UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

DISEÑO AUTOMÁTICO DE REDES DE NEURONAS ARTIFICIALES PARA LA PREDICCIÓN DE SERIES TEMPORALES

Autor:
Juan Peralta Donate

Directores:
Araceli Sanchis de Miguel
Germán Gutiérrez Sánchez

DEPARTAMENTO DE INFORMÁTICA

Leganés, Enero 2012

II

Departamento de Informática

Escuela Politécnica Superior
Universidad Carlos III de Madrid

Diseño Automático de Redes de Neuronas Artificiales para la Predicción de Series Temporales

AUTOR: Juan Peralta Donate
DIRECTORES: Araceli Sanchis de Miguel
Germán Gutiérrez Sánchez

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Carlos III de Madrid, el día ..... de ..... de 2011.

Presidente: D. .....

Vocal: D. .....

Vocal: D. .....

Vocal: D. .....

Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2011 en .....

Calificación: .....

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

A mi familia

"El medio para hacer cambiar la opinión es el afecto, no la ira."

(Dalai Lama)

"Hasta el más grande de los viajes empieza dando un paso."

(B. Franklin)

Agradecimientos

Quiero dedicar este trabajo a mis padres José y Juana, quienes me trajeron a este mundo y educaron, a quienes tengo siempre como referentes, y a quienes llevo en el corazón cada día. Mil gracias por todo.

A mis hermanas y hermano, Antonia, Luz y Pepe, a los que tanto quiero, por lo mucho que me han ayudado en todos los momentos de mi vida. A mis sobrinos, Quique, Javi, Luis, Jose María y Elvira porque me alegran cada momento que estoy con ellos.

A mis cuñados y cuñada, Ángel, Clemente y Elena, por haberme enseñado tanto para poder ser una mejor persona.

A mis tutores, Araceli y Germán, por haberme dado la oportunidad de llevar a cabo esta tesis doctoral.

A mis *otros* tutores con quienes he tenido la suerte de poder trabajar, Paulo, Martin, Lenka, Xiaodong, Ramón y Xin. Gracias por haberme aceptado en vuestro grupo de investigación y por haberme enseñado tanto.

A todos y cada uno de los integrantes del grupo CAOS; Jmaw, Raúl, José Antonio, Jorge, Bea, Javi, Andrés, Mari Paz, Agapito, Paula e Isaac. Gracias por vuestra ayuda y por hacerme el trabajo tan agradable. También al resto de compañeros de la universidad, de comedor y de cafés, gracias por hacerme pasar tan buenos momentos aquí.

A todos mis amigos que tanto se han interesado y tanta paciencia han tenido conmigo cuando les daba la lata sobre la tesis.

A Noemí, por haber tenido tanta paciencia conmigo y haberme apoyado tanto el tiempo que duró este trabajo y por estar ahí cada día. Gracias por los momentos que hemos pasado juntos y que espero se repitan por siempre.

A todos vosotros por vuestro apoyo, ¡muchas gracias!

Resumen

El ser humano ha avanzado mucho, tecnológicamente hablando, en el último siglo. La sed por descubrir e innovar no tiene límites y como no, aplicar dichas innovaciones para nuestro provecho y bienestar general. Uno de los campos de investigación a los que se ha aplicado dicha innovación es la predicción. Al hablar de predicción, lo primero que nos puede venir a la cabeza son temas no tan científicos como la astrología o la lectura de manos, pero en realidad, diversos métodos estadísticos y matemáticos pueden ayudar a proporcionar información sobre el futuro. Uno de los ejemplos más comunes es la predicción del tiempo meteorológico que podemos observar cada día en televisión.

Además de los métodos ya mencionados, en los últimos años ha proliferado el estudio de la predicción mediante técnicas de inteligencia computacional y dentro de las predicciones, aquellas que se ocupan de predecir series temporales. La predicción de series temporales consiste en llevar a cabo aproximaciones o estimaciones de qué valores tendrán los elementos futuros de una serie temporal partiendo de los valores de los elementos previos o ya conocidos. Como veremos en esta tesis doctoral, a lo largo de los años se han usado diferentes técnicas de inteligencia computacional con este propósito, aunque nosotros nos centraremos en las redes de neuronas artificiales. Plantearemos las ventajas y problemas que se pueden dar y nos centraremos en intentar solventar dichos problemas. Uno de los problemas clave que se plantea actualmente a la hora de aplicar redes de neuronas artificiales a cualquier dominio dado, es su correcto diseño. Estudiaremos pues las diferentes soluciones propuestas para el correcto diseño de las redes de neuronas artificiales, aunque terminaremos centrándonos en aquellas que hacen uso de la computación evolutiva. Este modelo de redes es el que se conoce como redes de neuronas artificiales evolutivas.

Esta tesis doctoral presenta tres enfoques diferentes para el modelado automático de redes de neuronas artificiales. Cada enfoque irá destinado a solventar cada uno de lo que nosotros consideramos los tres puntos o problemas claves existentes al diseñar una red de neuronas artificial. El primer enfoque consistirá en el tratamiento de los datos que son pasados como patrones a la red para que ésta aprenda y sea evaluada. El segundo enfoque se centrará en las diferentes técni-

cas evolutivas que pueden ser usadas, cómo obtener un fenotipo a partir de un genotipo (y viceversa) y cómo evaluar una red. El último enfoque que se estudiará, es el tipo de arquitectura de red que debe ser usada para la predicción de series temporales.

El objetivo final de esta tesis doctoral es llevar a cabo un sistema automático de diseño de redes de neuronas artificiales para solventar problemas de predicción de series temporales con la mayor exactitud posible y transparente al usuario, es decir, que este no tenga que ser un experto en la materia para poder hacer uso de él.

Abstract

The human being have progressed a lot, technologically speaking, in the last century. The thirst for discovery and innovation has no limits and of course, to apply these innovations to our benefit and general welfare. One of the research areas that have been applied to this innovation is the prediction. When we talk about prediction, the first thing that may come to our minds are not so scientific issues as astrology or hand reading, but in fact, several statistical and mathematical methods can help to provide information about the future. One of the most common examples is the weather forecasting that we can watch every day on television.

Besides the methods already commented, in recent years it has proliferated the prediction study using computational intelligence techniques and within these predictions, those consisting of time series forecasting. Time series forecasting consist of carrying out approximations or estimations about which values will have the future elements of a time series starting from the values of the previous already known elements. As we discuss in this PhD thesis, over the years it has been used different computational intelligence techniques for this purpose, although we will focus on artificial neural networks. We will present the advantages and problems that may appear and we will focus on trying to solve these problems. One of the key problems that currently arise in applying artificial neural networks to any given domain, is its correct design. We will study then the different solutions proposed for the proper design of artificial neural networks, although at the end, we will focus on those which use evolutionary computation. This network model is known as evolutionary artificial neural networks.

This PhD thesis presents three different approaches for the automatic design of artificial neural networks. Each approach will be dedicated to solve each of what we consider the three points or key problems in designing an artificial neural network. The first approach will consist of treating the data that are passed to the network as patterns to make it learn and be evaluated. The second approach will focus on the different evolutionary techniques that can be used, how to obtain a phenotype from a genotype (and vice versa) and how to evaluate a network. The last approach to be studied, is the type of network architecture to be used for time series forecasting.

The ultimate goal of this thesis is to implement an automatic system to design artificial neural networks to solve time series forecasting problems as accurately as possible and transparent to the user, i.e. that the user did not have to be an expert to make use of it.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El ser humano siempre ha querido saber que ocurrirá mañana, que le deparará el futuro. Existe pues una tendencia humana muy fuerte que consiste en explorar lo venidero.

Algunos desastres como las inundaciones o las enfermedades se producían sin aviso o razón aparente. La sociedad primitiva trataba de explicar dichos fenómenos naturales mediante un grupo de dioses y diosas que jugaban con ellos como si de niños caprichosos se trataran. Los hombres creían que no existía modo alguno de predecir los pasos a seguir por sus dioses, y la única esperanza era obtener su favor a través de regalos o conductas. Hoy en día, algunas personas siguen suscribiendo estas creencias tratando de firmar un pacto con la fortuna. Algunos llevan a cabo ciertas promesas para obtener a cambio una buena nota en una asignatura o para poder aprobar el examen de conducir.

Es por ello que, con el transcurso del tiempo, la sociedad ha llevado a cabo diferentes maneras de *"adivinar"* el futuro.

Algunos grupos intentaron escrutar los hechos mediante la magia o el contacto con lo sobrenatural. Para hacerlo, podían leer los augurios en las entrañas de los animales o en las hojas de té. En la antigua Roma, los generales utilizaban estos métodos para calcular sus probabilidades de éxito en la inminente batalla.

También tuvo un origen muy temprano la confianza en las estrellas como forma de predecir el futuro. La astrología, el estudio de la correlación entre lo sucedido en la Tierra y la posición y los movimientos de los astros, fue una ciencia esencial en la China, Grecia y Roma clásicas y en el Oriente Próximo islámico. Aunque la astrología y la astronomía siguieron caminos separados durante el siglo XVI, a finales del siglo XVII muchos europeos consultaban a los astrólogos para calcular el futuro de una boda o un síntoma de enfermedad. A pesar de que desde

hace muchos años los científicos han rechazado los principios de la astrología, en la actualidad, millones de personas aún creen en ella o la practican.

En esta línea, han existido diferentes visionarios a lo largo de la historia de la humanidad. Ya Michel de Nôtre-Dame o Miquèl de Nostradama, más conocido como Nostradamus [1], médico y consultor astrológico provenzal de origen judío en el siglo XVI, fue considerado uno de los más renombrados autores de profecías y eventos futuros.

Por otro lado, la sociedad se fue dando cuenta poco a poco de ciertos comportamientos regulares existentes en la naturaleza. Estas regularidades resultaban más obvias y observables en el movimiento de los cuerpos celestes a través del firmamento. Es por ello que la astronomía es una de las primeras ciencias conocidas. Se dotó de una firme base matemática, dada por Newton, hace ya más de 300 años y todavía hoy en día se hace uso de estas teorías, entre ellas la teoría de la gravedad, para poder predecir el movimiento de casi todos los cuerpos celestes.

Al igual que ocurría con los astros, se llegó a la conclusión de que otros fenómenos naturales también obedecen leyes científicas definidas. Esto llevó a la idea del determinismo científico, cuya primera defensa en público fue llevada a cabo por el científico francés Laplace.

Por lo tanto, la teoría de que el estado del universo en un instante dado determina el estado en cualquier otro momento ha sido uno de los principales dogmas de la ciencia desde los tiempos de Laplace. Esta teoría defiende que podemos predecir el futuro, al menos en principio. Pero en cuanto a la práctica se refiere, nuestra posible capacidad de predicción del futuro se encuentra limitada por las ecuaciones que lo rigen y más concretamente por su complejidad, además del hecho de que éstas normalmente presentan una propiedad denominada caos

Sobre este tema, Einstein no compartía la idea de una aparente aleatoriedad en la naturaleza [2]. Su opinión se resumía en su famosa frase *"Dios no juega a los dados"*. Él iba más allá y defendía que la incertidumbre era tan solo provisional y que existía una realidad subyacente en la que las partículas tendrían posiciones y velocidades bien definidas y se comportarían de acuerdo con leyes deterministas, en consonancia con Laplace. Esta realidad podría ser conocida por Dios, pero la naturaleza cuántica de la luz nos impediría verla, excepto tenuemente a través de un cristal.

Hoy en día los investigadores e ingenieros hacen uso de diferentes técnicas y estudios matemáticos, estadísticos e informáticos para llevar a cabo la compleja tarea de predecir, si no el futuro, al menos parte de él.

1.1. La Ciencia Aplicada a la Predicción de Series Temporales

Parémonos a pensar como de importante sería para nuestro día a día conocer nuestro futuro. De hecho, ya hoy en día hacemos uso de dichas predicciones para nuestra vida cotidiana. ¿Quién no ha comprobado alguna vez la predicción del tiempo meteorológico antes de salir de viaje?, ¿que empresa no ha hecho una estimación de sus gastos y ganancias para el año fiscal que se avecina?. Hoy en día, incluso los médicos pueden llegar a predecir el riesgo que tenemos de padecer un cáncer con una sola gota de nuestra sangre.

Todas estas *"predicciones"* son posibles gracias a métodos estadísticos como *Holt-Winters* [3, 4] (en los sesenta) o la metodología ARIMA [5, 6] (en los setenta). Más recientemente, diversos métodos de *Inteligencia Computacional* (IC) han sido aplicados a la predicción de series temporales con diferentes resultados. Entre dichos métodos, cabe destacar los *Sistemas Inmunes Artificiales* (SIA) [7], *Máquinas de Soporte Vectorial* (MSV) [8], *Redes de Neuronas Artificiales* (RNA) [9], *Técnicas Difusas* (TD) [10, 11] o incluso sistemas híbridos combinando cualquiera de las anteriormente presentadas con *Computación Evolutiva* (CE) [12, 13].

Para poder llevar a cabo una predicción de valores futuros, es necesario datos o valores pasados. Una series temporal es una secuencia de datos medidos y ordenados de manera cronológica y uniforme. Esta tesis doctoral se ha enfocado en la predicción de series temporales mediante RNA [14]. Las RNA proporcionan una metodología que puede resolver problemas no lineales, que son difíciles de resolver por medio de técnicas tradicionales. A menudo las series temporales muestran variabilidad espacial y temporal, y sufren la no linealidad de los procesos físicos. Las RNA son modelos flexibles capaces de extraer la relación entre las entradas y las salidas de un proceso y no requieren un conocimiento *"a priori"*, son además capaces de llevar a cabo un modelado no lineal y a menudo con tolerancia ante el *"ruido"*. Estas propiedades de las RNA las hacen apropiadas para ser aplicadas al dominio de la predicción de series temporales. De hecho, las RNA han sido aplicadas para la predicción en el mundo real en repetidas ocasiones, en campos como la predicción de mercados [15], la meteorología [16] y la predicción del tráfico de redes [17].

Llegados a este punto, el problema relacionado con el uso de RNA es su diseño que debe ser adecuado para una serie temporal determinada [9]. Será necesario determinar el tipo de RNA que solventará la tarea de la predicción (Perceptrón Multicapa), el algoritmo de aprendizaje a usar (Retropropagación) y que arquitec-

tura concreta (topología y valores de los pesos de conexión) serán mejores para predecir una serie temporal específica. Por lo tanto, se tiene que establecer el valor para cada parámetro de la RNA [14] (número de nodos de entrada, número de neuronas de salida, número de capas ocultas, cuántas neuronas ocultas habrá por cada capa oculta, cuál será el patrón de conexiones, pesos de dichas conexiones, etc.).

