El ECM se muestra con su desviación típica en el eje de ordenadas y la SCMW en el de abcisas. Puede observarse que para valores de ECM similares a los de ARC, los menores valores de SCMW se obtienen con ARRW. Con ARRU, la minimización de la SCMW se hace a costa de una gran degradación de las prestaciones en cuanto a ECM. También se observa que la desviación típica obtenida para el ECM con ARRW es muy inferior a la de ARRU, lo que parece indicar que cuando se usa ARRW para entrenar el MLP la solución converge a un mismo mínimo y por tanto, la probabilidad de estancarse en mínimos locales con alta SCMW es inferior que cuando se usa ARRU.

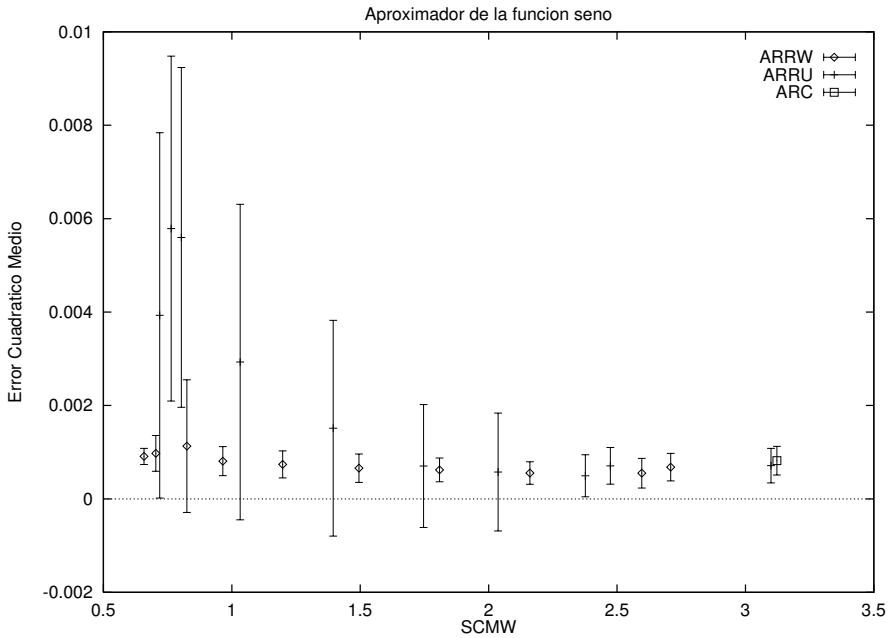

Figura 6.1. ECM frente a SCMW para distintos valores de K (ARRW) o de r_0 (ARRU).

6.3.2. Resultados para el predictor de la serie temporal de Mackey-Glass.

Los resultados obtenidos para el ECM y la SCMW mediante ARC, ARRU y ARRW se muestran en las Tablas 6.4, 6.5 y 6.6, respectivamente.

En este caso, para comparar resultados, como los valores de ECM obtenidos con los distintos algoritmos son muy similares, vamos a escoger los que proporcionan una menor SCMW. Con ARC se obtiene un valor de la SCMW de 0.4547, superior al obtenido con ARRU cuando r_0 es igual a 1.0 (0.3074). Sin embargo, cuando se usa ARRW este valor se hace aun menor (0.1991 cuando K vale 1.0).

Tabla 6.4. Resultados con el algoritmo clásico.

ECM	SCMW
0.000106 ± 0.000012	0.4547 ± 0.2296

Tabla 6.5. Resultados del algoritmo con ruido (ARRU).

r_o	ECM	SCMW
0.1	0.000120 ± 0.000036	0.4897 ± 0.2267
0.2	0.000122 ± 0.000028	0.4489 ± 0.1876
0.3	0.000118 ± 0.000021	0.4144 ± 0.1513
0.4	0.000114 ± 0.000022	0.3620 ± 0.1122
0.5	0.000115 ± 0.000017	0.3462 ± 0.1584
0.6	0.000115 ± 0.000038	0.3160 ± 0.1471
0.7	0.000113 ± 0.000021	0.3080 ± 0.1427
0.8	0.000109 ± 0.000012	0.2887 ± 0.1682
0.9	0.000106 ± 0.000012	0.2792 ± 0.1364
1	0.000110 ± 0.000012	0.3074 ± 0.1387

Tabla 6.6. Resultados del algoritmo robusto (ARRW).

K	ECM	SCMW
0.1	0.000115 ± 0.000017	0.3622 ± 0.6272
0.2	0.000125 ± 0.000035	0.2700 ± 0.6138
0.3	0.000115 ± 0.000031	0.2484 ± 0.5295
0.4	0.000116 ± 0.000031	0.2358 ± 0.3383
0.5	0.000117 ± 0.000036	0.2354 ± 0.3140
0.6	0.000115 ± 0.000029	0.2336 ± 0.2507
0.7	0.000121 ± 0.000032	0.2196 ± 0.1708
0.8	0.000121 ± 0.000046	0.2177 ± 0.1944
0.9	0.000122 ± 0.000019	0.2107 ± 0.1286
1	0.000123 ± 0.000042	0.1991 ± 0.0957

Nótese que aunque parezca que el ECM obtenido con ARRU es inferior al obtenido con ARRW, en realidad son muy similares, ya que la diferencia de sus medias es muy pequeña y sus

desviaciones típicas solapan. En cambio, las diferencias en la SCMW son más acusadas, y en algunos casos, ni siquiera solapan las correspondientes desviaciones típicas.

En la Figura 6.2 se puede apreciar el hecho anteriormente citado. Se representa el ECM en función de la SCMW. Puede observarse, que los resultados obtenidos con ARRW presentan una menor SCMW que los obtenidos con ARRU y, sin embargo, el ECM obtenido es similar. Se observa que las soluciones obtenidas con ARRW obtienen valores de la SCMW entre 0.1991 y 0.27 para K mayor que 0.2, mientras que en el caso de ARRU se obtienen valores entre 0.2792 y 0.4897. El ECM, en cambio está comprendido entre valores similares.

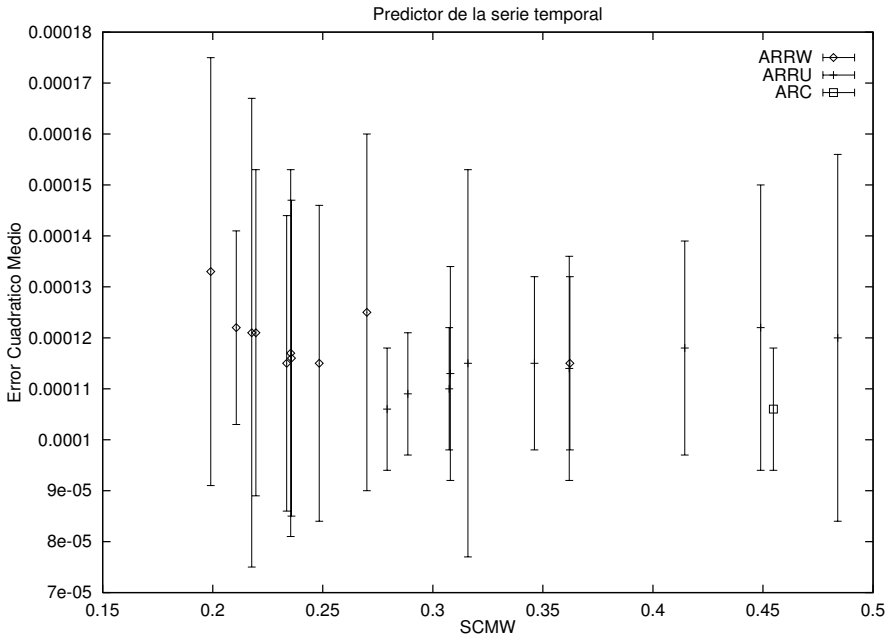

Figura 6.2. ECM frente a SCMW para distintos valores de K (ARRW) o de r_0 (ARRU).

6.3.3. Resultados para el clasificador del problema de Hart.

Los resultados obtenidos para el ECM y la SCMW mediante ARC, ARRU y ARRW se muestran en las tablas 6.7, 6.8 y 6.9, respectivamente. En este caso, se muestra el error de clasificación en % en lugar del ECM.

Tabla 6.7. Resultados con el algoritmo clásico.