A menudo, el proceso de diseñar el modelo de RNA correcto está basado en una heurística de prueba y error. Si un experto llevara a cabo un diseño manual de una RNA, diferentes topologías con diferentes factores de aprendizaje deberían ser entrenadas. Para cada una de ellas, se obtiene un error de entrenamiento y validación y aquella con la mejor capacidad de generalización será seleccionada para llevar a cabo la tarea correspondiente. Una alternativa mejor, consiste en usar un método automático de diseño de RNA, donde la CE juega un papel importante, en lo que se conoce como *Redes de Neuronas Artificiales Evolutivas* (RNAE) [18]. Este proceso automático consiste en una técnica de búsqueda evolutiva como puede ser un *Algoritmo Genético* (AG) [19], el cual hará uso de sus propiedades de exploración y explotación para encontrar una buena solución. Los parámetros de la RNA serán establecidos en el cromosoma y la capacidad de generalización de cada individuo será dada por la función de evaluación. Algunos retos en el diseño de RNA consisten en:

- Definir un buen modelo de RNA para llevar a cabo la tarea estipulada con un buen resultado.
- Conocer que datos son dados a la RNA para aprender y la manera en que estos datos se suministran en forma de patrones.
- Cuál es la mejor función de aprendizaje para un dominio concreto.
- Cómo codificar el modelo de la RNA dentro del cromosoma.
- Cuáles de las técnicas de CE es la mejor para llevar a cabo la búsqueda del modelo de RNA que se desea obtener.

Esta tesis doctoral propone tres temas claves diferentes o puntos de acción a tratar en el diseño de RNA para hacer frente a los retos previamente comentados. Cada uno de estos puntos irá destinado a un campo diferente, pudiendo dividir dichos campos entre tratamientos de datos, CE y arquitectura de la RNA.

El primero de los temas a tratar en esta tesis doctoral, se centra en definir exactamente qué datos de la serie temporal serán usados para la predicción, cómo

se tratarán dichos datos y cómo se presentaran a la RNA en forma de patrones de aprendizaje y validación.

El segundo tema clave tiene como finalidad definir cual de las diversas técnicas evolutivas existentes es más apropiada para llevar a cabo la búsqueda global del modelo deseado.

Por último, el tercer punto propuesto en esta tesis doctoral trata de definir la mejor de las arquitecturas posibles para las RNA candidatas durante el proceso de búsqueda local (i.e. entrenamiento de la RNA).

1.2. Objetivos

El objetivo general de esta tesis doctoral consiste en definir un marco que permita crear modelos de RNA de forma automática para predecir series temporales.

Para ello se implementará un sistema híbrido de diseño de Redes de Neuronas Artificiales mediante una búsqueda global de los parámetros de la RNA empleando técnicas de CE, además de una búsqueda local o entrenamiento de dichas RNA.

Se trata de ajustar mejor los parámetros de las RNA, pudiendo así obtener un mejor individuo final como solución al problema de la predicción de series temporales.

Dicho sistema será transparente al usuario y por lo tanto fácil de usar aunque éste no sea un experto en la materia.

Para lograr este objetivo general, se deben cumplir los objetivos específicos que se citan a continuación:

- Definición de un método que genere, a partir de los valores previamente normalizados, un conjunto de patrones. Dado que las RNA elegidas para esta tesis doctoral aprenden a partir de un conjunto de patrones y también serán evaluadas por los citados conjuntos (siempre disjuntos entre ellos), se deberá desarrollar un método capaz de generar dichos subconjuntos de patrones.

- Desarrollo de un método que a partir de una codificación previa de los parámetros más significativos e importantes de las RNA, lleve a cabo una búsqueda de orden global mediante técnicas de CE para intentar mejorar generación tras generación, los posibles candidatos a una buena solución.

- Aplicación de algoritmos de aprendizaje capaces de sintonizar con mayor precisión los pesos de las RNA expuestas como solución por la técnica de la CE previamente comentada, para afinar más si cabe la solución buscada.

- Definición de una función de evaluación lo más óptima posible acorde con el problema expuesto en esta tesis doctoral. Dado que dicha función será la encargada de valorar la capacidad de generalización de cada solución propuesta por el sistema, este punto puede ser considerado cómo muy importante.

- Desarrollo de un método que permita llevar a cabo el almacenamiento de las diferentes soluciones generadas. Toda solución generada deberá poder ser almacenada para su posterior estudio y utilización. Para ello, es necesario el desarrollo de un método capaz de crear y utilizar una *"biblioteca"* de posibles soluciones.

- Definición de un método, que a partir de las soluciones expuestas como finales por el sistema, sea capaz de, si no seleccionar la mejor de ellas, al menos obtener un único resultado a partir de un subconjunto de dichas soluciones finales.

- Realizar la experimentación necesaria para comprobar que los objetivos propuestos en los puntos anteriores se han llevado a cabo con éxito.

Aunque el diseño de RNA resulta útil en muchas y muy diversas aplicaciones, nos resulta difícil abordar todas estas aplicaciones y se ha optado para esta tesis doctoral por un dominio mas específico como es la predicción de series temporales. Sin embargo, dentro de la predicción de series temporales, no nos centraremos en un único dominio como puede ser el subgrupo de las financieras. Esta tesis doctoral trata de predecir con la mayor exactitud posible toda serie temporal con aplicaciones al mundo real, sin importar el dominio al que esta pertenezca.

1.3. Organización de la Memoria

Con la finalidad de dar una visión general de los temas que se tratan en esta tesis doctoral, en el capítulo 2 se revisa la literatura disponible sobre la predicción de series temporales y las diversas técnicas de IC que se han aplicado para ello. Dentro de estas técnicas, nos centraremos más en las RNA y principalmente en el diseño de éstas con técnicas evolutivas.

En el capítulo 3 se presenta una primera aproximación del sistema propuesto denominado *"Automatic Design of Artificial Neural Networks"* (ADANN) [13]. En este capítulo se explica en detalle la manera de proceder a la hora de solventar todos los temas comentados anteriormente. Estudiaremos en detalle el segundo tema clave de esta tesis doctoral, concerniente a las diferentes técnicas de CE por las que se ha optado a la hora de diseñar las RNA. Se propone además el tercero de los puntos a tratar en esta tesis doctoral, referente a las RNA en sí y más concretamente a las arquitecturas usadas.

En el capítulo 4 se lleva a cabo la experimentación y se muestran los resultados de lo estudiado en el capítulo anterior. Estudiaremos cuestiones tales como la normalización de los datos y la función de evaluación, común a todas las aproximaciones presentadas del sistema.

Estudiaremos el primer tema clave propuesto en esta tesis doctoral, el concerniente a los datos y como presentar estos a las RNA para intentar obtener mejores predicciones. También se realiza un estudio comparativo con diferentes métodos estadísticos, técnicas de IC y herramientas software de predicción de series temporales haciendo uso de un conjunto referenciado de series temporales.

Por último, en el capítulo 5 se detallan las conclusiones generales, las aportaciones de esta tesis doctoral y las líneas de trabajo futuras. Al final de este capítulo se muestran las publicaciones que se han llevado a cabo durante la realización de esta tesis doctoral.

Capítulo 2

Estado del Arte

Se denomina inteligencia artificial, a la rama de las ciencias de la computación dedicada al desarrollo de agentes racionales artificiales. Entendemos que un agente es capaz de recibir información de su entorno (recibir entradas), procesar esta información y actuar en este entorno (proporcionar salidas), y entendemos la racionalidad como la capacidad humana que nos permite pensar, evaluar y actuar.

Podemos distinguir varios procesos que nos permiten obtener resultados racionales:

- Llevar a cabo una respuesta predeterminada por cada entrada recibida (similar a los actos reflejos de los seres vivos).
- Pasar al estado requerido, en el conjunto de todos los estados posibles dados, según las acciones llevadas a cabo hasta el momento.
- Técnicas de búsqueda, entre ellas la CE y más concretamente los algoritmos genéticos (simulando el proceso de evolución de las cadenas de ADN).
- Aproximadores de funciones y más concretamente la redes de neuronas artificiales (intentan simular el funcionamiento físico del cerebro humano).
- Razonamiento lógico (simulando el pensamiento abstracto humano).

En este capítulo se presentan las investigaciones más relevantes relacionadas con los diversos temas involucrados en esta tesis doctoral. El apartado 2.1 trata sobre las series temporales, su análisis, predicción y la aplicación que estas predicciones tienen en los problemas del mundo real. Los distintos tipos de aprendizaje automático se resumen en el apartado 2.2. En el apartado 2.3 hablamos sobre una de las líneas más utilizadas de la inteligencia artificial, la de los algoritmos genéticos, para más tarde exponer diversas investigaciones sobre la predicción de series

temporales llevadas a cabo con algoritmos genéticos en el apartado 2.6. Es en la sección 2.7 donde explicamos en detalle las redes de neuronas artificiales, para más tarde en el apartado 2.8 mostrar los diferentes estudios que se han llevado a cabo sobre predicción de series temporales haciendo uso de redes de neuronas artificiales. En el apartado 2.9 nos introducimos en el estudio del diseño de la redes de neuronas artificiales mediante técnicas evolutivas o como son conocidas, redes de neuronas artificiales evolutivas para luego en la sección 2.10 explicar como se ha aplicado hasta la fecha esta técnica de la inteligencia artificial a la predicción de series temporales. Por último, hablamos en la sección 2.11 de la teoría de conjuntos, teoría que como veremos más adelante será de utilidad en esta tesis doctoral.

2.1. Series Temporales

En estadística, procesamiento de señales, y econometría, una serie temporal es descrita como una secuencia de datos, observaciones o valores, medidos en determinados momentos del tiempo, ordenados cronológicamente y, normalmente, espaciados entre sí de manera uniforme [20, 21].

Algunos ejemplos son los datos climáticos, las acciones de bolsa, o las series pluviométricas. Resulta complicado intentar buscar una rama de las ciencias en la que no aparezcan datos que puedan ser considerados como series temporales.

Esta misma definición puede ser extrapolada al ámbito industrial y en general a las ciencias sociales, dando lugar a que observaciones de una o varias de las características de una población se realicen ligadas al tiempo o se encuentren fechadas en instantes determinados del tiempo.

Así, por ejemplo, una de las características de una empresa que puede ser observada es su volumen de ventas y puede resultar interesante estudiar el comportamiento y evolución temporal de esa característica de la empresa. En este caso esa observación se realizará de forma repetida durante una sucesión de instantes de tiempo. Dicha observación repetida en el tiempo da lugar a una serie temporal.

En la figura 2.1 podemos observar un ejemplo de serie temporal. En dicho ejemplo, se ha tomado como medida el número anual de ventas realizadas por una empresa desde 1985 hasta 1994.

El análisis de las series temporales comprende diferentes métodos que sirven para interpretar este tipo de datos mediante la extracción de información representativa, tanto la que hace referencia a los orígenes o relaciones subyacentes como la que comprende la posibilidad de extrapolar y predecir su comportamiento futuro.

Año	Ventas
1985	3560
1986	3940
1987	3910
1988	3890
1989	4040
1990	4010
1991	4120
1992	3980
1993	4070
1994	4190

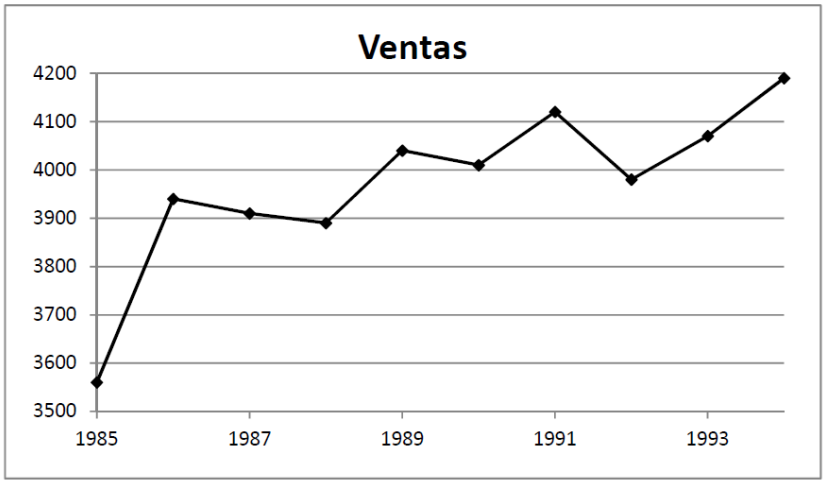


Figura 2.1: Ejemplo de serie temporal.

De hecho, uno de los trabajos más habituales con series de datos temporales es su análisis para su posterior predicción y pronóstico.

2.1.1. Predicción de Series Temporales

La predicción de series temporales ha sido y es uno de los dominios en los que más esfuerzo se ha dedicado en el campo de la predicción científica. Hoy en día son muchos los datos pasados, almacenados durante años, medidos en diferentes dominios que los científicos usan para poder estudiar que ocurrirá en un futuro próximo tales como datos económicos [22] e industriales [23]. El problema o tarea de predecir series temporales consiste en la obtención de la relación entre el valor del instante *"t"* y los valores conocidos previos a este (*t-1, t-2, . . . , t-k*). Esta relación viene dada mediante una función como se muestra en 2.1:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-k}) \tag{2.1}$$

donde $\hat{y}_t$ denota la predicción estimada dada por la función (f).

La predicción en series temporales es una línea de investigación fundamental en el campo de la estadística. El hecho de poder reproducir el comportamiento de un sistema dinámico a partir de medidas homogéneas (series temporales) de sus variables, posibilita la aplicación de los modelos de predicción basados en series temporales (en inglés *"time series based models"*) a una gran cantidad de campos del conocimiento. Estos modelos de predicción basados en series temporales com-

plementan la modelización física o, incluso, pueden sustituir a la modelización física cuando esta se hace demasiada compleja.

Para poder predecir una serie temporal, antes debe llevarse a cabo un análisis de dicha serie temporal. Para que este análisis pueda conducir a conclusiones acertadas no es suficiente con utilizar las técnicas apropiadas, sino que es imprescindible que los datos sean comparables, y para ello deben de ser homogéneos. Por ejemplo, si cambiamos la metodología de observación cada año, se cambian las definiciones y por lo tanto modificamos la población de referencia, dando como resultado una serie temporal definida por un conjunto de valores imposible de comparar al ser éstos muy heterogéneos. Esta homogeneidad puede perderse de forma natural con el transcurso del tiempo, entonces, cuando las series temporales son muy largas no existe garantía alguna de que los datos del inicio y del fin sean comparables.

Sin embargo, la necesidad de que las series temporales no sean de gran tamaño (i.e. series con muchos elementos), para que sus datos no pierdan la deseable homogeneidad, está en contradicción con el objetivo más básico de la estadística, que consiste en detectar regularidades en los fenómenos de masas.

La planificación, es necesario prever aquellos sucesos del futuro que pudieran ocurrir. Por otro lado, la previsión se basa en lo que ha ocurrido durante el pasado. Disponemos entonces de un tipo de inferencia estadística que se lleva a cabo sobre el futuro de alguna variable o compuesto de éstas, basándonos en sucesos pasados.