Error clasificación (%)	SCMW
4.25 ± 3.53	60.264 ± 29.382

Tabla 6.8. Resultados del algoritmo con ruido (ARRU).

r_o	Error clasificación (%)	SCMW
0.1	4.53 ± 3.61	54.427 ± 32.466
0.2	3.40 ± 2.02	73.028 ± 26.971
0.3	3.20 ± 2.19	66.050 ± 23.908
0.4	4.13 ± 2.19	44.637 ± 22.123
0.5	4.08 ± 2.52	37.059 ± 20.692
0.6	4.75 ± 2.97	24.831 ± 18.322
0.7	6.10 ± 2.47	24.870 ± 16.476
0.8	7.93 ± 3.13	12.940 ± 12.433
0.9	9.63 ± 2.91	7.601 ± 7.339
1	9.25 ± 2.92	8.827 ± 8.788

Se van a comparar los resultados de ARRU para r_0=0.6 con los de ARRW para K=0.175. En ambos casos se obtiene un valor del error de clasificación similar al obtenido con ARC (4.25 con ARC, 4.75 con ARRU y 4,38 con ARRW) y valores inferiores de la SCMW (60.264 con ARC, 24.831 con ARRU y 8.923 con ARRW, es decir, con ARRW la SCMW es casi 3 veces menor que con ARRU y 7 veces menor que con ARC). Así pues, se obtienen menores valores de la SCMW cuando se emplea ARRW que los obtenidos mediante los otros dos algoritmos.

Tabla 6.9. Resultados del algoritmo robusto (ARRW).

K	Error clasificación (%)	SCMW
0.075	4.62 ± 1.94	31.884 ± 12.792
0.1	3.60 ± 1.53	27.150 ± 6.521
0.125	3.35 ± 2.02	17.299 ± 4.858
0.15	4.68 ± 2.16	12.450 ± 4.595
0.175	4.38 ± 2.14	8.923 ± 2.759
0.2	5.30 ± 2.55	5.548 ± 1.972
0.225	6.97 ± 2.85	2.939 ± 1.576
0.25	8.05 ± 3.28	2.065 ± 1.243
0.3	11.45 ± 1.40	0.862 ± 0.187

La Figura 6.3 muestra el error de clasificación (EC) como función de la sensibilidad cuadrática media. Puede apreciarse que con ARRW se obtienen menores valores de la SCMW y valores similares del EC respecto de los obtenidos con ARRU. Con ARRU se pueden obtener valores pequeños de la SCM a costa de peores prestaciones en cuanto a error de clasificación.

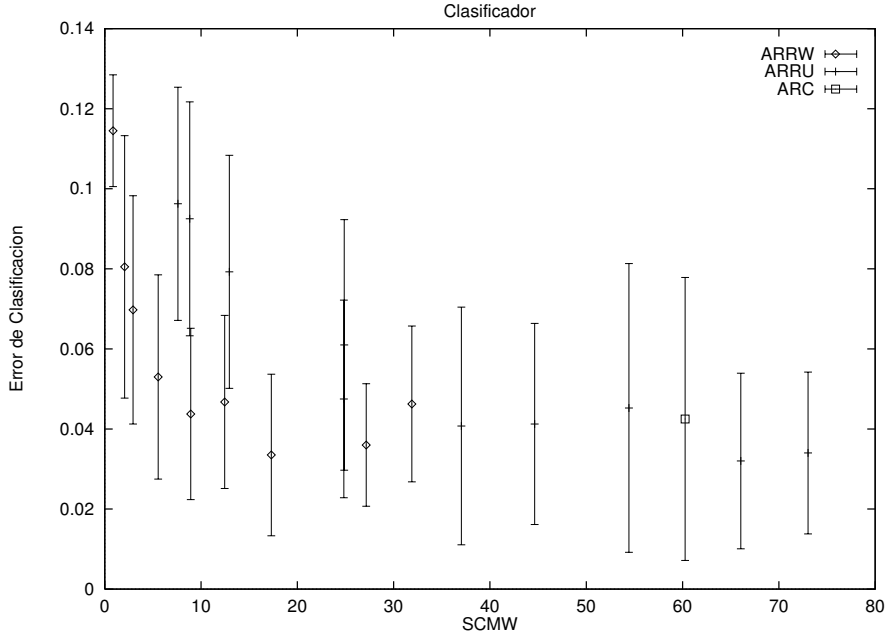

Figura 6.3. Error de clasificación frente a SCMW para distintos valores de K (ARRW) o de r_0 (ARRU).

Como conclusión de esta serie de experimentos se deduce que ARRU constituye un algoritmo eficaz para incrementar la tolerancia a desviaciones, si bien ARRW se muestra claramente superior pues se obtiene un incremento de tolerancia a fallos (una sensibilidad cuadrática inferior) más acusado que cuando se utiliza ARRU (o ARC). En cualquier caso, las prestaciones en cuanto a aprendizaje son del mismo rango que las obtenidas con ARC, e incluso, dependiendo del valor de r_0 (en el caso de ARRU) o de K (en el caso de ARRW), pueden ser mejores.

6.4. ESTUDIO DE LA CAPACIDAD DE GENERALIZACIÓN USANDO ARRW

En esta sección se realiza un estudio de la capacidad de generalización obtenida tras el entrenamiento del MLP cuando se usa ARRW en lugar de ARC. En [EDW94] se prueba que el algoritmo ARRU aumenta la capacidad de generalización del MLP, y en la sección anterior hemos probado que ARRW presenta mejores prestaciones en cuanto a tolerancia.

La capacidad de generalización de un MLP es un factor importante a la hora de comparar sus prestaciones. Aunque un MLP presente menor ECM de entrenamiento que otro, puede presentar una menor capacidad de generalización. Ello se debe al efecto de sobreentrenamiento (*overfitting*) que significa que cuando un MLP se entrena más de la cuenta, el MLP se especializa en los patrones de entrenamiento y pierde la capacidad de generalización, de forma que aunque reduce el ECM de entrenamiento, el ECM de test no disminuye, e incluso puede aumentar.

En el Capítulo 5 hemos encontrado una relación entre la SCM a ruido en las entradas (SCMN) y la capacidad de generalización. La SCMN evalúa la pendiente de las curvas de error cuadrático respecto de las entradas, de forma que un menor valor de esta magnitud implica un aprendizaje más homogéneo para cualquier patrón de entrada, es decir, el MLP produce errores de salida similares para cualquier patrón. Así mismo, hemos probado que al minimizar la SCMN se fuerza a que la curva de error sea más suave de forma que aumenta la capacidad de generalización.

También se ha visto en el capítulo anterior que las expresiones de la SCM a desviaciones en los pesos (SCMW) y la SCM a ruido en las entradas (SCMN) son similares (en particular, para las neuronas de la capa oculta y el modelo de desviación multiplicativo son idénticas), por lo que la minimización de una de ellas puede producir una reducción de la otra y viceversa. Es decir, al entrenar con ARRW además de hacer el MLP más tolerante a fallos en sus pesos se mejora su capacidad de generalización.

Para medir la capacidad de generalización se hace uso a menudo del llamado factor de *pérdida de generalización* (PG) [PRE94]. El valor de PG se calcula durante el entrenamiento del MLP

haciendo uso de un conjunto de patrones, llamado *conjunto de validación* , distinto del conjunto de entrenamiento.

El factor PG se define de la siguiente manera:

$$PG(t) = 100 \cdot \left(\frac{ECM_{va}(t)}{ECM_{\min}(t)} - 1 \right) \qquad (6.1)$$

donde $ECM_{va}(t)$ es el error cuadrático medio del conjunto de validación medido tras la época de entrenamiento t y $ECM_{min}(t)$ es el mínimo obtenido para ese error en épocas anteriores. Puede notarse que cuanto mayor sea este factor, mayor es el sobreentrenamiento y por tanto la capacidad de generalización es menor: en principio, conforme la red va aprendiendo, el ECM del conjunto de entrenamiento va disminuyendo y si va generalizando bien, también disminuye el ECM de validación. Sin embargo, cuando la red empieza a sobreentrenarse se especializa en los patrones de entrenamiento y por tanto, aunque el ECM de entrenamiento disminuye, el ECM de validación aumenta, y es este aumento el que trata de medir este factor.

A continuación vamos a comprobar si efectivamente el uso de ARRW como algoritmo de aprendizaje para aumentar la tolerancia a fallos también produce una mejora de la capacidad de generalización. Para ello mediremos el PG y también la SCMN.

En los experimentos realizados se han utilizado los ejemplos del aproximador del seno, el predictor de la serie temporal y el clasificador del problema de Hart. Se ha utilizado la misma estructura descrita en secciones anteriores, aunque se ha doblado el número de épocas de entrenamiento (4000 para el aproximador, 2000 para el predictor y 4000 para el clasificador). El factor PG se ha evaluado cada 10 épocas de entrenamiento usando como conjunto de validación, el conjunto de patrones de test, que es distinto del de entrenamiento. Para cada caso, los resultados se promedian sobre 40 pruebas tanto en el caso de ARC como en el caso de ARRW para el modelo de desviación multiplicativo.