La metodología que normalmente es utilizada para estudiar las series temporales es sencilla de comprender y se basa fundamentalmente en descomponer las series temporales en diferentes partes [20, 21]: tendencia, variación cíclica, variación estacional o periódica, y otras fluctuaciones irregulares:

- **Tendencia.** Consiste en la dirección general que toma la variable durante el periodo de observación, es decir, el cambio que se da a largo plazo en la media de la serie. Una serie temporal sin tendencia es conocida como estática.

- **Variación cíclica.** Es la componente de la serie temporal que representa oscilaciones periódicas de amplitud superior a un año. Estas oscilaciones periódicas son irregulares y se presentan principalmente en los fenómenos económicos cuando se dan de forma alternativa etapas de prosperidad o de depresión.

- **Periodicidad.** Es una componente de la serie temporal que recoge oscilaciones periódicas de amplitud diferentes a un año, mes, semanas, etc. Estas

oscilaciones periódicas son también irregulares y se presentan en los fenómenos económicos cuando se dan de forma alternativa etapas de prosperidad o de depresión.

- **Fluctuaciones irregulares.** Después de haber extraído de la serie temporal las componentes anteriores, nos quedan unos valores residuales, que pueden o no ser aleatorios. Volvemos al principio, pues de nuevo nos interesa determinar si esa secuencia temporal de valores residuales puede o no corresponderse con un patrón puramente aleatorio.

Lo que se pretende con el modelado de una serie temporal, es tanto describir como predecir el comportamiento de un fenómeno que cambia en el tiempo. Una vez que conocemos las diferentes componentes de una serie temporal, estas pueden asociarse de forma aditiva o multiplicativa para obtener un modelo que nos ayude a predecir los valores futuros.

En esta tesis doctoral tenemos por lo tanto como principal objetivo el desarrollo de una herramienta automática basada en técnicas de inteligencia artificial, que lleve a cabo la predicción de series temporales haciendo prácticamente innecesaria la intervención de ningún experto. Dicha herramienta aprenderá a partir de la información contenida en los propios datos, es decir, los valores conocidos de la serie temporal. Además, no será necesario ningún estudio, procesamiento o descomposición previo de los datos como ocurre con otras técnicas, entre ellas la estadística como, acabamos de ver.

2.1.2. Aplicaciones de la Predicción de Series Temporales

La predicción de series temporales a corto plazo, también llamada identificación de sistemas por los expertos en automática, es una disciplina que encuentra cada día más aplicaciones en áreas de planificación, gestión, producción, mantenimiento y control de procesos industriales. Existen varios estudios que explican cómo la predicción de series temporales ha ayudado en campos tan dispares como las ciencias de la salud [24, 25]; economía [26], ya sea para el control monetario nacional [27, 22] o para controlar la inflación en la República de Colombia [28]; para la predicción de la curva de demanda de energía eléctrica [29, 23] (aunque es tan solo un ejemplo del amplio abanico de aplicaciones industriales que pueden ser tratadas con estos métodos); psicología [30]; medio ambiente [31] y el último pero no menos importante que debería ser también incluido en el campo de la economía, la predicción de los valores de bolsa (índice de mercado) [32] de las diferentes compañías que cotizan en ellas o índices de referencia (Dow-Jones, IBEX35, NASDAQ, etc.).

Por otro lado, muchos fenómenos como por ejemplo, los niveles de la marea [33], consumo de agua [34, 35], velocidad de la sangre en el torrente sanguíneo [36], mortalidad en terremotos [37], pueden modelarse con series temporales, lo cual hace el problema de la predicción de series temporales tan complejo como interesante a la hora de salvar vidas u obtener beneficios económicos.

2.2. Aprendizaje Automático

El aprendizaje automático es una rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas para permitir a las computadoras solventar problemas. Más concretamente, trata de generar programas que sean capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Por lo tanto, consiste en un proceso donde se induce el conocimiento. Algunas veces el campo de actuación del aprendizaje automático y el de la estadística están solapados, debido a que ambas disciplinas están basadas en el análisis de datos. Sin embargo, el aprendizaje automático está más centrado en el descubrimiento y evolución de los sistemas.

El aprendizaje automático tiene una amplia gama de aplicaciones que incluyen, detección de fraude en el uso de tarjetas de crédito [38], diagnósticos médicos [39], clasificación de secuencias de ADN [40], reconocimiento del habla y del lenguaje escrito [41], juegos y robótica [42], motores de búsqueda [43], análisis del mercado de valores [44], predicción de series temporales [13], etc. Existen diferentes tipos de aprendizaje automático, que se detallan a continuación:

2.2.1. Aprendizaje Supervisado

El aprendizaje supervisado [45] consiste en un tipo de aprendizaje automático en donde al algoritmo que se utiliza para obtener el modelo de la relación entre la entrada y la salidas del agente se le proporcionan una serie de ejemplos con sus correspondientes etiquetas o salida que indican a que clase pertenecen o el valor de salida en el caso de problemas de regresión. Es decir, que todos los ejemplos han sido clasificados *"a priori"*.

De esta forma en el proceso de aprendizaje, el algoritmo compara su salida actual con la etiqueta del ejemplo para luego realizar los cambios que sean necesarios. Un claro ejemplo de este aprendizaje es la manera que tienen las Redes de Neuronas Artificiales (RNA) [13] de trabajar, las cuales hacen uso de un con-

junto de patrones, conocidos a priori, con unas entradas y las salidas correspondientes que se deben obtener con dichas entradas. Para que la salida de la RNA se aproxime o iguale a la salida del patrón, la RNA irá modificando los pesos de sus conexiones durante el proceso de aprendizaje.

2.2.2. Aprendizaje no Supervisado

El aprendizaje no supervisado es un método de aprendizaje automático en el cual un modelo es ajustado a las observaciones dadas. Se distingue del aprendizaje supervisado debido a que no existe un conocimiento a priori. En el aprendizaje no supervisado no existe información referente a las salidas de los patrones que vamos introduciendo, sólo sus entradas. Ha de ser el modelo, por lo tanto, el que vaya adecuando sus pesos en función de la información interna que vaya recogiendo de las entradas. Este tipo de entrenamiento se utiliza principalmente para clasificar y diferenciar rasgos significativos de un conjunto de datos no clasificado a priori, dado que la red internamente intenta encontrar redundancias y rasgos significativos para agrupar los datos. Ejemplos de este tipo de aprendizaje son los algoritmos de agrupamiento o clustering entre los que se pueden señalar COBWEB [46], EM [47] y K-Medias [48].

2.3. Algoritmos Genéticos

Los algoritmos genéticos son una técnica de búsqueda basada en la teoría de la evolución de Charles Darwin. Dichos algoritmos han ganado popularidad en el área de las ciencias de la computación durante las últimas décadas. Esta técnica se basa en los principios de la selección natural y la supervivencia de los individuos más fuertes, postulados por Darwin [49]. Según esto, los individuos más idóneos de una población son los que sobreviven y tienen descendencia, al adaptarse más fácilmente a los cambios que se producen en el entorno. Esta descendencia puede tener ligeras modificaciones sobre sus predecesores de modo que aquellos cambios que den lugar a un descendiente mejor adaptado al medio, serán los que se *“conserven”*.

Hoy en día, es sabido que estos cambios se llevan a cabo en los genes (unidad básica de codificación de cada uno de los atributos de un ser vivo) de un individuo, y que los atributos más deseables (i.e. aquellos que permiten al individuo adaptarse mejor al medio) son transmitidos a sus vástagos mediante la reproducción sexual.

John Holland, basándose en la selección natural desarrolló esta técnica a finales de los años 60 que permite ser aplicada a las ciencias de la computación. El objetivo buscado era conseguir que las computadoras pudieran aprender por sí mismas. A esta técnica inventada por Holland se le pasó a denominar originalmente *"planes reproductivos"*, pero se popularizó con el nombre de *"algoritmo genético"* tras la publicación de su libro en 1975 [50].

Una definición completa de un AG es la propuesta por John Koza [51]:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo, usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud."

Por lo tanto, podemos considerar los AG como métodos que adaptan modelos que son usados para resolver problemas de búsqueda y optimización. Dichos AG se basan en el proceso genético llevado a cabo por los organismos vivos. Como todo ser vivo, dispondrán de generaciones, i.e. conjunto de individuos, que irán evolucionando con el tiempo. Los AG imitan el proceso evolutivo que se da en la naturaleza, siendo estos capaces de generar soluciones para solventar diferentes problemas del mundo real.

En nuestro caso, generamos RNA para predecir series temporales. Para que se dé una evolución de estas soluciones, obteniendo valores más óptimos del problema, antes debe llevarse a cabo una buena codificación de las mismas, al igual que una buena selección de la función de evaluación. La manera elegida para codificar cada individuo o RNA dentro de un cromosoma se detalla en la sección 3.6.1 y describimos las diferentes funciones de evaluación planteadas en la sección 3.3.

Un AG consiste entonces en un programa o función matemática que recibe como entradas un conjunto de posibles soluciones (i.e. individuos que forman una generación) y da como salida una nueva generación evolucionada con respecto a la dada como entrada. Para ello, los AG llevan a cabo un ciclo iterativo en el que toman a una generación de individuos y generan una nueva generación, reemplazando ésta a la antigua, un número de veces determinado por el diseño del propio AG.

Los principios básicos de los AG fueron establecidos por Holland, y están bien detallados en la literatura [52, 19]. En la naturaleza los individuos de una especie compiten entre ellos por la búsqueda de recursos como pueden ser comida, agua y

refugio, además de competir también estos individuos en la búsqueda de un compañero con el que poder reproducirse. Aquellos individuos que se adaptan mejor al medio y al final tiene más éxito al sobrevivir y encontrar compañeros para la reproducción, tienen también mayor probabilidad de procrear un gran número de descendientes. Esto significa que los genes de aquellos individuos que se adaptan mejor al medio se propagarán en futuras generaciones en un gran número de posibles individuos futuros. Al darse una combinación de las buenas características provenientes de diferentes progenitores, se puede obtener a veces descendientes denominados *"superindividuos"*, los cuales, tienen una mejor adaptación al medio que la que pudieran tener cualquiera de sus antepasados. De esta forma, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Los AG son análogos al comportamiento natural. Empezamos con una primera población de individuos, representando cada uno de ellos una posible solución a un problema dado. Cada individuo obtiene un valor que representa la eficacia que éste tiene como posible solución. Denominamos a este valor, valor de adecuación. Cuanto mejor se adapte un individuo al problema, mejor valor de adecuación se le asigna a éste, y tendrá más probabilidades de que sea seleccionado para llevar a cabo la reproducción, cruzando su material genético con otro individuo seleccionado de la misma manera. Esta *"reproducción"* dará lugar a nuevos descendientes, manteniendo éstos algunas de las características de sus progenitores. En el polo opuesto se encuentran los individuos con menor grado de adaptación al medio. Dichos individuos tendrán una menor probabilidad para ser seleccionados y poder así reproducirse, siendo por lo tanto más complicado el propagar su material genético entre generaciones venideras.

Una vez que la reproducción se ha llevado a cabo entre los individuos de una generación para obtener una nueva población de posibles soluciones, esta nueva reemplazará a la anterior y se podrá comprobar que al menos tiene las mismas buenas características que la generación anterior y posiblemente las haya mejorado durante el proceso evolutivo (que da lugar a un buen valor de adecuación). Así, conforme vayan pasando las generaciones, dichas características se irán propagando a través de la población, mejorando poco a poco el cruce de los individuos mejor adaptados al medio y explorando las áreas más prometedoras del espacio de búsqueda de las posibles soluciones. Si el AG ha sido diseñado correctamente, las poblaciones irán convergiendo hacia una solución óptima del problema. Algunas de las ventajas de aplicar AG a la resolución de problemas son:

- Llevan a cabo una búsqueda en paralelo operando de forma simultánea con varias soluciones.

- No precisan conocimiento profundo sobre el problema que intentan resolver.
- Al ser usados en problemas de optimización son menos proclives a caer en máximos locales que las técnicas tradicionales.
- Son sencillos de implementar y ejecutar en arquitecturas paralelas.

Sin embargo, esta técnica de inteligencia artificial también presenta desventajas o limitaciones.

- Pueden tardar mucho en converger, o no llegar nunca a converger. Esto es debido en gran medida a los parámetros con los que se utilicen: tamaño de la población, número de generaciones, etc.
- Pueden converger prematuramente hacia óptimos locales debido a una serie de problemas de diversa índole como puede ser la no inicialización aleatoria de la población inicial.

Como hemos visto con anterioridad, los AG se utilizan normalmente para resolver problemas que tiene su origen en diversas áreas de estudio. Generalmente, al plantear un problema, se estudia el modo en que éste será resuelto y de este modo es implementado en forma de instrucciones que finalmente ejecutará una computadora. Desafortunadamente, esta tarea de encontrar un algoritmo adecuado para solventar un problema específico no resulta siempre sencilla, como ocurre en la mayoría de problemas de búsqueda u optimización. Los AG son métodos adaptativos que pueden ser utilizados para resolver este tipo de problemas. Si bien nunca se garantiza que el AG encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable en un tiempo razonable comparado con el resto de algoritmos.

El campo de aplicación de los AG se relaciona con los problemas de clase NP para los que no existen técnicas especializadas para resolverlos mejorando a una búsqueda exhaustiva de complejidad computacional exponencial. En el caso en que exista una técnica especializada que supere a una búsqueda exhaustiva, pueden efectuarse mejoras de las mismas hibridándolas con los algoritmos genéticos. Un claro caso de esto son los sistemas híbridos de AG y RNA como el que se presenta en esta tesis doctoral.

Sin embargo, los AG no son apropiados para resolver toda clase de problemas de optimización. Así pues, se deben tener en cuenta las siguientes características del problema antes de aplicar AG:

- El espacio de búsqueda del problema, i.e. sus posibles soluciones, debe estar delimitado dentro de un cierto rango.
- Debe ser posible definir una función de evaluación que nos permita conocer lo eficaz que es una posible solución dada.
- La codificación de las posibles soluciones debe ser fácil de implementar.

En el primer punto, referente al espacio de búsqueda, los AG son eficientes si el problema a resolver dispone de un amplio espacio de búsqueda discreto, con un gran número de dimensiones y muchos valores posibles para cada una de ellas. También es posible aplicarlos a espacios de búsqueda continuos, pero en este caso el problema debe tener un rango de posibles soluciones relativamente pequeño.

En cuanto a la función de evaluación se refiere, consiste en la función objetivo del problema de optimización. El AG irá proporcionando como resultado soluciones que la optimicen, ya sea maximizando o minimizando su valor (definido por el problema). Para esta función debe tenerse en cuenta que ésta debe *"premiar"* a las buenas soluciones y *"castigar"* a las malas, buscando así que los mejores individuos puedan seguir perpetuando sus buenas características a la especie en un futuro. En nuestro caso, el objetivo de la función de evaluación es minimizar el error de validación dado por una generación de RNA.

Respecto a la codificación de las soluciones, la manera más común de llevarla a cabo es mediante las cadenas binarias. Ésta es la propuesta hecha originalmente por Holland [50], además de ser la más sencilla de implementar. Cabe destacar otras codificaciones haciendo uso de números enteros, reales y caracteres.

Veamos a continuación una muestra del proceso general de un AG:

1. $D_0 \Leftarrow$ generamos y evaluamos una población inicial de soluciones
2. Iteramos desde generación k = 0,1,2,..., hasta que se alcanza un criterio de parada
   a) $D_k^{Elitism} \Leftarrow$ seleccionar un subconjunto de soluciones de D_k
   b) $D_k^{Crossover} \Leftarrow$ aplicar sobrecruzamiento a las soluciones de D_k
   c) $D_k^{Mutation} \Leftarrow$ aplicar mutación a las soluciones de $D_k^{Crossover}$
   d) $D_{k+1} \Leftarrow$ crear una nueva población con las soluciones de $D_k^{Mutation}$ y $D_k^{Elitism}$
   e) $D_k \Leftarrow D_{k+1}$
   f) Evaluar soluciones de la nueva generación de individuos D_k

2.4. Algoritmo de Estimación de Distribución

Los *Algoritmos de Estimación de Distribución* (AED) constituyen una familia de metaheurísticas derivadas de los algoritmos evolutivos [53]. En los AG, una población de soluciones individuales de un problema se mantiene como parte de la búsqueda de una mejor solución. Dependiendo de la especificación del AG, los cromosomas pueden ser cadenas de bits, números reales, etc. En un AED, esta representación explícita de la población se sustituye por una distribución de probabilidad sobre las posibles opciones en cada elemento del cromosoma que representa un miembro de la población.

Por ejemplo, si la población está representada por cadenas de bits de longitud 4, el AED para dicha población sería un único vector de cuatro casillas (p1, p2, p3, p4), donde cada p_i es la distribución de probabilidad de que dicha posición tome cualquiera de los valores posibles. Gracias al uso de este vector de distribución de probabilidades es posible crear un número arbitrario de soluciones candidatas.

En CE, las nuevas soluciones se generan a menudo combinando y modificando los ya existentes de una manera estocástica. La probabilidad de distribución de las nuevas soluciones sobre el espacio de todas las posibles normalmente no es especificado. En AED una población se puede aproximar con una probabilidad de distribución y se pueden obtener nuevas soluciones candidatas por muestreo de dicha distribución. Esto puede dar lugar a ventajas tales como evitar una posible convergencia prematura y ser una representación mas compacta.

Existen diferentes propuestas a la hora de llevar a cabo la distribución de probabilidad, usando modelos univariados [54, 55], bivariados [56, 57] y n-variados [58]. A continuación presentamos una breve descripción de estos:

- Modelos univariados: Los modelos más simples. No consideran relaciones de dependencias entre los distintos elementos de los cromosoma, las distribuciones son entonces univariantes. Las dos principales aproximaciones son el algoritmo UMDA del inglés *"Univariate Marginal Distribution Algorithm"* significa algoritmo de distribución univariada marginal [55] y el algoritmo *PBIL* del inglés *("Population-Based Incremental Learning")* [54].

- Modelos bivariados: En estos modelos se permite que un elemento del cromosoma dependa de otro. Si el modelo es observado como un grafo dirigido, esto implica que cada elemento puede tener a lo sumo un elemento padre. Cabe destacar que en los modelos univariados debido a la ausencia de dependencias, el grafo resultante sería un grafo totalmente desconectado. El

modelo bivariado más destacado es *MIMIC*, del inglés *"Mutual Information Maximization for Input Clustering"* [56].

- Modelos n-variados: En el que no existe restricciones en cuanto al número de dependencias entre elementos de los cromosoma. El modelo n-variado más habitual es aquel que permite estimar el modelo en forma de red bayesiana. *EBNA*, del inglés *"Estimation of Bayesian Networks Algorithm"* [59].

Tal vez porque el AED es una técnica reciente, se han llevado a cabo pocos estudios híbridos, es decir combinando RNA y AED. Además, estos han sido aplicados tan sólo a dominios de clasificación [60].

2.5. Algoritmo Evolución Diferencial

El algoritmo *Evolución Diferencial* (ED) es un método de optimización matemática de funciones multidimensionales que pertenece a la clase de optimizadores dentro de la computación evolutiva.

ED encuentra el mínimo global de una función multimodal (existe más de un mínimo) y multidimensional. ED es un algoritmo de optimización estocástica llevado a cabo por Storn y Price [61]. Originalmente, el método estaba enfocado a la resolución del problema de ajuste de polinomio de Tchebychev [62] utilizando una variante del método llamado recocido genético [63]. La comunidad de ED ha crecido desde mediados de 1990 y hoy en día, cada vez más investigadores trabajan en ED y con ED [64, 65, 66, 67].

Lo que caracteriza al ED es la generación de vectores de prueba, los cuales compiten con los individuos de la población actual con el objetivo de sobrevivir. ED se diferencia de otros algoritmos evolutivos en su mecanismo de generación de descendencia. En los algoritmos evolutivos, el individuo desempeña el papel de un padre para generar descendientes. ED sin embargo añade la diferencia asociada a un peso entre dos cromosomas de la población a un tercero. Éste será entonces el nuevo individuo de prueba que compita con los individuos de la generación actual. En ED, una población de cromosomas (posibles soluciones) es actualizado sucesivamente por adición, sustracción, e intercambio de componentes, hasta que la población converge. No se utilizan instrumentos derivados, muy pocos parámetros son fijados y se trata de un método simple y fiable [61].

En 1996, ED fue presentado en el *Concurso Internacional sobre Optimización Evolutiva* (CIOE), que buscaba comparar el potencial de distintos métodos de optimización de computación evolutiva, finalizando ED en tercer lugar. ED resultó

ser el mejor tipo de los algoritmos evolutivos para resolver los verdaderos valores de la función de prueba presentada como problema en dicho concurso.

2.6. Predicción de Series Temporales con Computación Evolutiva

Se han llevado a cabo varios trabajos hasta la fecha para predecir series temporales mediante AG [68]. En algunos de ellos, a partir de los datos conocidos, incluso se han generado modelos matemáticos mediante programación genética [69] (i.e. aproximación de ecuaciones), los cuales serán usados posteriormente para predecir los valores futuros.

Algunos estudios como el llevado a cabo por Szpiro [70], son capaces de predecir series temporales caóticas pero deterministas. Para ello, primero se estudia si los datos de una serie temporal son generados por un proceso determinista o bien por un proceso aleatorio. Esto se puede determinar indirectamente por la estimación de la dimensión fractal de la serie temporal, haciendo uso de métodos tales como el algoritmo Grassberger-Procaccia [71]. Una vez que se ha establecido con suficiente nivel de confianza que la serie temporal es originada por un proceso determinista, la predicción de los valores futuros, basándose en los valores pasados, se puede llevar a cabo.

Porecha et al. [72] se toparon con la limitación de poder predecir tan solo series temporales no estacionarias usando AG. Chai et al. [73] usan AG para estimar el valor de los parámetros de un modelo autorregresivo de media móvil (*AutoRegressive Moving Average models*, ARMA) para predecir series temporales. Cortez et al. [74] llevaron a cabo un estudio del uso de meta algoritmos genéticos para la predicción de series temporales. En él hacían uso de un AG binario para la búsqueda de un modelo para la predicción de series temporales, a la vez que otro AG era usado para la optimización de los parámetros propuestos por el autor del estudio.

Packard [75] también usó algoritmos genéticos para afrontar el problema de predicción en sistemas dinámicos. En [76, 77], dicho autor presenta un enfoque diferente en el cual se divide el espacio de entrada mediante reglas condicionales evolutivas, surgiendo más tarde trabajos que intentan mejorar dicho enfoque [78, 79]. Trabajos más recientes al llevado a cabo por Packard et al. dentro de los algoritmos evolutivos usan *"gene expression programming"* [80]. Cristobal et al. [81] proponen una tesis doctoral dedicada a este tema íntegramente.

Aunque donde más ha destacado el uso de AG para predecir series temporales

ha sido generando métodos híbridos, i.e. haciendo uso de AG con otras técnicas, como pueden ser RNA [82]. Otros trabajos también han usado la idea de construir modelos locales [83] usando el sistema de clustering k-windows.

No existen muchos estudios en los que se haya hecho uso de ED para llevar a cabo predicción de series temporales, en [84] Bauwens et al. presentan un algoritmo basado en ED para estimar y predecir cambios estructurales en series temporales. Además, no ha sido posible encontrar trabajos previos aplicados a este dominio con AED.

2.7. Redes de Neuronas Artificiales

Estudios sobre el cerebro humano [85] han concluido que éste puede contener más de cien mil millones (10^{11}) de neuronas y cientos de billones (10^{14}) sinapsis. Dichos estudios destacan también que puede haber más de 1000 sinapsis tanto a la entrada como a la salida de cada neurona. Cabe destacar que el tiempo de conmutación de la neurona (unos pocos milisegundos) es aproximadamente un millón de veces menor que el de las computadoras, y además, las neuronas biológicas tienen una conectividad miles de veces superior al de los actuales supercomputadores.

Existen tres partes bien definidas en una neurona:

- El cuerpo de la neurona.
- Las ramas de extensión llamadas dendritas para recibir las entradas.
- Un axón que lleva la salida de la neurona a las dendritas de otras neuronas.

Las conexiones entre neuronas (i.e. sinapsis) son la clave para el procesado de la información. La gran mayoría de las neuronas disponen de una estructura en forma de árbol, denominadas dendritas, las cuales reciben las señales de entrada provenientes de otras neuronas a través de las uniones. Mientras que algunas neuronas están comunicadas tan solo con las neuronas más cercanas, otras se conectan con miles.

La forma de interactuar entre dos neuronas no es totalmente conocida. En general, una neurona envía una señal de salida a otras a través de su axón. Éste transmite la información por medio de diferencias de potencial electroquímicos. La neurona recoge las señales a través de sus sinapsis y suma todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas son las

dominantes, entonces la neurona obtiene una señal positiva y envía este mensaje a otras neuronas a través de sus sinapsis de salida.

Las Redes de Neuronas Artificiales (RNA) [14] son un paradigma de procesamiento y aprendizaje inspirado en la forma en que funciona el sistema nervioso de los animales. Consiste en un sistema constituido por la interconexión de neuronas artificiales, que transforman una señal de entada en una señal de salida. El objetivo es conseguir un sistema que se caracterice por su capacidad de generalización y su robustez.

La salida dada por una RNA, es el resultado de aplicar consecutivamente las tres funciones siguientes:

1. **Función de propagación** (también conocida como función de excitación). Por lo general se basa en el sumatorio de cada entrada multiplicada por el peso asociada a la interconexión con otras neuronas. En el caso en que el peso es positivo, la conexión pasa a denominarse excitatoria; si por el contrario es negativo, se denomina inhibitoria.

2. **Función de activación.** Modifica el resultado de la salida de la función de propagación. Cabe la posibilidad de que no exista, pasando a ser en este caso la salida la misma función de propagación.

3. **Función de transferencia.** Es aplicada al valor dado por la función de activación. Es utilizada para acotar la salida de la neurona. Algunas de las funciones más utilizadas son la función sigmoidal (para acotar los valores entre el intervalo [0,1]) y la tangente hiperbólica (para obtener valores comprendidos en el intervalo [-1,1]). Debido a esto, esta función junto con la interconectividad de las neuronas artificiales son las características principales que le da a la RNA su capacidad de generar un modelo preciso a problemas no lineales. En ocasiones en la literatura la función de activación y la de transferencia se interpreta como una única función.

Se define capa de una RNA como cada no de los niveles de nodos o neuronas en los que estará dividida dicha red. Cada capa podrá tener un número de neuronas diferentes y cada una de estas neuronas podrá pertenecer sólo a una capa a la vez. Las capas pueden clasificarse en tres tipos:

- Capa de entrada: formada por aquellos nodos que introducen la información de entrada en la red. En estos nodos no se lleva a cabo procesamiento, es por ello que no pueden ser denominados como neuronas.

- Capas ocultas: constituidas por las neuronas cuyas entradas provienen de las capas anteriores y sus salidas pasan a neuronas de las capas posteriores.
- Capa de salida: las neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

El perceptrón multicapa es un tipo de RNA constituida por varias capas, por lo que se puede aplicar para obtener un modelo de un problema que pueda resolver problemas que no son linealmente separables [14], siendo esta la limitación principal del perceptrón (también denominado perceptrón simple).

Existen dos tipos de perceptrón multicapa, totalmente o parcialmente conectado. En el primer de los casos, cada una de las salidas de una neurona de la capa *"i"* es una entrada de todas y cada una las neuronas de la siguiente capa *"i+1"*. En el segundo caso, cada neurona de la capa *"i"* es entrada de una serie de neuronas (región) de la capa *"i+1"*. En la figura 2.2 se muestra un ejemplo de perceptrón multicapa.

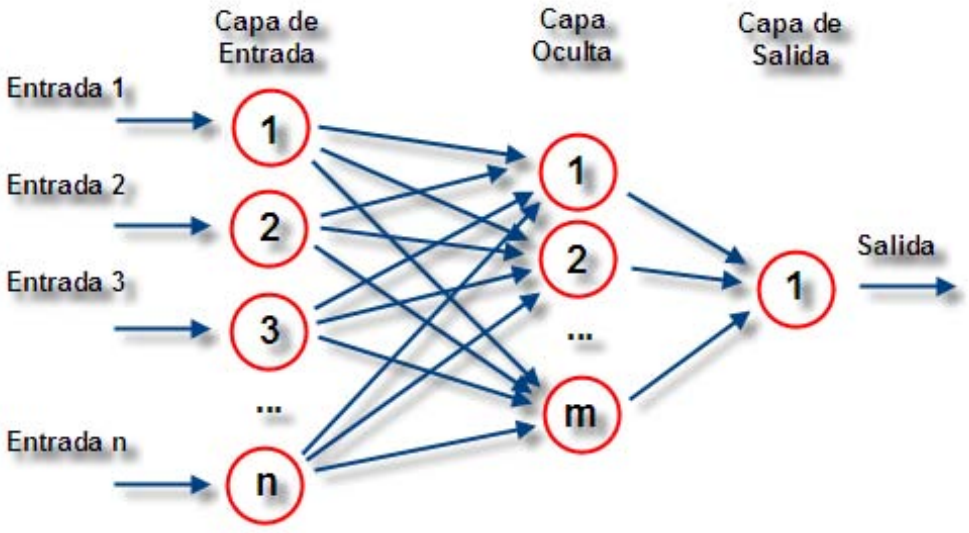


Figura 2.2: RNA perceptrón multicapa, totalmente conectado con *n* nodos de entrada, *m* neuronas en su capa oculta y una neurona de salida.