6.4.1 Aproximador del seno.

Las Tabla 6.10 presenta los resultados obtenidos con ARC y ARRW. Para ARRW se ha fijado el valor de la constante K del factor de robustez a 0.7 y se muestran los resultados medios junto con las desviaciones típicas muestrales del ECM y la SCMW obtenidos al finalizar el aprendizaje sobre el conjunto de entrenamiento; el ECM, la SCMW y la SCMN evaluados en el punto mínimo de ECM sobre el conjunto de validación; así como del factor PG obtenido al finalizar

el aprendizaje del MLP. El número de épocas se ha fijado en 4000.

Los resultados muestran de nuevo que se obtienen mejores prestaciones en cuanto a la tolerancia a fallos y la capacidad de generaliación cuando se usa ARRW, siendo similares las prestaciones en cuanto a ECM respecto de las obtenidas con ARC. Con ARC se obtiene una media del factor PG igual a 27.703, unas 5 veces superior a la obtenida con ARRW (5.764), la SCMN también es inferior en el caso de ARRW (0.48384) que con ARC (0.42985). Ambas medidas indican que efectivamente se obtienen mejoras en cuanto a capacidad de generalización, además de la tolerancia a fallos cuando se usa ARRW como algoritmo de entrenamiento.

Tabla 6.10. Resultados para el aproximador del seno.

	ARC		ARRW	
	Media	Desv. típica	Media	Desv. típica
ECM entrenam.	0.000659	0.000317	0.00047	0.000161
SCMW entrenam.	4.811	0.859	1.467	0.097
ECM validación	0.000388	0.000238	0.000368	0.000122
SCMW validación	4.282	1.297	1.107	0.144
SCMN validación	0.48384	0.06568	0.42985	0.03105
PG	27.703	21.14	5.764	7.279

6.4.2 Predictor de la serie temporal.

Las Tabla 6.11 muestra los resultados obtenidos. En el caso de ARRW se ha fijado el valor de la constante K del factor de robustez a 1.0 y el número de épocas se ha fijado en 2000.

El ECM obtenido con ambos algoritmos es similar mientras que la SCMW es menor en el caso de usar ARRW. Con ARRW se obtiene también un valor de PG (4.559) inferior en media que el obtenido con ARC (8.297), pero la desviación típica es grande. Respecto a la medida propuesta para la generalización, con ARRW y ARC se obtiene un resultado similar. Ello indica que si bien se obtiene una cierta mejora respecto a generalización comparando el algoritmo ARRW con ARC, esta mejora es pequeña.

Tabla 6.11. Resultados para el predictor de la serie temporal.

	ARC		ARRW	
	Media	Desv. típica	Media	Desv. típica
ECM entrenam.	0.000102	0.00002	0.000103	0.000009
SCMW entrenam.	0.637	0.359	0.23	0.016
ECM validación	0.00008	0.000005	0.000082	0.000004
SCMW validación	0.62	0.34	0.232	0.017
SCMN validación	0.33601	0.00633	0.32852	0.00506
PG	8.297	20.888	4.559	6.182

De cualquier manera, el valor de PG obtenido es generalmente pequeño (inferior a 6 en aproximadamente el 75% de los casos) lo cual parece indicar que para este problema en concreto, con la estructura de red, patrones y parámetros escogidos, es poco probable que se vea afectado por un sobreentrenamiento, pero en los pocos casos en que se produce este fenómeno, el efecto es superior cuando se usa ARC que cuando se usa ARRW.

6.4.3 Clasificador del problema de Hart

En la Tabla 6.13 se muestran los resultados. Para ARRW se ha fijado el valor de la constante K del factor de robustez a 0.0175 y el número de épocas se ha fijado en 4000. A pesar de ser un clasificador se muestra el ECM.

Tabla 6.13. Resultados para el clasificador del problema de Hart.

	ARC		ARRW	
	Media	Desv. típica	Media	Desv. típica
ECM entrenam.	0.008643	0.015876	0.041478	0.003559
SCMW entrenam.	29.18	12.237	3.3	0.356
ECM validación	0.010333	0.014627	0.039053	0.002451
SCMW validación	61.508	20.181	3.449	0.359
SCMN validación	30.97502	10.50857	2.13039	0.31959
PG	290.683	225.507	20.777	10.478

El valor obtenido para el factor PG con ambos algoritmos es en este caso claramente distinto. Con ARC se obtiene un PG medio igual a 290.683 mientras que con ARRW se obtiene 20.777. Respecto a la SCMN estas diferencias son también grandes, con ARC la SCMN es 30.97502 mientras que con ARRW el valor medio de la SCMN es igual a 2.13039. En este caso, se tiene un ejemplo típico en el que usando ARC se obtienen mejores prestaciones en cuanto a ECM pero el riesgo de sobreentrenamiento y, por tanto de deficiencias en cuanto a generalización, son mayores. Hay que notar también en este caso que, a pesar de obtener un ECM mejor con ARC, sin embargo el error de clasificación es similar y que además, por ser la SCMW obtenida mucho menor (del orden de 20 veces) en el caso de ARRW, ARRW proporciona configuraciones de pesos más robustas y que presentan una mayor capacidad de generalización con un error de clasificación similar al obtenido con ARC.

De esta forma hemos confirmado que la capacidad de generalización es mejor cuando se usa ARRW, de aquí se deduce que podemos considerar este algoritmo como una técnica de regularización [GIR95], tal como se ha comentado en 6.2. También se ha comprobado que el uso de la SCMN (SCM a ruido en las entradas) como medida de la capacidad de generalización es apropiado ya que varía como el factor PG. Sin embargo, el factor PG es una medida heurística mientras que la SCMN ha sido deducida matemáticamente.

6.5. PRUEBAS CON EJEMPLOS DE PROBEN1

Los problemas utilizados hasta ahora para estudiar el comportamiento de los algoritmos presentados si bien se basan en ejemplos ampliamente conocidos y utilizados por diversos autores, se pueden considerar ejemplos de laboratorio. Por ello, a continuación se aplicará el algoritmo ARRW con el modelo multipicativo a problemas reales utilizados como *benchmarks* y que se encuentran reunidos en el conjunto PROBEN1(disponible en INTERNET en la dirección *http://wwwipd.ira.uka.de/~prechelt/announce/proben1.html*).

PROBEN1 [PRE94] consiste en un conjunto de problemas utilizados habitualmente como *benchmarks* con objeto de disponer de un conjunto normalizado de patrones de entrenamiento, test y validación que permita la comparación de distintas implementaciones de redes neuronales artificiales, así como de estrategias de entrenamiento.

Los ejemplos contemplados en PROBEN1 son problemas reales, cuyos datos han sido reunidos por diversas instituciones. En general, en cualquier problema de PROBEN1 se proporciona un conjunto de patrones, de los cuales se especifica cuáles deben ser usados como conjunto de entrenamiento, cuáles como conjunto de validación y cuáles como conjunto de test. El conjunto

de entrenamiento se usa para actualizar los pesos durante el entrenamiento; el de validación también es usado durante el entrenamiento, pero para evaluar el sobreentrenamiento (Apartado 6.4) y de esta manera poder seleccionar el conjunto de pesos óptimo. Por último, el conjunto de test se utiliza para estimar los resultados que se obtendrían cuando al MLP se le presentan patrones distintos de los usados para el aprendizaje.

Los tres conjuntos de patrones son disjuntos, es decir, los patrones de cada conjunto son distintos. En [PRE94] se menciona que según qué patrones se escojan para cada conjunto, los resultados pueden diferir, por ello, PROBEN1 proporciona tres ordenaciones distintas de tales conjuntos. En cada ordenación los patrones pertenecientes a cada conjunto se hayan prefijados.

Para comparar ARRW y ARC se han escogido tres aplicaciones de PROBEN1. Con cada ejemplo, se han aplicado los algoritmos mencionados a cada una de las tres ordenaciones de patrones. En cada caso se han realizado 40 entrenamientos para obtener valores medios. Los ejemplos seleccionados son los siguientes:

- Cáncer (*cancer*): diagnosis de cáncer de mama. Clasifica si un tumor es benigno o maligno basado en características celulares obtenidas mediante examen microscópico (uniformidad del tamaño de las células, forma de las células, adhesión, etc). Dispone de 9 variables de entrada, 2 salidas y 699 muestras.