En esta tesis doctoral se ha optado por usar perceptrón multicapa como modelo computacional debido a su capacidad de aproximación de acuerdo con Cybenko [86].

Otro aspecto importante de las RNA es el proceso de aprendizaje. En el caso de aprendizaje supervisado, a la red de neuronas se le proporciona un conjunto de

ejemplos con los que aprender. En este aprendizaje los ejemplos dados a la red constan de valores de entrada y objetivo y tendremos además los valores de salida dados por la red de neuronas. Este aprendizaje se lleva a cabo modificando los valores asociados a cada una de las conexiones, los pesos de cada conexión, de modo que se minimice el error cometido entre las salidas que la red genera para cada patrón de entrada y el valor objetivo de salida.

Existen diferentes algoritmos de entrenamiento. El algoritmo QUICKPROP [87] es un método que acelera el aprendizaje mediante el uso de información sobre la curvatura de la superficie de error. Por otro lado, el principio básico de RROP [88, 89, 90] (*"Resilient Propagation"*) es eliminar la influencia nociva del tamaño de la derivada parcial sobre la escala de pesos.

El algoritmo conocido como propagación hacia atrás de errores o retropropagación [91] (del inglés *"BackPropagation"*) forma parte de la familia de los algoritmos de aprendizaje supervisado y es usado para llevar a cabo el entrenamiento de las redes de neuronas artificiales. Este algoritmo se basa en tratar de minimizar un error (normalmente cuadrático) mediante del descenso de gradiente, por lo que la parte esencial del algoritmo es el cálculo de las derivadas parciales de este error con respecto a los valores obtenidos como salida por la red de neuronas y el valor objetivo o real. Es por ello que el perceptrón multicapa es también conocido como red de retropropagación.

Diversas características de las RNA las hacen especialmente valiosas y atractivas a la hora de predecir series temporales.

En primer lugar, en contraposición a algunos modelos, las RNA son métodos dirigidos por los datos. Aprenden de ejemplos y capturan las relaciones funcionales entre los datos, incluso si dichas relaciones son difíciles de describir o desconocidas. Las RNA se adaptan bien para problemas cuyas soluciones requieren un conocimiento que es difícil de especificar pero para el que se dispone de suficientes datos u observaciones. En este sentido podrían ser tratadas como un método estadístico multivariante no paramétrico y no lineal [92]. Este modelo con la habilidad de aprender de la experiencia pasada es útil para muchos problemas prácticos, donde es sencillo tener datos pero no lo es en absoluto para realizar una conjetura apropiada sobre las leyes que gobiernan el sistema del que se han obtenido dichos datos. El problema con las RNA por otro lado, es que las observaciones o datos contienen ruido al proceder de problemas del mundo real.

En segundo lugar, las RNA pueden generalizar. Después de aprender los datos que se le presentan como ejemplo, pueden inferir la parte desconocida, incluso si los datos dados como ejemplo contienen ruido.

Ya que la tarea de predecir series temporales consiste en la predicción del com-

portamiento futuro a partir de los ejemplos pasados, resulta un área de aplicación interesante para las RNA.

En tercer lugar, las RNA son aproximadores funcionales universales. Se ha demostrado que una red puede aproximar cualquier función continua a cualquier precisión [86, 93, 94]. Las RNA tienen más formas funcionales flexibles y generales que los tradicionales métodos estadísticos [95]. Cualquier modelo de predicción asume que existe una relación subyacente entre las entradas (valores pasados de las series temporales) y las salidas (valores futuros). Frecuentemente, los modelos tradicionales de predicción estadísticos tienen limitaciones en estimar dicha función subyacente debido a la complejidad del sistema real. Las RNA es un método alternativo adecuado para identificar dicha función.

Finalmente, hemos de indicar que las RNA son no lineales, si la función de activación de la neuronas es no lineal (i.e. función sigmoidal o tangente hiperbólica).

En general, las RNA tienen las siguientes ventajas:

- Aprendizaje adaptativo: Las RNA disponen de la capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial mediante una etapa denominada de aprendizaje (memorizar y asociar hechos).

- Auto organización: Una RNA puede generar su propia representación de la información dentro de ella misma, descargando al usuario de esta tarea.

- Tolerancia a fallos: Puesto que una RNA puede almacenar la información de forma redundante, dicha RNA puede seguir respondiendo de forma aceptable incluso si ésta es dañada parcialmente. La destrucción parcial de una red conduce a una degradación de su estructura, sin embargo, algunas capacidades de la red se pueden retener.

- Flexibilidad: Una RNA puede manejar cambios que no son realmente importantes en la información dada como entrada, como pueden ser señales con ruido u otros cambios. Supongamos que la información de entrada es la imagen de un objeto, la respuesta asociada no sufre cambios si esta imagen es modificada un poco, ya sea cambiando su brillo o el objeto en sí ligeramente.

- Tiempo real: Los cómputos neuronales pueden ser realizados en paralelo, para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

Estas características de las RNA hacen de ellas bastante apropiadas para aquellos problemas en los que no disponemos a priori de un modelo identificable que pueda ser programado, pero por el contrario, sí dispongamos de un conjunto suficiente de ejemplos de entrada, ya hayan sido estos previamente clasificados o no.

Cabe destacar, que en los últimos años las RNA se han aplicado en conjunción con los AG. Esta disciplina que trata la evolución de redes de neuronas artificiales mediante computación evolutiva se denomina *Redes de Neuronas Artificiales Evolutivas* (RNAE) y será vista en más detalle en la sección 2.9. Este tipo de aplicación consiste en diseñar RNA, el genoma del AG lo constituyen los parámetros de la red (topología, algoritmo de aprendizaje, funciones de activación, etc.) y la evaluación de la red viene dada por la evaluación del comportamiento exhibido ante el problema al que se aplica.

Así pues, las RNA son aplicadas con éxito en problemas tales como reconocimiento de patrones de voz [96], señales [97], imágenes [98], etc. Como podremos ver en el apartado 2.8, también han sido utilizadas para realizar predicciones, desde meteorológicas hasta hacer predicciones en el mercado financiero, etc.

2.8. Predicción de Series Temporales con Redes de Neuronas Artificiales

Trabajos previos con RNA han demostrado que éstas tienen habilidades muy potentes de clasificación y reconocimiento de patrones [96, 98]. Inspiradas en los sistemas biológicos, las RNA son capaces de aprender y generalizar de la experiencia pasada. Actualmente las RNA se emplean en campos tan diferentes como el de los negocios, la industria o la ciencia [99, 100]. Una de las mayores aéreas de aplicaciones las RNA es en la predicción de series temporales [101, 102].

La predicción de series temporales ha sido durante mucho tiempo un ámbito de aplicación de modelos estadísticos lineales. Las propuestas de modelos lineales para la predicción de series temporales, como la de Bob-Jenkins o también denominado método ARIMA [20], asume que las series temporales bajo estudio son generadas por procesos lineales. Los modelos lineales tienen la ventaja de que pueden ser entendidos y analizados en detalle. Sin embargo, pueden ser completamente inapropiadas si el mecanismo subyacente es no lineal. Así pues, no es razonable asumir *"a priori"* que una serie temporal ha sido generada por un proceso lineal. De hecho, los sistemas del mundo real son normalmente no lineales [103].

Han sido desarrollados diferentes sistemas no lineales como el sistema bilineal [104], el modelo TAR (Threshold Autoregressive Model) [105] y el modelo ARCH (AutoRegressive Conditional Heteroskedasticity) [106]. Sin embargo, estos modelos no lineales están todavía limitados en cuanto a que una relación explícita de los datos de la serie tiene que ser obtenida mediante hipótesis con pocos conocimientos de las leyes subyacentes. De hecho, la formulación de un modelo no lineal para un conjunto de datos es una tarea muy difícil ya que existen muchos patrones no lineales posibles y un modelo no lineal pre-especificado puede que no considere todas y cada una de las características importantes de la relación entre las salidas y las entradas. Por el contrario, las RNA son capaces de llevar a cabo un modelado no lineal sin que sea necesario aplicar conocimiento previo sobre las relaciones entre las variables de entrada y salida. Las RNA se basan en los datos para determinar el modelo. Por lo tanto, y debido a sus características, las RNA son herramientas de predicción más generales y flexibles.

La idea de usar RNA para llevar a cabo predicción de series temporales no es nueva. La primera aplicación conocida data de 1964. Hu [107] en su tesis doctoral, hace uso de una red adaptativa lineal para realizar predicciones meteorológicas. No fue sin embargo hasta 1986 cuando el algoritmo, de aprendizaje de los pesos, retropropagación fue usado por Rumelhart et al. [108]. En 1988, Werbos [109] demuestra que las RNA entrenadas con el algoritmo de retropropagación mejoran los resultados obtenidos por los métodos estadísticos tales como la aproximación Box-Jenkins.

En 1987 Lapedes y Farber [110] conducen un estudio simulado en el que se concluye que las RNA se pueden emplear para obtener modelos y predecir series temporales no lineales. Tanto Weigend et al. [111] en 1992 como Cottrell et al. [112] en 1995 estudian cómo las estructuras de las RNA afectan a la predicción de series temporales. A principio de los 90, Tang et al. [113], Sharda y Patil [114] y Tang con Fishwick [115], entre otros, dan a conocer resultados de varias comparaciones de predicciones entre RNA y el método estadístico Box-Jenkins.

En la competición organizada por Weigend y Gershenfeld [111], los ganadores para todos los conjuntos de datos llevaron a cabo la predicción con RNA. Otra competición más reciente, NNGC1 (2009), y sus ediciones anteriores NN3 y NN5 [116], proponen un conjunto de series temporales de características diversas, con el objetivo de evaluar la exactitud que los métodos propios de inteligencia artificial (entre los que se encuentra las RNA). Los vencedores en la competición NN3, celebrada en 2007, Xue y Yan [117] hacen uso de la fusión de dos modelos, siendo el primero de ellos redes de regresión general [118]. En estas competiciones, el objetivo era diseñar un sistema que obtuviera un modelo (en nuestro caso una RNA) para predecir cada una de las series temporales propuestas. En la propuesta

planteada en esta tesis doctoral, cada modelo, i.e. cada RNA, se obtiene mediante un sistema híbrido en el que se combinarán RNA y técnicas evolutivas para diseñar las primeras de un modo automático.

2.9. Redes de Neuronas Artificiales Evolutivas

Se denomina *"Redes de Neuronas Artificiales Evolutivas"* (RNAE) como la combinación de las RNA y los procesos de búsqueda evolutivos. Se puede distinguir tres características de las RNA sobre las que aplicar procesos de computación evolutiva: la evolución de los pesos de las conexiones, la evolución de las topologías y la evolución de las reglas de aprendizaje.

2.9.1. Evolución de los Pesos de Conexión

El aprendizaje en las RNA puede ser dividido, como se comentó anteriormente, en supervisado y no supervisado. El aprendizaje supervisado ha sido formulado como un proceso de entrenamiento de los pesos, en el cual se lleva a cabo un esfuerzo para encontrar un óptimo (o lo más óptimo posible) conjunto de pesos de conexiones para una red, de acuerdo a algún criterio de optimización.

Los algoritmos de aprendizaje tratan de minimizar el error cuadrático medio total entre la salida actual y el valor que debería tener (valor objetivo) de la RNA. Aunque estos algoritmos han sido usado en diversas áreas con resultados satisfactorios, en ocasiones se queda atrapado en mínimos locales de la función de error por lo que es ineficaz buscando mínimos globales [88].

Esta circunstancia se puede solventar considerando el proceso el aprendizaje como un proceso de búsqueda y esta búsqueda como la evolución de los pesos de las conexiones llevado a cabo por un AG. Los AG son adecuados para realizar una búsqueda en espacios de búsqueda largos y complejos, como es el espacio de los pesos asociados a cada conexión. Se han llevado a cabo muchos trabajos sobre este tema [119, 120, 121, 122]. En ese punto dos cosas se deben tener en cuenta:

- La representación de los pesos de las conexiones en el cromosoma.
- La evolución llevada a cabo por el AG, i.e. como se implementa.

Respecto a las representaciones, se pueden considerar principalmente dos tipos, la binaria y la decimal, las cuales se van a comentar a continuación.

Codificación Binaria

Una de las representaciones más convenientes es la codificación binaria [119, 121]. El conjunto de los pesos de conexiones en una RNA esta representado por la concatenación de todos los pesos de la red codificados en binario. Después de decidir el esquema de representación, se usa un AG para evolucionar una población inicial de individuos:

1. Decodificar cada individuo (cromosoma) de la generación actual (i) en un conjunto de pesos de conexiones y construir su correspondiente RNA a partir de dicho conjunto.

2. Calcular el error cuadrático medio total entre las salidas obtenidas y las salidas objetivo para cada RNA mediante la alimentación de patrones de aprendizaje a las RNA y definir dicho error como el valor de adecuación de cada una de ellas.

3. Obtener un número de descendientes haciendo uso de los operadores genéticos (selección, sobrecruzamiento y mutación), siendo estos la nueva generación ($i+1$).

En este caso, la arquitectura de la RNA, i.e. el número de nodos ocultos, la función de activación, la topología (el patrón de conexión entre todos los nodos) y la regla de aprendizaje, están prefijadas al proceso evolutivo. Además, en este caso no se modifica el peso de las conexiones hasta que no han sido presentados todos los patrones de entrenamiento a la RNA, al contrario de lo que ocurre en BP, donde se modifican con cada patrón. Por otro lado, el hecho de usar pocos bits para representar cada peso de conexión puede llevar a entrenamientos muy largos o fallidos [119, 123]. Si se usan muchos bits, se obtendrán cadenas binarias muy largas, que prolongaran la evolución drásticamente y la harán impracticable. El número de bits a usar para cada peso de conexión es todavía una cuestión abierta.

Es bien sabido que los nodos en una capa oculta de un perceptrón multicapa funcionan básicamente como detectores de características de entrada [14]. Por lo tanto en la codificación binaria, la separación dentro del cromosoma de los valores (pesos) asociados a cada conexión que llegan a un mismo nodo oculto dará lugar a que la explotación de las interacciones no lineales entre ellos sea mucho más difícil debido a la posible interrupción causada por el sobrecruzamiento [124]. Es por ello que para que el proceso evolutivo permita obtener un conjunto de pesos satisfactorios, sería necesario mucho más tiempo. Diferentes estudios han demostrado que la concatenación de los pesos de conexiones codificados en el cromosoma de acuerdo a los nodos ocultos a los que llegan es beneficiosa para la explotación y

para mantener los bloques funcionales útiles que se han formado alrededor de los nodos ocultos [125]. La probabilidad totalmente aleatoria del sobrecruzamiento de poder romper el cromosoma en cualquier punto ha sido también estudiado para llevar a cabo un sobrecruzamiento mejorado con una mayor probabilidad de corte entre la frontera de dichos bloques de pesos en lugar de romper el cromosoma de forma totalmente uniforme en cualquier punto [126, 127].