- Edificio (*building*): predicción del consumo de energía en un edificio. Se predice el consumo horario de electricidad, agua caliente y agua fría, basándose en el día, hora, temperatura exterior, humedad exterior, radiación solar y velocidad del viento. El número de variables es 14, se proporcionan 3 salidas y se proporcionan 4208 muestras.

- Diabetes (*diabetes*): diagnosis de la diabetes. Se basa en datos personales (edad, número de embarazos) y los resultados de un examen médico (presión, glucosa, etc) y trata de decidir si un individuo tiene diabetes o no. Hay 8 variables de entrada, 2 salidas y se dispone de 768 muestras.

6.5.1. Clasificador del cáncer de mama

Los datos para este problema fueron recogidos por el Dr. William H. Wolberg de la Universidad de Winsconsin.

La estructura escogida para aplicar ARC y ARRW al MLP es la de una red con 9 neuronas de entrada, 5 en la capa oculta y 2 neuronas de salida. Se escoge como válida la salida ganadora.

Durante el entrenamiento se han realizado 1000 iteraciones (épocas) y los parámetros de aprendizaje α (factor de aprendizaje) y β (momento) se han fijado a 0.9 y 0, respectivamente. Las neuronas poseen entrada de sesgo.

Con ARRW se ha utilizado el modelo multiplicativo. De forma similar a como se hizo en el Capítulo 5, y a como se utiliza en el algoritmo ARRU, el factor de robustez γ se ha hecho decrecer dinámicamente durante el transcurso del entrenamiento partiendo de un valor inicial de 0.4 hasta un valor final de 0.01 para asegurar la convergencia.

Tal y como se ha mencionado antes, se han realizado 40 pruebas con cada algoritmo para cada una de las 3 ordenaciones proporcionadas de los conjuntos de entrenamiento, validación y test. En cualquier caso el número de patrones de entrenamiento es de 350, el de patrones de validación es de 175 y el conjunto de test cuenta con 174 patrones. Las 3 ordenaciones se referirán como cancer1, cancer2 y cancer3.

La Tabla 6.14 resume los resultados obtenidos con el algoritmo clásico, mientras que los resultados obtenidos con ARRW se muestran en la Tabla 6.15. En dichas tablas se presenta el error cuadrático medio (ECM) y la sensibilidad cuadrática media a desviaciones en los pesos (SCMW) para los conjuntos de entrenamiento, validación y test. También se muestra el error de clasificación obtenido sobre el conjunto de test, la SCM a ruido en las entradas sobre el conjunto de test (SCMN) y el factor PG, que mide la pérdida de generalización según muestra (6.1).

Tabla 6.14. Resultados con ARC para el clasificador del cáncer.

	cancer1		cancer2		cancer3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0	0.0034	0.011	0.0013	0	0.002
SCMW entrenamiento	0.344	0.1885	0.094	0.0363	0.233	0.2413
ECM validación	0.02	0.0011	0.017	0.001	0.029	0.0006
SCMW validación	0.169	0.2218	0.035	0.0072	0.048	0.0263
ECM test	0.01	0.0036	0.032	0.0019	0.024	0.0023
SCMW test	0.178	0.2151	0.1021	0.0237	0.1085	0.0445
Error clasificación %	2.237	0.5008	1.8678	0.2489	3.3764	0.1901
SCMN test	0.072	0.064	0.054	0.011	0.058	0.032
PG	66.47	34.514	67.007	10.896	75.95	27.293

Tabla 6.15. Resultados con ARRW para el clasificador del cáncer.

	cancer1		cancer2		cancer3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0.02	0.001	0.018	0.001	0.016	0.0012
SCM entrenamiento	0.06	0.0098	0.059	0.0063	0.087	0.0113
ECM validación	0.02	0	0.018	0	0.029	0.0004
SCM validación	0.06	0.0122	0.034	0.0041	0.037	0.0104
ECM test	0.01	0.001	0.036	0.001	0.024	0.0009
SCM test	0.06	0.0137	0.067	0.0105	0.066	0.0181
Error clasificación %	2.888	0.1567	2.2126	0.2052	3.4483	0
SCMN test	0.040	0.009	0.052	0.006	0.046	0.009
PG	5.098	4.4762	14.12	6.8838	9.4898	6.4279

El ECM y la SCMW de entrenamiento se han evaluado sobre los pesos obtenidos tras la última época del aprendizaje mientras que los valores referidos a validación y test se han calculado usando el conjunto de pesos que proporciona menor ECM con el conjunto de validación. El ECM de validación se ha evaluado cada 10 épocas. En cualquier caso se muestran los valores medios sobre las 40 pruebas, así como la correspondiente desviación típica muestral

Comparando los resultados mostrados en las tablas 6.14 y 6.15 se obtiene que:

- Las prestaciones en cuanto a error de clasificación y ECM son similares.
- La SCMW obtenida con ARRW es inferior a la obtenida con ARC.
- El factor PG y la SCMN obtenidos con ARRW son inferiores a los obtenidos con ARC.
- Las desviaciones típicas para cualquier factor, son generalmente bastante inferiores cuando se usa ARRW.

De estos hechos se deduce que usando ARRW como algoritmo de entrenamiento se reduce el riesgo de sobreentrenamiento obteniéndose configuraciones de pesos más robustas, manteniendo prestaciones respecto de aprendizaje similares a ARC. Por otro lado, el que las desviaciones típicas sean menores parece indicar que cuando se usa ARRW, se hace una selección más restrictiva de los mínimos locales donde la solución puede quedar estancada, de forma que sólo son válidos aquellos que presentan mayor insensibilidad a desviaciones en los valores de los pesos.

Respecto a generalización, el factor PG es claramente inferior cuando se usa ARRW. Hay que matizar que la SCMN se ha evaluado sobre el conjunto de pesos que proporcionaba un menor valor de validación el cual, por tanto, apenas es susceptible de presentar sobreentrenamiento, pero aún así, la SCMN es inferior en el caso de usar ARRW.

6.5.2. Predictor del consumo energético en un edificio

Se ha escogido una red con 14 neuronas de entrada, 4 en la capa oculta y 3 neuronas de salida. Las salidas predicen el consumo de electricidad, agua caliente y agua fría en un edificio a partir de las variables de entrada. Las neuronas tienen entrada de sesgo y se han utilizado 1000 épocas para el aprendizaje fijando los parámetros de α y β a 0.9 y 0, respectivamente. El factor de robustez γ en el caso de ARRW se ha hecho decrecer dinámicamente durante el transcurso del entrenamiento partiendo de un valor inicial de 0.4 hasta un valor final de 0.01 para asegurar la convergencia.

El número de patrones de entrenamiento es de 2104 y tanto el de patrones de validación como el de patrones de test es de 1052. Las 3 ordenaciones se referirán como edificio1, edificio2 y edificio3.

La Tabla 6.16 resume los resultados obtenidos con el algoritmo clásico, mientras que los resultados obtenidos con ARRW se muestran en la Tabla 6.17. Las magnitudes presentadas son similares a las mencionadas en 6.5.3, salvo que en este caso no se muestra el error de clasificación, obviamente.

En este caso se obtienen resultados que son similares en cuanto a capacidad de aprendizaje, se puede observar que usando tanto ARC como ARRW el ECM obtenido es prácticamente el mismo. Sin embargo, la SCMW se reduce drásticamente cuando se utiliza ARRW (del orden de 8 veces menor que la obtenida con ARC). El factor PG es menor en el caso de ARRW, salvo para la prueba *edificio1*, en la cual se obtienen valores similares para dicho factor. En cualquier caso, este factor es pequeño lo que probablemente se deba a que el MLP no termina de aprender bien la función que debe implementar y por lo tanto apenas se produce sobreentrenamiento. Sin embargo la SCMN es aproximadamente la mitad en el caso de ARRW lo que indica que una mayor homogeneidad del error cuadrático respecto de los vectores de entrada.

Tabla 6.16. Resultados con ARC para el predictor del consumo energético.

	edificio1		edificio2		edificio3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0.06	0.0016	0.1819	0.0021	0.185	0.0019
SCMW entrenamiento	1.135	0.7195	0.6103	0.1993	0.6796	0.2457
ECM validación	0.08	0.003	0.2012	0.0022	0.1905	0.002
SCMW validación	0.877	0.343	0.3738	0.1382	0.036	0.1556
ECM test	0.09	0.0033	0.1954	0.0024	0.192	0.0018
SCMW test	0.956	0.3803	0.3803	0.1412	0.382	0.1837
SCMN test	0.242	0.036	0.126	0.048	0.114	0.040
PG	4.166	2.2067	2.5585	1.2842	2.0517	1.211

Tabla 6.17. Resultados con ARRW para el predictor del consumo energético.

	edificio1		edificio2		edificio3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0.07	0.001	0.1854	0.0019	0.1891	0.0023
SCMW entrenamiento	0.325	0.0188	0.1055	0.0137	0.1091	0.0147
ECM validación	0.08	0.001	0.2019	0.001	0.1914	0.0005
SCMW validación	0.189	0.0307	0.082	0.0158	0.085	0.0193
ECM test	0.08	0.001	0.1955	0.001	0.1918	0.0009
SCMW test	0.191	0.0335	0.081	0.0164	0.085	0.0196
SCMN test	0.096	0.016	0.050	0.013	0.036	0.006
PG	4.697	1.7905	1.4559	1.3707	1.2003	1.3364

6.5.3. Clasificador de la diabetes

Como estructura del clasificador se ha utilizado un MLP con 8 neuronas de entrada, 16 en la capa oculta y 2 neuronas de salida. El diagnóstico viene dado por la neurona de salida ganadora.

El número de épocas y valor de parámetros de aprendizaje son los mismos que en los dos ejemplos anteriores.

El conjunto de entrenamiento cuenta con 384 patrones mientras que los de validación y test cuentan con 192 patrones cada uno. Las 3 ordenaciones de estos conjuntos proporcionadas por PROBEN1 se referirán como diabetes1, diabetes2 y diabetes3.

Los resultados obtenidos con ARC se muestran en la Tabla 6.18 mientras que los correspondientes a ARRW se muestran en la Tabla 6.19.

Tabla 6.18. Resultados con ARC para el clasificador de la diabetes.

	diabetes1		diabetes2		diabetes3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0.08	0.0076	0.075	0.0092	0.089	0.0081
SCMW entrenamiento	8.552	2.654	8.7118	2.7043	5.9466	1.7303
ECM validación	0.154	0.0013	0.1639	0.0021	0.1743	0.0018
SCMW validación	0.516	0.1266	0.517	0.1646	0.7007	0.3413
ECM test	0.155	0.0028	0.1857	0.0087	0.156	0.0214
SCMW test	0.532	0.1192	0.5623	0.1743	0.7481	0.2711
Error clasificación %	22.83	1.152	27.291	1.693	22.199	1.243
SCMN test	0.972	0.121	1.187	0.190	1.451	0.245
PG	35.95	5.655	37.237	7.713	25.242	7.25

Tabla 6.19. Resultados con ARRW para el clasificador de la diabetes.

	diabetes1		diabetes2		diabetes3	
	Media	Des.Tip.	Media	Des.Tip.	Media	Des.Tip.
ECM entrenamiento	0.141	0.0064	0.1297	0.0091	0.1334	0.0077
SCMW entrenamiento	0.229	0.0385	0.2372	0.0424	0.2392	0.039
ECM validación	0.153	0.0071	0.1614	0.001	0.1708	0.0063
SCMW validación	0.174	0.0289	0.1695	0.034	0.1275	0.0316
ECM test	0.155	0.0013	0.1913	0.0037	0.154	0.0022
SCMW test	0.184	0.0303	0.187	0.0395	0.1392	0.0355
Error clasificación %	22.04	0.66	28.469	0.69	22.133	0.652
SCMN test	0.763	0.101	0.952	0.170	0.761	0.165
PG	5.322	4.939	6.7747	5.4885	6.4813	4.5308

A partir de los resultados mostrados en las tablas anteriores se deduce que el error de clasificación obtenido con ambos algoritmos es muy similar, así como el ECM de validación y test. La SCMW obtenida con ARC es claramente superior (de 3 a 4 veces) a la que se obtiene con ARRW por lo que las configuraciones obtenidas con este último algoritmo son más robustas a desviaciones en los pesos. Por último, se puede apreciar que el factor de pérdida de generalización PG proporciona un valor inferior con ARRW, lo que indica que los efectos de un sobreentrenamiento son menores en este caso (la SCMN también es inferior con ARRW). Se puede apreciar también, al igual que en 6.5.1, que las desviaciones típicas son menores en el caso de ARRW, lo que parece indicar que las soluciones posibles cuando se usa ARRW implican una mayor uniformidad que las obtenidas con ARC.

De estos experimentos se deduce que el uso de ARRW es beneficioso para incrementar la tolerancia a fallos y la capacidad de generalización en MLPs.

6.6. CONCLUSIONES

En primer lugar hemos deducido que la SCMW evalúa la pendiente de la curvatura de la superficie de error respecto del valor de los pesos, de forma que puede considerarse una medida de la saliencia. Así, un menor valor de la SCMW indica pendientes más suaves del error cuadrático medio y, por tanto, perturbaciones en el valor de un peso producen cambios más suaves en las prestaciones del MLP respecto de error de salida. También se ha obtenido que, en el caso de perturbaciones multiplicativas, la SCMW depende también de la magnitud de los pesos, por lo que es más adecuada el uso de ésta como medida de la tolerancia a fallos que la saliencia.

Se ha realizado un estudio comparativo del algoritmo propuesto ARRW con otro algoritmo basado en la perturbación con ruido durante el entrenamiento y al que nos hemos referido como ARRU. A partir de los análisis realizados se desprende que ambos algoritmos reducen la sensibilidad cuadrática media a desviaciones en los pesos, manteniendo las prestaciones de aprendizaje del algoritmo clásico ARC, sin embargo, se obtienen soluciones más tolerantes a fallos cuando se usa ARRW. En concreto, se puede concluir que ARRU es un caso particular de ARRW pues se puede ver el efecto de este último como una perturbación de los pesos modulada por la SCM.

También se ha estudiado la capacidad de generalización que muestran los MLPs entrenados con ARRW respecto de los entrenados con ARC deduciéndose que cuando se usa el primero como algoritmo de entrenamiento, se reduce la pérdida de generalización (PG) así como la SCM a ruido en las entradas. Para ello, se han realizado diversos tests donde se ha medido PG y la SCMN siendo éstos menores en los casos donde se usó ARRW, indicando que el riesgo de sobreentrenamiento es menor y que el error de salida cometido para cada patrón de entrada presenta mayor uniformidad, por lo que se evita la especialización del MLP en sólo algunos patrones.

Por último se ha aplicado ARRW a problemas reales pertenecientes al paquete PROBEN1. Los resultados han confirmado las conclusiones obtenidas a lo largo de esta memoria de tesis: con ARRW se reduce la SCMW (SCM a desviaciones en los pesos) por lo que las salidas son más estables frente a desviaciones en los pesos, la capacidad de generalización es mayor y se mantienen las mismas prestaciones de aprendizaje respecto de los algoritmos de retropropagación clásicamente utilizados.

CAPITULO 7

Conclusiones y principales aportaciones

En este trabajo se ha realizado un estudio de la tolerancia a desviaciones de los perceptrones multicapa y del incremento de la misma mediante un algoritmo de aprendizaje aquí propuesto. Las principales aportaciones y conclusiones obtenidas quedan resumidas en los siguientes puntos:

1).- Se han obtenido las expresiones que permiten evaluar la sensibilidad estadística a desviaciones en los pesos. La sensibilidad estadística indica cómo se degrada la salida de un perceptrón multicapa cuando los pesos se modifican. Estas expresiones se han obtenido para dos modelos de desviación, aditivo y multiplicativo, y pueden evaluarse localmente para cada neurona de la red.

2).- Se ha obtenido una relación explícita entre la sensibilidad estadística y el error cuadrático medio sujeto a perturbaciones mediante una magnitud llamada sensibilidad cuadrática media (SCM), que se calcula a partir de las sensibilidades estadísticas de las neuronas de salida promediadas sobre un conjunto de patrones.

3).- De la relación entre la sensibilidad cuadrática media y el error cuadrático medio se ha deducido, y posteriormente comprobado, que un valor menor de la sensibilidad cuadrática media a desviaciones en los pesos (SCMW) implica una mayor estabilidad de las prestaciones del perceptrón a desviaciones en los mismos, de manera que la SCMW puede utilizarse para evaluar cuantitativamente la robustez o tolerancia a faltas de un perceptrón.

4).- La SCMW permite también predecir el comportamiento del perceptrón cuando sus pesos son perturbados, de forma que se puede conocer como se degradarán las prestaciones en cuanto a error de salida cuando los valores de los pesos obtenidos tras el entrenamiento se ven afectados por desviaciones en sus valores, por ejemplo, al trasladarlos a una implementación física concreta cuyos componentes analógicos presentan un determinado valor conocido de tolerancia.