Codificación Decimal

Con el objetivo de solucionar las desventajas que presenta la codificación binaria se propone la codificación decimal, en la que se usa un número real codificado para representar el peso de conexión [120, 122, 128]. Así pues, una RNA se presenta como una concatenación de números reales, donde los pesos de las conexiones de entrada de un mismo nodo se incluyen de forma consecutiva en el cromosoma. Por ello, tampoco es aconsejable usar cualquier operador genético en la codificación decimal como se hacía en la binaria [124] ya que aquí el sobrecruzamiento también puede dar lugar a un corte no deseado entre bloques de pesos de conexión. Los operadores genéticos deben ser analizados y ajustados anteriormente a la ejecución del AG (o proceso evolutivo). Fogel [122], solo hacía uso de un operador genético, la mutación, y más concretamente de *"Gaussian random mutation"*, lo que produce un resultado parecido a trabajar con múltiples poblaciones en paralelo.

En esta tesis doctoral proponemos por lo tanto el uso de una codificación decimal. Pero en lugar de codificar todos y cada uno de los pesos dentro del cromosoma, se ha propuesto la codificación de la semilla de inicialización de los pesos para evitar el problema de los cortes no deseados dados por el sobrecruzamiento.

Retropropagación y Algoritmo Genético

La propuesta de un entrenamiento evolutivo parece atractiva, porque puede manejar el problema de la búsqueda global mejor en un espacio de búsqueda como es el de los pesos de las conexiones, un espacio vasto, complejo y no diferenciable. Sin embargo, implica un coste computacional excesivo, ya que el entrenamiento evolutivo computa despacio y de manera más intensa que el entrenamiento basado en descenso de gradiente, como ocurre en retropropagación. Kitano [129], llevó a cabo un estudio comparativo entre AG y variantes del algoritmo de retropropagación para el aprendizaje de los pesos de la red, y mostró que los AG consumen más tiempo a la hora de converger en comparación con el algoritmo de retropropagación.

Propuesta Híbrida

Visto lo anterior, una combinación de ambos métodos (AG y retropropagación) puede resultar beneficiosa a la hora de mejorar los resultados. El AG puede ser usado en primera instancia para llegar rápidamente, mediante una búsqueda global, a una adecuada región del espacio de búsqueda. Esta región representará entonces un punto de partida en el espacio de los pesos para llevar a cabo una búsqueda local, aplicando algoritmos como el BP, y encontrar así una solución subóptima, i.e. lo suficientemente cercana a la solución óptima.

Belew et al. [124] hicieron uso de AG para buscar un buen conjunto de pesos de conexión iniciales y después emplearon algoritmos de retropropagación para afinar esos pesos iniciales. Sus resultados mostraron que el sistema híbrido AG/retropropagación era más eficiente que los AG solos y además muy competitivo en comparación con retropropagación, a pesar de que el sistema híbrido necesitara más tiempo de computación. Aunque para poder igualar los resultados obtenidos por el algoritmo de retropropagación con los del sistema híbrido, los algoritmos de retropropagación tenían que ser lanzados en repetidas ocasiones, con lo que no era tanta la diferencia del coste computacional entre ambos esquemas.

Hoy en día son variados los trabajos de sistemas híbridos que se pueden encontrar en la literatura [130, 131, 132]. Esta tesis doctoral aporta un nuevo sistema híbrido a los ya presentados hasta la fecha.

2.9.2. Evolución de la Arquitectura

Aunque los pesos de las conexiones de una red son relevantes para la eficacia de la red, es sabido que la arquitectura de la red juega un papel importante en la habilidad de procesado de las RNA.

Aún hoy en día la tarea de buscar una arquitectura adecuada y los pesos de sus conexiones se llevan a cabo por expertos mediante técnicas de prueba y error. No existe un método sistemático para diseñar una arquitectura óptima que solucione una tarea particular. Algunos trabajos [133, 134, 135] basados en algoritmos constructivos/destructivos han sido uno de los esfuerzos llevados a cabo para el diseño automático de RNA. Los métodos constructivos empiezan con una red mínima (red con un mínimo número de neuronas ocultas y conexiones) y van añadiendo nuevos nodos y conexiones, si es necesario, durante el entrenamiento, mientras que los métodos destructivos comienzan con una red de tamaño máximo y progresivamente borran los nodos y conexiones innecesarios durante el entrenamiento.

Sin embargo, Miller et al. [136] describen algunas características que hacen

de la propuesta de evolución basada en AG, una mejor candidata, de entre las propuestas heurísticas como pueden ser los métodos constructivos/destructivos, para buscar en el espacio de búsqueda de las posibles topologías. Los motivos son los siguientes:

- El espacio de búsqueda es infinitamente extenso ya que el número de nodos y conexiones no tiene límites.
- El espacio de búsqueda es no diferenciable ya que los cambios en el número de nodos o conexiones son discretos y pueden tener un efecto discontinuo sobre el rendimiento de las RNA.
- El espacio de búsqueda es complejo y ruidoso ya que depende de las condiciones iniciales, i.e. valores iniciales de los pesos.
- El espacio de búsqueda es engañoso ya que redes con arquitecturas similares pueden tener rendimiento y habilidades de procesamiento diferentes.
- El espacio de búsqueda es multimodal ya que las RNA con diferentes arquitecturas pueden tener capacidades similares.

Debido a las ventajas del diseño evolutivo de la topología de la red, se han llevado a cabo trabajos muy variados aplicados a diferentes dominios [130, 131, 132, 136, 137, 138, 123] y es la opción elegida por nosotros en esta tesis doctoral.

Similar a la propuesta de la evolución de los pesos, el primer paso a llevar a cabo en este caso es decidir una codificación de la topología de una red en el cromosoma (por ejemplo la matriz de conexiones [139], gráfico [137], reglas de generación [140]).

La cuestión del diseño de RNA radica en decidir qué información de la topología debe ser codificada en el cromosoma. Por un lado toda la información sobre una topología (o la arquitectura, si se le incluye también los valores de los pesos de cada conexión) puede ser representada directamente por cadenas binarias. Es decir, cada nodo y/o conexiones aparecen especificados por algunos bits en el cromosoma. Este tipo de representación es conocido como *esquema de codificación directa*.

Otra opción es codificar tan solo los parámetros o características más importantes de una arquitectura, como puede ser el número de nodos, número de conexiones y el tipo de función de transferencia de las neuronas.

Diferentes trabajos han estudiado la codificación de una RNA dentro de un cromosoma para llevar a cabo su evolución. Algunos de ellos haciendo uso de un

esquema de codificación directa [141, 122], mientras que otros usan un esquema de codificación indirecta [142, 143, 137]. Una ventaja conocida del esquema de codificación directo es que es fácil de evolucionar RNA con propiedades de conectividad especiales [144], ya sea mediante la limitación de las representaciones permitidas o mediante la inclusión de algún término sancionador específico en la función de evaluación . Una desventaja es que estas representaciones no escalan bien en RNA grandes, con cientos de nodos y muchas más conexiones [144]. Pero cuando se trata de redes de menor tamaño, se han obtenido buenos resultados. El esquema de codificación directo ha sido adoptado por Whitley et al. [145], por Schaffer et al. [123] y por Maniezzo [144] entre otros.

A continuación se presentarán las características más importantes de cada una de ellas.

Esquema de Codificación Directa

En el esquema de codificación directa, cada conexión de cada red se especifica directamente mediante su representación binaria [138, 123, 146], o decimal [13], por ejemplo considerando la matriz de conexiones como el propio cromosoma de un individuo.

En general, una matriz de tamaño $N \times N$, donde la matriz bidimensional $C = (c_{ij}), i, j = 1 \ldots N$ representa la conectividad de una red con N nodos. El elemento c_{ij} indica la presencia o ausencia de conexión del nodo *i* hacia el nodo *j*. La cadena binaria en que quedaría codificado el cromosoma sería tan solo la concatenación de las filas (o columnas) de la matriz. Esta codificación es muy adecuada para redes pequeñas, es decir, con un número de nodos limitado.

Sin embargo, esta forma de codificar las conexiones pueden dar lugar a búsquedas lentas ya que las RNA grandes implican grandes matrices que las representen lo que hará que la evolución sea más lenta o incluso que no llegue a converger. Una manera de reducir el tamaño de las matrices es hacer uso del máximo conocimiento previo del dominio del problema y restringir así el espacio de búsqueda. Además, el problema funcional de los elementos contiguos del cromosoma que hacen referencia a elementos contiguos en la topología de la red presentado en el apartado 2.9.1, también esta presente para este tipo de codificaciones.

La función de evaluación (i.e. fitness) puede definirse fácilmente y, se ha estudiado y demostrado, que aquellas funciones de evaluación en las que se usa el error de validación en lugar del de entrenamiento hacen que los modelos obtenidos generalicen mejor [123].

Esquema de Codificación Indirecta

Una forma de reducir la longitud de la representación de la conectividad es codificar tan solo las características más importantes de la red (nodos de entrada o nodos en la capa oculta) en lugar de cada una de las conexiones [140, 147, 148, 138, 149, 150, 151].

En este caso un cromosoma no representa directamente una posible solución en forma de fenotipo. Requiere el acceso a datos externos y un proceso de traducción no trivial para producir una solución real.

La codificación dentro del cromosoma de reglas de desarrollo para generar redes de neuronas (lo cual constituye uno de los mejores esquemas de codificación indirecta) puede producir representaciones muy compactas, pero sufre los problemas del ruido causados por la transformación de las reglas de desarrollo en arquitecturas de red neuronal.

Un beneficio inmediato del esquema de codificación indirecto es una representación compacta de la conectividad de las RNA. Además, este esquema de codificación es, biológicamente hablando, más plausible que la codificación directa. Debido a esto, se ha seleccionado este tipo de codificación para ser usada en el presente estudio. Existen diferentes maneras de llevar a cabo la codificación indirecta y las más frecuentes se detallan a continuación.

A) Codificación de los Parámetros de Conectividad

Una de las posibles formas de llevar a cabo una codificación indirecta es mediante la codificación de los parámetros de conectividad. Existen varios conjuntos de parámetros que pueden ser usados para especificar la conectividad. Harp et al. [138] adoptaron las *"blueprints"*, una clase de representación binaria indirecta que está compuesta por uno o más segmentos, para codificar la conectividad de la red y los parámetros de aprendizaje usados por el algoritmo de entrenamiento retropropagación [138, 149]. Cada segmento consiste en dos partes:

1. Un parámetro de especificación del área, el cual es de tamaño fijo y parametriza el área (i.e. capa en una red) en términos de su dirección, numero de nodos que tiene, organización de dichos nodos, y parámetros de aprendizaje asociados a dichos nodos.

2. Campos de especificación de proyección, es decir, la densidad de conexión, la dirección de las áreas objetivos y la organización de las conexiones.

Una representación paramétrica para realizar una conectividad similar ha sido llevada a cabo por Hancock [151] y Dodd et al. [150]. Aunque estos métodos pueden reducir el tamaño del cromosoma que especifica la conectividad de la RNA, todavía no han conseguido la suficiente escalabilidad requerida por muchos problemas del mundo real, ya que la longitud crece rápidamente con RNA de gran tamaño.

B) Codificación de las Reglas de Desarrollo

Un método de codificación bastante diferente del que se acaba de describir consiste en codificar las reglas de un método de desarrollo de lo que finalmente se considerará la matriz de conexiones de una red. Exactamente se codifican en el cromosoma las reglas de una gramática bidimensional [137, 152, 142].

Este método presenta las siguientes ventajas, una mejor escalabilidad, mejor regularidad y capacidad de generalización de las redes resultantes, puesto que las reglas no crecen con el tamaño de la red que genera. Además el problema funcional de los bloques, causado por el sobrecruzamiento, será menos severo ya que el método de codificación de reglas afecta más sobre las conexiones entre grupos de nodos que entre esos nodos en sí [137].

Una regla de desarrollo en el método de codificación de reglas se describe normalmente como una regla recursiva [152], o una regla de generación similar a una regla de producción en un sistema basado en conocimiento, con una parte izquierda y otra derecha [137]. El patrón de conectividad de una red en la forma de una matriz de conexiones se construye partiendo desde un elemento base o axioma. Aplicando repetidamente las reglas de derivación para los elementos no terminales de la matriz actual, hasta que la matriz contenga tan solo elementos terminales. Estos elementos terminales indicarán la presencia o ausencia de una conexión.

Otra posible variante consiste en que en lugar de representar los elementos terminales como *"0"* o *"1"* para indicar la presencia o no de una conexión, permitir que dichos elementos terminales puedan tomar valores reales para indicar el peso de dicha conexión.

El paso a llevar a cabo en la construcción de una matriz de conexión consiste en aplicar las reglas cuya parte izquierda aparecen en la matriz y reemplazarlas por la sus respectivas parte derecha. En la figura 2.3 se observa ejemplos de reglas de escritura usadas para construir la matriz de conectividades mediante una codificación indirecta basada en gramáticas, siendo S el elemento inicial o axioma. Mientras que en la figura 2.4 se puede observar un ejemplo del desarrollo de una

topología de una RNA usando las reglas de la figura 2.3.

$$S \longrightarrow \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$A \longrightarrow \begin{pmatrix} a & a \\ a & a \end{pmatrix} \quad B \longrightarrow \begin{pmatrix} i & i \\ i & a \end{pmatrix} \quad C \longrightarrow \begin{pmatrix} i & a \\ a & i \end{pmatrix} \quad D \longrightarrow \begin{pmatrix} a & e \\ a & e \end{pmatrix}$$

$$a \longrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad c \longrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad e \longrightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad i \longrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Figura 2.3: Ejemplos de las reglas de escritura usadas para construir la matriz de conectividades, en una codificación indirecta basada en gramáticas. S es el elemento inicial o axioma.

$$\text{S} \qquad \begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad \begin{pmatrix} a & a & i & i \\ a & a & i & a \\ i & a & a & e \\ a & c & a & e \end{pmatrix}$$

(a) (b) (c)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(d)

Figura 2.4: Ejemplo del desarrollo de una topología de una RNA usando las reglas de la figura 4 (a) Paso inicial: axioma; (b) Paso 1; aplicar la regla del axioma; (c) Paso 2: aplicar las reglas para cada uno de los no terminales; (d) Paso 3: cuando todos los elementos de la matriz son terminales, es decir, 1 ó 0, y se tiene una matriz de conexiones.

Kitano [137] generó su propia versión de codificación de reglas con la que obtuvo mejores resultados, con problemas de diferentes tamaños, comparadas con la codificación directa. Aunque más tarde Siddiqi et al. [139] llevaron a cabo un

estudio que mostraba que la codificación binaria directa obtiene resultados comparables con los sistemas de codificación basados en reglas como el presentado por Kitano.