5).- A partir de la sensibilidad estadística media a desviaciones en los pesos (SSM) se ha desarrollado un nuevo algoritmo de entrenamiento llamado ARR. Dicho algoritmo se basa en la adición de un término en la regla de aprendizaje clásica, relacionado con la sensibilidad estadística media, de forma que mediante un algoritmo de descenso de

gradiente se puede realizar una minimización multiobjetivo tanto del error cuadrático de salida como de la sensibilidad estadística media de la red.

6).- El algoritmo ARR se ha comparado con el algoritmo clásico (ARC) proporcionando menores valores de sensibilidad estadística y al tiempo que se mantienen las prestaciones en cuanto a error de salida. Se ha realizado un análisis de la T de Student y otro de la varianza (ANOVA) para demostrar la significación estadística de estos resultados y quedando claro que es precisamente el término relacionado con la sensibilidad estadística que se introduce en la ley de aprendizaje el responsable de la reducción de la sensibilidad estadística y que, además, no afecta a las prestaciones de aprendizaje.

7).- En la misma línea que ARR, se ha desarrollado otro algoritmo llamado ARRW, que difiere de ARR en que se realiza la minimización simultanea del error cuadrático medio junto con la sensibilidad cuadrática media, en vez de la sensibilidad estadística de salida. Los resultados indican que, por medio de este algoritmo, se obtienen configuraciones de pesos más robustas que con un algoritmo de entrenamiento clásico, siendo menos costoso computacionalmente que ARR y proporcionando una tolerancia a fallos más homogénea.

8).- El algoritmo ARRW se ha comparado con el algoritmo clásico y con otro propuesto en la literatura y basado en la introducción de ruido durante el entrenamiento para aumentar la tolerancia a faltas, que hemos llamado ARRU. Como ejemplos se han utilizado diversos problemas ampliamente conocidos y usados frecuentemente como benchmarks. Los resultados muestran que con ARRW:

- La sensibilidad cuadrática media es menor por lo que el perceptrón es más robusto a desviaciones en los pesos.

- Las prestaciones en cuanto a error de salida son similares.

- La capacidad de generalización aumenta.

9).- Se ha mostrado que ARRW puede considerarse una técnica de regularización explícitamente especificada.

10).- Se ha estudiado también el efecto de ruido en las entradas del perceptrón. Para este caso, se ha obtenido el valor de la sensibilidad cuadrática media a ruido en las entradas considerando los modelos de perturbación aditiva y multiplicativa. Por medio de la sensibilidad cuadrática a ruido en las entradas se puede estimar a priori los efectos de la

adición de ruido en los patrones de entrada y cómo se degrada la salida del perceptrón. Se ha mostrado también una relación explícita entre la sensibilidad cuadrática media a ruido en las entradas y la capacidad de generalización, proponiéndose por tanto, la utilización de la primera como medida cuantitativa de tal capacidad de generalización.

11).- Se ha desarrollado otro algoritmo llamado ARRN, el cual, en la misma línea que ARRW, minimiza la sensibilidad cuadrática a ruido en las entradas junto con el error de salida, de forma que se obtienen configuraciones de pesos más inmunes al ruido y con mayor capacidad de generalización que, sin embargo, presentan las mismas prestaciones en cuanto a aprendizaje que el algoritmo clásico.

12).- A partir de las expresiones obtenidas se pueden justificar matemáticamente algunas de las conjeturas presentadas en trabajos anteriores, a saber:

- La relación existente entre la tolerancia a fallos y la capacidad de generalización.

- La introducción de ruido en los pesos durante el entrenamiento para incrementar la tolerancia a desviaciones en los mismos.

- La introducción de ruido en las entradas durante el entrenamiento para incrementar la capacidad de generalización.

- La influencia de la magnitud de los pesos en la tolerancia a fallos del MLP.

El trabajo realizado ha abierto nuevas perspectivas que se van a explorar en trabajos futuros. Concretamente, algunas de las tareas que se pretenden acometer a medio plazo son las siguientes:

1).- Realizar un estudio cuantitativo de la distribución del aprendizaje entre las neuronas cuando se utilizan distintos algoritmos de entrenamiento. Para ello, se hará uso de la posibilidad de evaluar la sensibilidad estadística localmente en cada neurona, que se obtuvo en la presente memoria.

2).- La sensibilidad cuadrática a desviaciones en los pesos y la sensibilidad cuadrática a ruido en las entradas pueden sumarse para estudiar la acción simultanea de ambos efectos. De esta forma, los algoritmos ARRW y ARRN pueden combinarse para obtener perceptrones más inmunes al ruido y más estables frente a desviaciones en los pesos. Se analizarán las prestaciones de su uso conjunto.

3).- Desarrollar nuevos algoritmos de entrenamiento con menor coste computacional para reducir la sensibilidad cuadrática media.

4).- Particularizar las expresiones y algoritmos para otros modelos de desviación y, en particular, para faltas de anclaje y otras medidas de faltas propuestas en la literatura (stuck-open, stuck-off, ...).

5).- Extender el estudio a otras arquitecturas de red, como es el caso de las redes de funciones de base radial (RBFs).

Bibliografía

[ALI94] C. Alippi, V. Piuri, M. Sami, “Sensitivity to Errors in Artificial Neural Networks: a Behavioral Approach”, in *Proc. IEEE Int. Symp. On Circuits & Systems*, pp. 459-462. May 1994.

[AND89] J.R. Anderson, “A Theory of the Origins of Human Knowledge”. *Artificial Intelligence*, vol 40, pp. 313-351, 1989.

[ANG91] M. Anguita, A. Prieto, F.J. Pelayo, J. Ortega, A. Díaz, "CMOS Implementation of a Cellular Neural Network with Dynamically Alterable Cloning Templates". *Lecture Notes in Computer Science*, vol. 540, pp. 260-267, Springer-Verlag, 1991.

[ANG95] M. Anguita,F.J. Pelayo, F.J. Fernandez, A. Prieto, "A Low-Power Analog Implementation of Cellular Neural Networks". *Lecture Notes in Computer Science*, Vol.930, pp.898-905, Springer Verlag, 1995.

[ANG96] M. Anguita, F.J. Pelayo, E. Ros-Vidal, D. Palomar, A. Prieto, "VLSI Implementation of CNNs for Image processing and Vision Tasks: Single and Multiple Chip Approaches", invited talk in: *Proceedings of the 4th IEEE Int. Workshop on Cellular Neural Networks and Applications*, IEEE Computer Soc. Press, June 24-26, 1996.

[ANZ89] Anza Plus. *Users Guide and Neurosoftware Documents. HNC Neurosoftware*. HNC Inc.. Jun 1989.

[BAP94] P. Bapat, M.V. Hedge, M. Naraghi-Pour, “Fault Tolerant Radial Basis Function Networks”. *World Congress on Neural Networks*, San Diego, CA, vol. 3, pp. 463-469, June 1994.

[BEL89] L.A. Belfore, B.W. Johnson, “The Fault-Tolerance of Neural Networks”. Neural Networks, vol. 1, no.1, pp. 24-41. Jan 1989.

[BEN95] M. Benitez, J.Ortega, I. Requena., "Asynchronously Parallel Boltzmann Machines Mapped onto Distributed-memory Multiprocessors". *Lecture Notes in Computer Science*, Vol.930, pp.744-751, Springer Verlag, 1995.

[BER93A] J.L. Bernier, A. Lloris, J. Ortega, A. Prieto.: "Técnicas para el test de circuitos analógicos". *Libro Homenaje a Safwan Al Khouri Ibrahim*, pp.95-106, Universidad de Granada, ISBN 84-604-6334-6, 1993.

[BER93B] J.L. Bernier, J. Ortega, A. Lloris, A. Prieto,"Diagnosis de circuitos analógicos

mediante signaturas basadas en variaciones de parámetros de alto nivel". VIII Congreso de Diseño de Circuitos Integrados. Málaga, 9-11 Nov. 1993, pp.205-210.

[BER95A] J.L. Bernier,J. Ortega, A. Prieto, A. Lloris, "Selección de Puntos de Test en Circuitos Analógicos", *X Congreso de Diseño de Circuitos Integrados*, pp. 36-41. Zaragoza, 15-17 Nov. 1995.

[BER95B] J.L. Bernier,J.J. Merelo, J. Ortega, A. Prieto, "Test Pattern Generation for Analog Circuits Using Neural Networks and Evolutive Algorithms". *Lecture Notes in Computer Science*, Vol.930, pp.838-844, Springer Verlag, 1995.

[BER96] J.L. Bernier; C. Ilia Herráiz, J.J. Merelo, S. Olmeda, A. Prieto,. "Solving Mastermind Using GAs and Simulated Annealing: A Case of Dynamic Constraint Optimization". In *Proc. of Parallel Problem Solving from Nature IV (PPSN IV)*. Springer-Verlag, 1996.

[BER97] J.L. Bernier, J. Ortega, A. Prieto, "A Modified Backpropagation Algorithm to Tolerate Weight Errors". *Lecture Notes in Computer Science*, vol. 1240, pp. 763-771. Jun 1997.

[BIS95] C. Bishop, "Training with Noise is Equivalent to Tikhonov Regularization". *Neural Computation*, vol. 7, no. 1, pp. 108-116. 1995.

[BJO95] M.H. Bjorndal, A. Caprara, P.I. Cowling et al, "*Some Thoughts on Combinatorial Optimisation*". European Journal of Operational Research, vol. 83, pp. 253-270, Elsevier Science 1995.

[BOS97] K.M. Bossley, "Regularisation Theory applied to Neurofuzzy Modelling". *Technical Report ISIS-TR3*, Dept. Of Electronics and Computer Science, University of Southampton, 1997.

[BOX89] G.E.P. Box, *Estadística para Investigadores. Introducción al Diseño de Experimentos, Análisis de Datos y Construcción de Modelos*. Ed. Reverté, 1989.

[CAÑ91] A. Cañas, J. Ortega, F.J. Fernández, A. Prieto, F.J. Pelayo, "An Approach to Isolated Word Recognition using Multilayer Perceptrons". *Lecture Notes in Computer Science,* vol. 540. pp. 340-347. Springer-Verlag, 1991.

[CAS90] G. Casella, R.L. Berger, *Statistical Inference*. Duxbury Press, 1990.

[CAS91] F. Castillo, P. Amengual, J. Cabestany, "A Sensibility Study of the Backpropagation Algorithm". *Artificial Neural Networks*, pp. 665-670, Elsevier Science Publishers B.V., 1991.

[CHI94] C. Chiu, K. Mehrotra, C.K. Mohan, S. Ranka, "Modifyng Training Algorithms for Improved Fault Tolerance", in *Proc. IEEE Int. Conf. on Neural Networks*, vol. 1, pp. 333-338. Jun 1994.

[CHO92] J.Y. Choi, C. Choi, "Sensitivity Analysis of Multilayer Perceptron with Differenciable Activation Functions". *IEEE Trans. on Neural Networks*, vol. 3, no. 1, pp. 101-107. Jan 1992.

[CHU88] L.O. Chua, L. Yang, "Cellular Neural Networks: Theory". *IEEE Trans. on Circuits and Systems*, vol. 35, no. 10, pp. 1257-1272. Oct. 1988.

[EDW93] P.J. Edwards, A.F. Murray, "Analogue Synaptic Noise - Implications and Learning Improvements". Int. Journal of Neural Systems, vol. 4, pp.427-433, 1993.

[EDW95] P.J. Edwards, A.F. Murray, "Can Deterministic Penalty Terms Model the Effects of Synaptic Weight Noise on Network Fault-Tolerance?". Int. Journal of Neural Systems, vol.6, no. 4, pp. 401-416,1995.

[EDW96] P.J. Edwards, A.F. Murray, "Modelling Weight- and Input-Noise in MLP Learning", Proc. International Conference on Neural Networks, Washington D.C., vol. 1, pp.78-83, June 1996.

[EDW97] P.J. Edwards, A.F. Murray, "Penalty Terms for Fault Tolerance". *Proc. IEEE ICNN'97*, vol. 2, pp. 943-947, June 1997.

[EDW98A] P.J. Edwards, A.F. Murray, "Weight Saliency Regularisation in Augmented Networks". European Symposium on Artificial Neural Networks (ESSAN), D-Facto, pp. 261-266, 1998.

[EDW98B] P.J. Edwards, A.F. Murray, "Fault Tolerance via Weight Noise in Analog VLSI Implementations of MLP's -- A Case Stdy with EPSILON". IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no.9, pp. 1255-1262. Sep. 1998.

[EDW98C] P.J. Edwards, A.F. Murray, "Towards Optimally Distributed Computation". *Neural Computation*, vol. 10, pp. 997-1015, 1998.

[ELS94] H. Elsimary, S. Mashali, A. Darwish, S. Shasheen, "Performance Evaluation of a Novel Fault Tolerance Trainig Algorithm", in *Proc. Int. Conf. On Electronics, Circuits & Systems*, pp. 566-570. Dec 1994.

[FEL91] D.B.I. Feltham, W. Maly, "Physically Realistic Fault Models for Analog CMOS Neural Networks". *IEEE Journal of Solid-State Circuits*, vol. 26, no. 9. Sep. 1991.

[FER93] J.J.Fernandez-Rodríguez, A.Cañas, E.Roca, F.J.Pelayo, J.Fernandez-Mena, A.Prieto: "Application of Artificial neural Networks to Chest Image Classifications". *Lecture Notes in Computer Science,* vol. 686, Springer Verlag, 1993.

[FIS36] R.A. Fisher, "The Comparison of Samples with Possibly Unequal Variances", *Annals of Eugenics*, vol. 9, pp. 174-180, 1936.

[FRY91] R.C. Frye, E.A. Rietman, C.C. Wong, "Back-Propagation Learning and Nonidealities in Analog Neural Network Hardware". *IEEE Trans. on Neural Networks*, vol. 2, no.1, pp. 110-117. Jan 1991.

[FUK80] K. Fukushima, "Neurocognitron: A Self-Organizing Neral Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". *Biological Cybernetics*, vol 36, pp. 139-202, 1980.

[GIL98] C. Gil, J. Ortega, J.L. Bernier, M.G. Montoya, "Parallel Test Pattern Generator using Circuit Partitioning in a Shared-memory Multiprocessor". *Lecture Notes on Computer Science*, Springer Verlag, June 1998.

[GIR95] F. Girosi, M. Jones, T. Poggio, "Regularization Theory and Neural Networks Architectures". *Neural Computation*, vol. 7, pp. 219-269, 1995.

[GRA94] A. Grace, "Optimization Toolbox", in *Matlab User´s Guide*. The MathWorks Inc.. Jan 1994.

[GRO72] S. Grossberg, "A Neural Theory of Punishment and Avoidance. Part I: Qualitative Theory". *Mathematical Biosciences*, vol. 15, pp. 39-47, 1972.

[HAR68] P.E. Hart, "The Condensed Nearest Neighbor Rule". IEEE Trans. Information Theory, vol. 125, 1968.

[HAY94] S. Haykin, *Introduction to the Theory of Neural Computation*. McMillan, 1994.

[HEC90] R. Hecht-Nielsen, *Neurocomputing*. Addison-Wesley, 1990.

[HOP82] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". *Proc. Natl. Acad. Sci*., vol.79, pp. 2554-2558, 1982.

[HOR89] K.M. Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Networks are Universal Approximators". *Neural Networks*, 2, 359-366, 1989.

[JAI96] A.K. Jain, J. Mao, K.M. Mohiuddin, "Artificial Neurals Networks: A Tutorial". *IEEE Computer*, pp. 31-44. Mar. 1996.

[KAN95] E.R. Kandel, J.H. Schwartz, T.M. Jessell, *Essentials of Neural Science and Behavior*. Prentice Hall, 1995.

[KIL68] W.L. Kilmer, W.S. MacCulloch, J. Blum, "Some Mechanisms for a Theory of the Reticular Formation". *System Theory and Biology*. Springer-Verlag, New York, 1968.

[KOH87] T. Kohonen, "State of the Art in Neural Computing". *IEEE Proc. of Int. Conference on Neural Networks*, pp. 179, 190, Jun 1987.

[KOH88] T. Kohonen, "An Introduction to Neural Computing". *Neural Networks*, vol. 1, no.1, pp. 3-16, 1988.

[KOH90] T. Kohonen, "The Self-Organizing Map". *Proc. of the IEEE*, vol. 78, no.9, pp. 1464-1480, 1990.

[LIN94] C. Lin, I. Wu, "Maximizing Fault Tolerance in Multilayer Neural Networks", in *Proc. IEEE Int. Conf. On Neural Networks*, vol. 1, pp.419-424. Jun 1994.

[LIP87] R.P. Lippmann, "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine*, pp. 4-22. Apr 1987.

[MAO93] J. Mao, A.K. Jain, "Regularization Techniques in Artificial Neural Networks". *World Congress on Neural Networks*, pp. 75-79, 1993.

[MAR76] D. Marr, T. Poggio, "Cooperative Computation to Understanding Neural Circuitry". *Science*, vol 194, pp. 283-287.

[MAR82] D. Marr, *Vision*. Freeman, New York, 1982.

[MAR90] A. Maren, Harston, R. Pap, *Handbook of Neural Computing Applications*. Academic Press, 1990.