La propuesta de optimizar la conectividad y los pesos de conexión al mismo tiempo fue llevada a cabo por Bornholdt y Graudenz, aunque ellos no usaron el método de codificación de reglas para la conectividad [153], sino una codificación directa.

C) Codificación basada en Autómatas Celulares

Gutierrez et al. presentan en [154] una codificación basada en autómatas celulares. En ella se hacía uso de una matriz binaria para codificar las conexiones y la topología de la red (RNA con una única capa oculta) a la vez. Dicha matriz representaba las conexiones que se daban entre los nodos de la capa de entrada y las neuronas de la única capa oculta y las conexiones existentes entre las neuronas de la capa oculta y las de la capa de salida.

Por lo tanto, las dimensiones de la matriz serán dadas por el tamaño del problema. El número de nodos presentes en la capa de entrada estará definido por el número de atributos de que disponga cada patrón del dominio de aplicación, mientras que el número de neuronas de salida será equivalente al número de variables de salida de dicho patrón.

Según esta codificación, cada columna de la matriz representará todas las posibles conexiones dadas desde o hacia un nodo de la capa oculta, mientras que una fila de esta matriz equivale a las conexiones de un nodo de la capa de entrada hacia las neuronas de la capa oculta y las conexiones de una neurona de la capa oculta hacia una misma neurona de salida.

La matriz de conexiones tiene entonces dimensiones $N \times M$ donde *N* es el número de filas y *M* el número de columnas. Sea a_{ij} el valor del elemento de la matriz, donde *i* y *j* indican respectivamente el número de fila y columna, y sean *E* y *S* el número de entradas y salidas de la RNA respectivamente. El elemento a_{ij} se interpreta entonces como:

- si $i < E$, entonces a_{ij} representa una conexión entre el i-ésimo nodo de entrada y la j-ésima neurona de la capa oculta.
- Si E $i < (E + S)$, entonces a_{ij} representa una conexión entre la j-ésima neurona de la capa oculta y (i-E+1)-ésima neurona de la capa de salida.

Entonces, un *"1"* en la matriz de conexiones es interpretado como la existencia de la conexión indicada por la celda y un *"0"* como la ausencia de la misma. En la figura 2.5 podemos observar un ejemplo de una matriz de conexiones binaria.

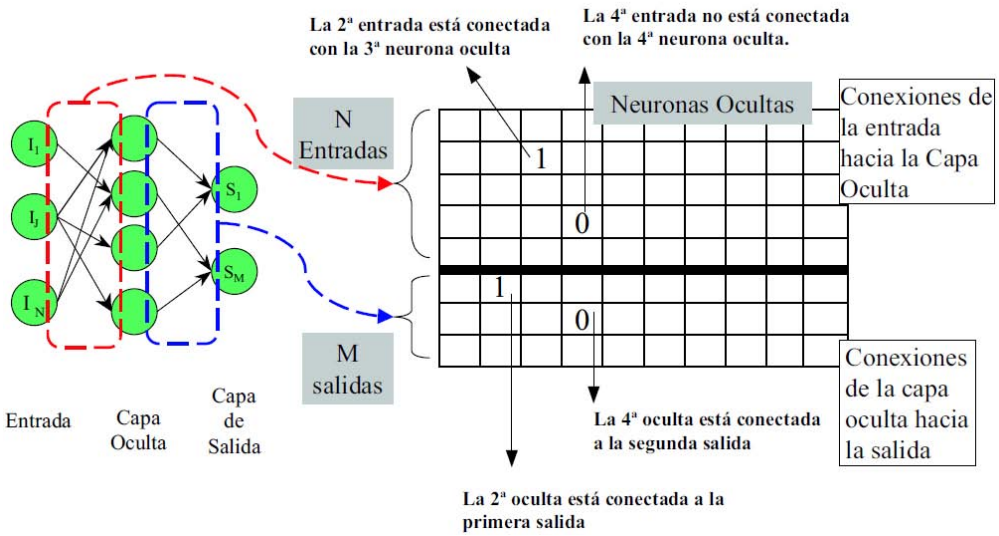


Figura 2.5: Estructura y significado de la matriz de conexiones.

D) Representación Fractal de la Conectividad

Merrill y Port propusieron un método para llevar a cabo la codificación de la conectividad, el cual está basado en el uso de subconjuntos de fractales del plano [155]. Ellos argumentaban que la representación fractal de la conectividad era, biológicamente hablando, más plausible que la representación de la codificación de reglas porque existen evidencias de que parte del cuerpo humano, como el pulmón [156], está estructurado fractalmente. Aun así, este método tiene menos escalabilidad que el método basado en reglas.

Evolución de las Funciones de Transferencias entre Nodos

No se han llevado a cabo muchos trabajos referentes a la evolución de las funciones de transferencias de los nodos a pesar de que se ha demostrado que dicha función es una parte importante de una arquitectura y tiene un impacto significativo en el rendimiento de las redes [157, 158].

Mani [157] propuso un algoritmo de retropropagación modificado que lleva a cabo una búsqueda del descenso de gradiente en el espacio de búsqueda de los pesos al mismo tiempo que realiza una búsqueda en el espacio de las funciones de transferencia, aunque la conectividad de la red era fija.

Stock et al. [158] fueron los primeros en aplicar una propuesta basada en AG para la evolución de ambos, conectividad y funciones de transferencia de los nodos, aunque solo fueron investigadas redes muy simples (alrededor de unos 7 nodos).

La función de transferencia en los nodos de una red puede ser diferente y decidida de manera automática por un proceso evolutivo en lugar de de ser asignada por un experto. En general, los nodos dentro de un grupo, como una capa en una RNA, tienden a tener el mismo tipo de función de transferencia con posibles diferencias en algunos parámetros, mientras que diferentes grupos de nodos pueden tener diferente funciones de transferencia.

2.9.3. Evolución de las Reglas de Aprendizaje

Es sabido que diferentes topologías y tareas de aprendizaje necesitan diferentes algoritmos de entrenamiento. Sin embargo, los AG son adecuados para entrenar redes con conexiones recursivas y redes profundas, es decir, con varias capas ocultas; mientras que el algoritmo de retropropagación obtiene buenos resultados entrenando redes con pocas capas ocultas. Es por ello que el sistema híbrido diseñado y presentado en esta tesis doctoral presenta las ventajas dadas por el algoritmo de retropropagación al entrenar redes con pocas capas ocultas, en nuestro caso con una única capa oculta.

Incluso después de haber seleccionado un algoritmo de entrenamiento, existen varios parámetros del algoritmo que deben ser especificados. La optimización de los algoritmos de entrenamiento y sus parámetros para una red y una tarea de aprendizaje es normalmente muy difícil de llevar a cabo y se basa en la técnica de prueba y error.

Se han llevado a cabo algunos trabajos para ajustar los parámetros del algoritmo BP, tales como la razón de aprendizaje y el momento, por medio de métodos heurísticos o evolutivos [129, 159]. Pero el asunto más fundamental, optimizar la regla de aprendizaje, i.e. la regla que actualiza los pesos de las conexiones, tan solo ha sido usado en un número limitado de estudios [160, 161, 162]. Aunque la regla de aprendizaje Hebbian [163] es extensamente aceptada y usada como base en muchos algoritmos de aprendizaje, trabajos llevados a cabo por Hancock et al. [164] han mostrado que otra regla de aprendizaje basada en los trabajos de

Artola et al. [165] es más potente que la mas adecuada regla Hebbian. Esta puede aprender más patrones, además de excepciones y regularidades. En el presente, esta clase de búsqueda de una regla de aprendizaje óptima solo puede ser llevada a cabo por expertos debido a su experiencia mediante prueba y error. Resulta muy atractivo desarrollar un método automático que optimice las reglas de aprendizaje para una red y una tarea de aprendizaje. El ajuste mediante un proceso evolutivo de los parámetros del algoritmo BP, como pueden ser la razón de aprendizaje y el momento, a lo largo de la evolución, puede ser considerado como un primer paso de la evolución de las reglas de aprendizaje [129, 138].

2.10. Predicción de Series Temporales con Redes de Neuronas Artificiales Evolutivas

El campo de la predicción de series temporales es un buen dominio para llevar a cabo pruebas de sistemas basados en RNAE, ya que el desarrollo neuronal no dependerá solo de la topología de la red, sino también del correcto conjunto de sus entradas, en este caso de los instantes anteriores de la serie temporal que usamos para predecir el instante siguiente y como tratamos dichos datos.

Aunque estos sistemas híbridos han sido usados en mayor medida para resolver problemas de clasificación [166, 167], existen diferentes estudios en los que tales sistemas se aplican al campo de la predicción de series temporales [166, 167, 168, 169, 170].

Sin embargo se han llevado a cabo pocos estudios [64, 65] sobre predicción de series temporales empleando un sistema híbrido formado por RNA y un algoritmo evolución diferencial.

En cuanto a AED, hasta la fecha, tan solo han sido llevados a cabo algunos estudios híbridos aplicándolos a dominios de clasificación [53], nunca aplicándolos a predicción de series temporales.

En el caso del diseño automático de RNA para la predicción de series temporales, la manera de usar dichos algoritmos consistiría en sustituir el AG que se encarga de llevar a cabo la búsqueda a nivel global, por los citados algoritmos.

2.11. Teoría de la Sabiduría de los Grupos

La sabiduría de los grupos [171] es una teoría que explica por qué el conocimiento compartido por muchos individuos es mayor y más inteligente que el aportado por unos pocos y cómo la sabiduría colectiva da forma a los negocios, economía, sociedades y naciones. Dicha teoría trata sobre la combinación de la información en grupos, que termina en decisiones que a menudo son mejores que las que podrían haber sido tomadas individualmente por cada miembro del grupo.

A la hora de buscar una solución que se aproxime lo más posible a un objetivo para un problema dado, se han empleado diferentes técnicas de inteligencia artificial, incluso se ha estudiado [97] cómo a veces estas técnicas se combinan para obtener un sistema híbrido que nos ayude a mejorar el resultado.

En esta propuesta se aplicará la idea de seleccionar y combinar modelos simples con el fin de incrementar la exactitud del sistema frente a la de un único modelo simple. Se trata de comprobar que la cooperación alcanzada entre diferentes modelos puede resultar en un incremento de la eficacia frente al alcanzado por un único modelo. La meta consiste en diseñar un conjunto de estrategias de aprendizaje que proporcionen una buena generalización y que puedan adaptarse dinámicamente al dominio con el que trabajan.

Yao [130, 172] ha llevado a cabo varios estudios en los que al finalizar la ejecución de un sistema evolutivo aplicado al diseño de RNA, en lugar de usar el mejor individuo (i.e. la mejor red y sólo esa), considera un conjunto de las diferentes soluciones proporcionadas por los que constituyen los mejores individuos (*"Ensembles"*). A la hora de determinar qué individuos forman el conjunto, Yao consideraba distintas opciones:

- Todos y cada uno de los individuos de la última generación.
- O para reducir el coste computacional, seleccionar tan solo algunos de los mejores individuos de la última generación según su valor de adecuación.

Claro está, siempre respetando la diversidad genética (elegir individuos, con muchas semejanzas no tiene ningún sentido puesto que sus soluciones también podrán serlo). Otros estudios que relacionan la evolución de RNA con la teoría de la sabiduría de los grupos son [166, 16], donde son aplicados tanto al dominio de clasificación como al de regresión.

2.12. Resumen

A lo largo de este capítulo, se ha analizado lo práctico que es el uso de RNA para la predicción de series temporales. También se ha comentado previamente que determinar la topología de las RNA (i.e. conectividad), es una tarea complicada que se lleva a cabo normalmente por expertos mediante procedimientos de prueba y error, es decir una búsqueda ciega, en un conjunto amplio de posibles configuraciones. Se han estudiado métodos más elaborados, tales como algoritmos de poda y constructivos [133, 134, 135]. Además estos métodos presentan dos grandes desventajas, tienden a quedarse atrapados en un mínimo local y buscan solo en una pequeña porción de arquitecturas como posibles soluciones en lugar de hacerlo en el espacio de búsqueda completo.

Una alternativa diferente es la que ofrece la CE, inspirada en la selección natural. Los algoritmos evolutivos son candidatos para la optimización de tareas, llevando a cabo una búsqueda global, localizando áreas de alta calidad rápidamente, incluso cuando el espacio de búsqueda es extenso y complejo [173].

La combinación híbrida de computación evolutiva y RNA (RNAE, [166, 167]) es un candidato adecuado para el diseño de las topologías debido a las características del error de la superficie: los cambios son discretos, pueden proveer efectos discontinuos y topologías similares pueden presentar actuaciones diferentes, mientras que arquitecturas diferentes pueden proveer salidas similares.

En general, encontrar un modelo de RNA adecuado para una serie temporal particular es un tema importante. Diferentes estudios han tratado por separado con el diseño de RNA centrándose en tres aspectos de éstas:

- Pesos de las conexiones: Valor para cada conexión presente en una RNA.
- Topología: Número de nodos en cada capa oculta, número de capas ocultas, número de nodos en la capa de entrada y salida.
- Reglas de aprendizaje: Factor de aprendizaje y valores del momento.

En esta tesis doctoral se propone un sistema híbrido evolutivo que hace uso de ambos, AG y del algoritmo de aprendizaje de retropropagación. Esta aproximación implica la evolución de las topologías de las RNA, de los pesos de conexión e incluso de las reglas de aprendizaje, en forma de factor de aprendizaje asociado al algoritmo BP, conjuntamente.

Además, hemos llevado a cabo una breve introducción sobre AED y ED como alternativa a la hora de llevar a cabo una búsqueda global de los parámetros de las RNA en lugar del AG.

También hemos presentado los *"Ensembles"*, que ayudará a mejorar el resultado haciendo uso de una combinación de soluciones en lugar de optar por una de forma individual.

Se propone por tanto en esta tesis doctoral el uso de estos algoritmos evolutivos (AG, AED y ED) en combinación con RNA, siendo este un campo de investigación novedoso e interesante, para tratar de obtener mejores resultados de los obtenidos hasta la fecha por otros trabajos o métodos. Un estudio comparativo entre estos diferentes algoritmos (i.e. AG, AED y ED), resulta también de gran interés.

Para finalizar, se llevará a cabo un estudio comparativo de estas técnicas frente a métodos estadísticos y herramientas software de predicción de series temporales además de compararlos con otros métodos conocidos basados también en inteligencia computacional.

Capítulo 3

Diseño de Redes de Neuronas Artificiales mediante Computación Evolutiva

En este capítulo se presenta una primera aproximación del sistema de diseño de Redes de Neuronas Artificiales (RNA) mediante Computación Evolutiva (CE) que se propone desarrollar en esta tesis doctoral. A dicho sistema lo denominamos ADANN, del inglés *"Automatic Design of Artificial Neural Networks"*.