[MCC43] W.S. McCulloch, W. Pitts, "A Logical Calculus of the Ideas Inmanent in Nervous Activity". *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[MER91] J.J. Merelo, M.A. Andrade, C. Ureña, A. Prieto, F. Morán, "Application of Vector Quantization Algorithms to Protein Classification and Secundary Structure Computation". *Lecture Notes in Computer Science*, vol. 540, Springer-Verlag. pp. 415-421, 1991.

[MER94] J.J. Merelo, M.A. Andrade, A. Prieto, F. Moran, "Proteinotopic Feature Maps", *Neurocomputing*, Vol.6, No.4, pp.443-454, Aug 1994.

[MIN69] M.L. Minsky, S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[MIR71] J. Mira, *Modelos Cibernéticos de Aprendizaje*, Tesis Doctoral, Fac. De Ciencias, Universidad de Madrid, 1971.

[MIR93] J. Mira, A.E. Delgado, J.R. Alvarez, A.P. de Madrid, M. Santos, "Towards More Realistic Self Contained Models of Neurons: High-order, Recurrence and Local Learning". *Lecture Notes in Computer Science*, vol. 686, pp. 55-62, Springer-Verlag, 1993.

[MIR95] J. Mira, A.E. Delgado, J.G. Boticario, F.J. Díez, *Aspectos Básicos de la Inteligencia Artificial*. Ed. Sanz y Torres, 1995.

[MON91] D.C. Montgomery, *Diseño y Análisis de Experimentos*. Grupo Editorial Iberoamérica, 1991.

[MOR68] R. Moreno-Diaz, W.S. McCulloch, "Circuilarities in Nets and the Concept of Functional Matrices". In *Byocibernetics of the CNS*, Little-Brown, MA, pp. 145-150, 1968.

[MOR72] R. Moreno Diaz, J. Mira, A. Roy Yarza, "Realización de redes no determinísticas". *Revista de Automática*, vol. 11, pp. 5-14, 1972.

[MUR93] A.F. Murray, P.J. Edwards, "Synaptic Weight Noise During Multilayer Perceptron Training: Fault Tolerance and Training Improvements". *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp.722-725. Jul 1993.

[MUR94] A.F. Murray, P.J. Edwards, "Synaptic Weight Noise During MLP Training: Enhanced Performance and Fault Tolerance Resulting from Synaptic Weight Noise Training", IEEE Trans. on Neural Networks, vol. 5, no. 5, pp. 792-802. Sep 1994.

[MUS92] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris, D.M. Hummels. "On the Training of Radial Basis Function Classifiers". *Neural Networks*, vol. 5, no. 4, pp. 595-603, 1992.

[NEU45] J. von Neumann, *First Draft of a Report on the EDVAC*. Moore School, Univ. of Pennsylvania, 1945.

[NIJ89] J.A.G. Nijhuis, L. Spaanenburg, "Fault Tolerance of Neural Associative Memories". *IEE Proceedings*, vol. 136, no. 5, pp. 389-394. Sep 1989.

[ORT91] J. Ortega, A. Lloris, A. Prieto, F.J. Pelayo, "Using Reed- Muller coefficients to synthesise optimal prediction modules for concurrent testing". *Electronics Letters*, Vol.27, No.14, pp.1243-1245. July, 1991.

[ORT93A] J. Ortega, A. Prieto, A. Lloris, F.J. Pelayo, "A Generalized Hopfield Neural Network for concurrent testing". *IEEE Trans. on Computers*, Vol.42, No.8, pp. August 1993.

[ORT93B] Ortega,J.; Lloris,A.; Prieto,A.; Pelayo,F.J.: "Test- pattern generation based on Reed-Muller coefficients". *IEEE Transactions on Computer*. Vol.42, No.8, pp. August 1993.

[PAR94] L. Parrilla, J. Ortega, A. Lloris, "Using Simulated Annealing in the Minimization of AND-EXOR Functions". *Electronics letters*, vol. 30, no. 22, pp.1838-1839, Oct. 1994.

[PAT96] D.W. Paterson, *Artificial Neural Networks. Theory and Applications*. Prentice Hall, 1996.

[PHA95] D.S. Phatak, I. Koren, "Complete and Partial Fault Tolerance of Feedforward Neural Nets". *IEEE Trans. On Neural Networks*, vol.6, no. 2, pp. 446-456. Mar 1995.

[PRE94] L. Prechelt, "PROBEN1 -- A Set of Neural Network Benchmark Problems and Benchmarking Rules". Technical Report 21/94. Universität Karlsruhe, Germany. September, 1994.

[PRI91] A. Prieto (Ed.), "Artificial Neural Networks". *Lecture Notes in Computer Science*, vol. 540, pp. V-VII. Springer Verlag, 1991.

[PRI96] A. Prieto, "Una Visión Actual sobre la Neurocomputación". *Conferencia Inaugural del Curso 1996-97 en la Escuela Técina Superior de Ingeniería Informática*. Servicio de publicaciones de la Universidad de Granada, 1996.

[REY91] L.M. Reyneri, E. Filippi, "An Analysis of the Performance of Silicon Implementations of Backpropagation Algorithms for Artificial Neural Networks". *IEEE Trans on Computers*, vol. 40, no. 12, pp. 1380-1389. Dec 1991.

[RUM85] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning Internal Representations by Error Propagation". *ICS Report 8506*, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, Sept. 1985.

[RUM86] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning Internal Representations by Error Propagation". *Parallel Distributed Processing: Explorations in the Microestructure of Cognicion Vol 1: Foundations*. MIT Press, Cambridge, MA, pp. 318-362, 1986.

[SAN96] E. Sánchez, M. Tomassini (Edts.), *Towards Evolvable Hardware*. Lecture

Notes in Computer Science, vol. 1062, 1996.

[SAN97] E. Sánchez, D. Mange, M. Sipper, M. Tomassini, A. Pérez-Uribe, "Phylogeny, Ontogrny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware". In *Proc. of the First Intern. Conf. On Evolvable Systems: from Biology to Hardware* (ICES96), Springer Verlag, vol. 1259, pp.35-54. 1997.

[SEG94] B.E. Segee, M.J. Carter, "Comparative Fault Tolerance of Parallel Distribuited Processing Networks". *IEEE Trans. On Computers*, vol. 43, no. 11, pp. 1323-1329. Nov 1994.

[SEQ90] C.H. Séquin, R.D. Clay, "Fault Tolerance Training in Artificial Neural Networks", in Proc. Int. Joint Conf. On Neural Networks, pp. I-703-708. Jun 1990.

[SHA96] Y. Shang, B.W. Wah, "Global Optimization for Neural Network Training". Computer, pp. 45-54. Mar 1996.

[STE90] M. Stevenson, R. Winter, B. Widrow, "Sensitivity of Neural Networks to Weight Errors". *IEEE Trans. On Neural Networks*, vol.1, no. 1, pp. 71-80. Mar 1990.

[STE97] M. Stevenson, "The Effects of Limited-Precision Weights on the Threshold Adaline". *IEEE Trans. on Neural Networks,* vol. 8, no. 3, pp. 549-552. May 1997.

[SUD94] T. Sudkamp, R. Hammell, "Interpolation, Completion and Learning Fuzzy Rules". *IEEE Trans. on Systems, Man & Cybernetics*, vol. 24, no.2, pp. 332-342. Feb 1994.

[TOH96] Y. Tohma, Y. Koyanagi, "Fault-Tolerant Design of Neural Networks for Solving Optimization Problems". IEEE Trans. on Computers, vol. 45, no. 12, pp.1450-1455. Dec 1996.

[WAN94] L. Wang, *Adaptive Fuzzy Systems and Control. Design and Stability Analysis*. Englewood Cliffs, Prentice Hall, 1994.

[WID90] B. Widrow, M.A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", in *Proc. of the IEEE*, vol. 78, no. 9, pp. 1415-1441, Sep 1990.

[XIE92] Y. Xie, M.A. Jabri, "Analysis of the Effects of Quantization in Multilayer Neural Networks Using a Statistical Model". *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 334-338. Mar 1992.

[YOS96] T. Yoshino, I. Nagayama, N. Akamatsu, “The Learning Capability of Cyclic Activation BP for Two-Spirals Problem”, in *Proceedings of IIZUKA ‘96*, pp. 684-687, 1996.

[ZAD95] L.A. Zadeh, M. Jamshidi, “Soft Computing and Fuzzy Logic: Toward High MIQ Systems”. Intern. J. Intelligent Automation Soft Computing, pp. 1-30, vol.1, no.1, 1995.