El principal objetivo que aquí planteamos, consiste en desarrollar un único método de Inteligencia Computacional (IC) que pueda ser empleado para predecir cada una de las series temporales consideradas. Así, para cada serie temporal nuestro sistema obtendrá una RNA específica, aplicando siempre el mismo método automático. Esta RNA será utilizada después para predecir los valores futuros, y por lo tanto desconocidos, de esa serie temporal.

Esta aproximación tiene como objetivos los establecidos en la sección 1.2. Así pues, en primer lugar es necesario normalizar los datos de entrada de la serie temporal para que la RNA pueda trabajar con ellos y después se llevará a cabo otro paso importante que consiste en generar los subconjuntos de patrones, a partir de los datos previamente normalizados, con los que la RNA aprenderá y será evaluada. Este proceso se repetirá dentro de la evaluación de cada individuo durante el proceso evolutivo.

Para la codificación y representación de las RNA (fenotipo) en cromosomas (genotipo), hemos seleccionado los parámetros que hemos considerado más importantes, según lo estudiado en el estado del arte, y que explicaremos en más detalle en la sección 3.6.1. Como técnica de CE para llevar a cabo la búsqueda

global, se ha optado por los Algoritmos Genéticos (AG) (haciendo uso de su poder de exploración y explotación) y como algoritmo de aprendizaje, para aproximar mejor los valores dados por la búsqueda del AG a los parámetros de la RNA, se ha optado por el algoritmo de retropropagación. En cuanto al tipo de RNA elegido, se ha optado por el perceptrón de una sola capa oculta totalmente conectado y una única neurona de salida como podremos ver más adelante en la sección 3.6.1.

Para esta aproximación, se lleva a cabo además un estudio sobre la función de evaluación usada. Se ha estudiado y demostrado que aquellas funciones de evaluación en las que se usa el error de validación en lugar del de entrenamiento generalizan mejor [123]. Nosotros queremos en este capítulo corroborarlo presentando tres funciones de evaluación diferentes, en las que haremos uso del conjunto de entrenamiento, validación y un tercer conjunto al que llamaremos validación II y mostraremos mediante experimentación en el capítulo siguiente cual es la mejor función para nuestro sistema.

Otro método que se explica en este capítulo es la validación cruzada. Será usada en los casos en que se dispone de pocos datos para poder llevar a cabo una predicción acertada de una serie temporal determinada. Consiste en tratar de mejorar las predicciones dividiendo el conjunto de patrones en diversos subconjuntos y usar todos ellos tanto para el proceso de entrenamiento como para la validación. Así se puede obtener un valor de adecuación más acertado de cada individuo. Para este nuevo método nos apoyaremos en la teoría de la sabiduría de los grupos (*"Ensembles"*) comentada en la sección 2.11 del estado del arte. Veremos también una versión de validación cruzada con pesos.

En este capítulo se presenta además el primero de los puntos de acción más importante propuestos en esta tesis doctoral. Este punto propone usar diferentes técnicas de CE para llevar a cabo la búsqueda a nivel global dentro de nuestro sistema híbrido. Para ello, y como podemos observar en la figura 3.1, se estudiarán los algoritmos de CE empleados para la búsqueda global. Después de haber estudiado en detalle otros algoritmos evolutivos, se optó por la posibilidad de cambiar el AG usado en ADANN por otros dos, Evolución Diferencial (ED) y Algoritmo de Estimación de Distribución (AED). La razón de esta nueva propuesta fue la existencia de diferentes estudios previos usando estos algoritmos que presentaban buenos resultados. En el caso del algoritmo ED, estos estudios estaban también orientados al dominio de la predicción de series temporales [64, 65]. Sin embargo, aunque los trabajos basados en AED también obtenían resultados prometedores [60], estos están basados en el dominio de clasificación. No ha sido posible encontrar ningún trabajo previo donde se haga uso de AED para diseñar RNA con el objetivo de predecir series temporales. Esta fue la última y más importante razón por la cual finalmente hicimos uso de estos algoritmos evolutivos.

A continuación, se presentan dos mejoras importantes llevadas a cabo sobre ADANN que darán lugar a una nueva versión del sistema denominada AEDRNA. En ella, y por razones que se detallarán en este capítulo, se pasará a hacer uso de AED como método evolutivo y el algoritmo de aprendizaje de retropropagación será sustituido por RPROP (figura 3.1).

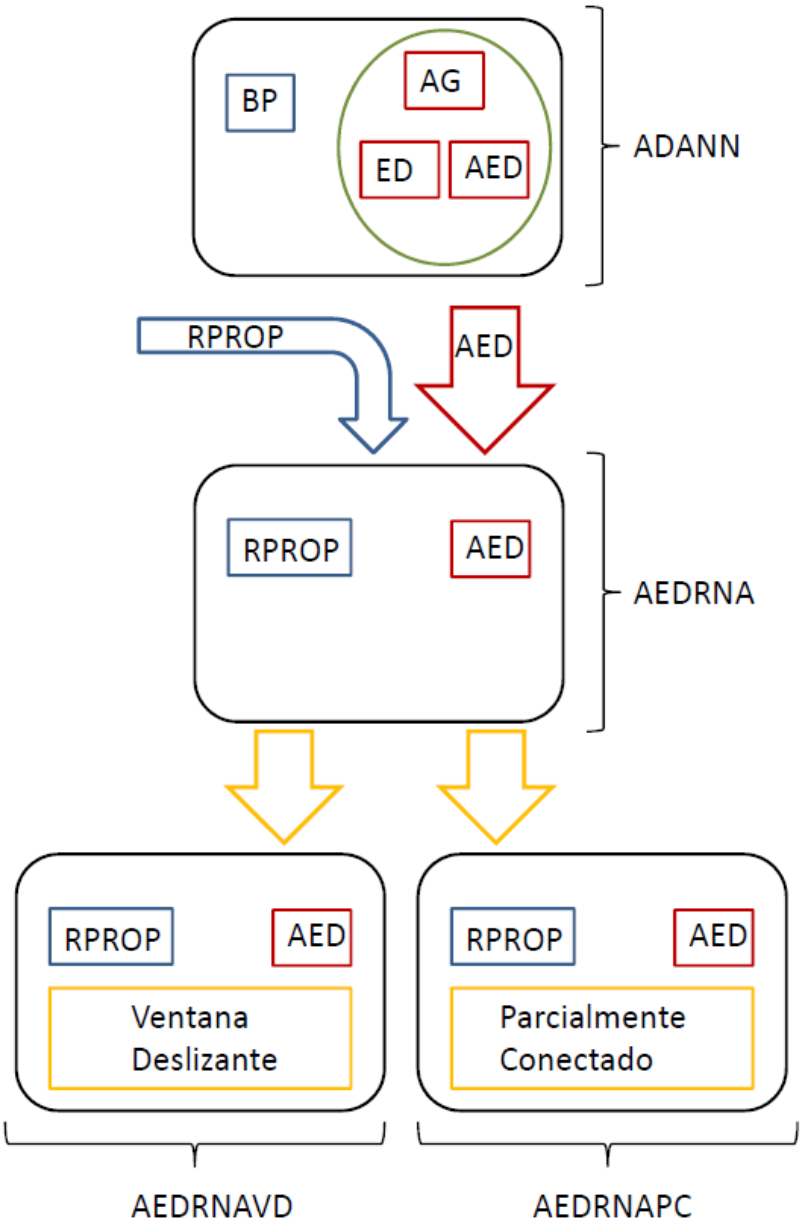


Figura 3.1: Esquema de evolución del sistema.

En este capítulo se detalla también otro tema clave propuesto en esta tesis doctoral. Éste propone profundizar en el diseño de la arquitectura de las RNA utilizadas durante el proceso de búsqueda desde dos puntos de vista diferentes.

El primero, estará orientado a los datos de entrada y cómo la selección de éstos puede afectar al número de nodos de entrada de la RNA. Para ello se hará uso de una ventana deslizante que especificará al sistema AEDRNA que instantes previos de la serie temporal deberá tener en cuenta y cuales no, dando como resultado el sistema en su versión denominada AEDRNAVD (figura 3.1). En este caso, al igual que venía ocurriendo en AEDRNA, se seguirá incluyendo en el genoma cuántos *n* instantes anteriores se consideran para especificar los nodos de entrada en la topología de la RNA y generar los subconjuntos de patrones pertinentes. Sin embargo, la diferencia primordial con respecto a AEDRNA consiste en que se va a realizar una búsqueda más exhaustiva de qué instantes exactos de esos *n* serán considerados como entradas de la RNA y cuales no. Esto se implementará mediante una nueva parte del genoma a la que llamaremos *"time lags"* o ventana deslizante binaria.

El segundo consistirá en hacer uso de una arquitectura diferente a la usada en AEDRNA donde se tomaba como base el perceptrón multicapa totalmente conectado. En esta ocasión se hará uso del perceptrón multicapa parcialmente conectado, con lo que algunas conexiones entre nodos podrán desaparecer. Para ello habrá que codificar dentro del cromosoma qué conexiones se llevan a cabo entre nodos y cuales no. Esto se hace ampliando el cromosoma para poder representar una codificación directa con una parte binaria en la que un *"1"* representa que se lleva a cabo una conexión y un *"0"* significa que no existe tal conexión. Como podemos observar en la figura 3.1, esta nueva versión del sistema será denominada AEDRNAPC.

3.1. Competiciones de Series Temporales

La investigación llevada a cabo en esta tesis doctoral estuvo inicialmente motivada por las competiciones de predicción de series temporales NN3 (2007), NN5 (2008) y NNGC1(2009) [116]. Dichas competiciones consistían en un conjunto de series temporales cuyos datos son conocidos por los competidores. El objetivo de la competición era desarrollar un método único de IC para predecir los *n* valores próximos, y por tanto desconocidos, de cada una de las series temporales con la mayor precisión posible, es decir, obteniendo predicciones lo más cercanas posibles a los valores reales. En nuestro caso, para cada serie temporal se obtendrá una RNA específica, aplicando siempre el mismo método.

En la competición NN3, no existía conocimiento previo alguno sobre la procedencia de los datos y por lo tanto de sus posibles ciclos o periocidad. Cabe destacar además, que no se lleva a cabo ningún análisis previo de los datos de las series temporales expuestas en los problemas para poder aplicar este método a cualquier tipo de serie temporal. En NN3, cada serie temporal tenía entre 125 y 230 valores conocidos para el competidor, y éste tenía que predecir los 19 valores desconocidos siguientes. Por otro lado, en la competición NN5 los competidores conocían de antemano que los datos conocidos de las series temporales representaban la cantidad diaria de dinero obtenido de diferentes cajeros automáticos distribuidos por todo Reino Unido y medidos durante 2 años (730 elementos por serie temporal). En este caso, la competición exigía predecir los 56 valores desconocidos siguientes. En la última de ellas (NNGC1), los datos provenían de medidas diarias, semanales, mensuales y anuales de medios de transporte como pueden ser, el número de aviones que aterrizan en un aeropuerto en un día o cuántos coches pasan por un túnel en una semana. Las series de esta última competición tenían tamaños muy dispares, entre 23 y 730 elementos.

3.2. Generación del Conjunto de Patrones

Como se indica en [21], el problema de predecir series temporales es considerado como la obtención de la relación que tienen los valores del instante temporal *t* con respecto a los valores dados por los instantes anteriores. En este sistema todas las RNA resultantes tan solo necesitarán una neurona de salida (como explicaremos más adelante) y los valores de los elementos previos de esta misma serie temporal $(y_{t-1}, y_{t-2}, \ldots, y_{t-k})$ para obtener una función como se muestra en 3.1:

$$\hat{y_t} = f(y_{t-1}, y_{t-2}, \ldots, y_{t-k}) \tag{3.1}$$

donde $\hat{y_t}$ denota la predicción estimada por la RNA (f), y k el número de instantes anteriores (y por lo tanto el número de nodos de entrada de esta RNA).

Para poder obtener una RNA que permita predecir los futuros valores de una serie temporal, se debe llevar a cabo otro paso importante con los valores ya normalizados de esta serie. Un vez que sus elementos han sido normalizados en el rango [0,1] (dando lugar a los valores n_t), se deben generar los conjuntos de patrones a partir de estos valores normalizados.

Para esta primera aproximación se han considerado RNA con una única neurona de salida (i.e. predicción *1 a N* en adelante) [174] porque si se permitía a las RNA tener varios nodos (*n*) en la capa de salida, la tarea de predecir se llevaría

a cabo para varios valores futuros a la vez (i.e. predicción 1 en adelante). Esta última manera de predecir (predicción 1 en adelante) nos llevaría a predecir no solo el valor y_t a partir de $y_{t-k}, \ldots, y_{t-2}, y_{t-1}$ sino a predecir también los valores $y_{t+1}, y_{t+2}, \ldots, y_{t+n}$ partiendo sólo de $y_{t-k}, \ldots, y_{t-2}, y_{t-1}$. Por lo tanto en cada paso de predicción, se estaría perdiendo información previa y quizás importante para $y_{t+1}, y_{t+2}, \ldots, y_{t+n}$. Por ejemplo, si disponemos de una RNA con tres nodos de salida y predecimos el valor y_{t+2}, al estar prediciendo a la vez y_t y y_{t+1}, y_{t+2} pierde la información que los dos datos anteriores conocidos pudieran aportarle a su propia predicción. Por ello, consideramos mejor predecir los valores futuros de la serie temporal de uno en uno (predicción *1 a N* en adelante).

Así pues, los valores normalizados de la serie temporal serán transformados en un conjunto de patrones, dependiendo de los k nodos de entrada de una RNA específica y cada patrón consistirá en:

- k valores de entrada, que corresponden a los k valores normalizados previos: $n_{t-1}, n_{t-2}, \ldots, n_{t-k}$.

- Una neurona de salida n_t que corresponde al valor objetivo que se quiere predecir.

Se generará pues un conjunto de patrones para cada RNA y éste se usará para entrenar y validar dicha RNA cada vez que el sistema transforme la información dada por un genotipo en un fenotipo durante el proceso evolutivo. Por lo tanto, el conjunto total de patrones obtenido para cada RNA se dividirá en dos subconjuntos, entrenamiento y validación. Este conjunto total de patrones estará ordenado de la misma manera que lo está la serie temporal. El primer *x %*, donde *x* es un parámetro del sistema (**Prct_tr**), del conjunto total de patrones se usará para generar el subconjunto de entrenamiento con el que entrenaremos a la RNA (en nuestro caso *70 %*). El subconjunto de validación se obtendrá del resto de patrones que queden en el conjunto total de patrones. Entonces, el subconjunto de entrenamiento y el subconjunto de validacion serán disjuntos.

Cabe destacar que el subconjunto de test estará formado por los valores futuros de la serie temporal y por lo tanto desconocidos para la RNA y para el proceso de búsqueda que supone el evolutivo. Un ejemplo del proceso descrito haciendo uso de una RNA con tres nodos de entrada se puede ver en la figura 3.2.

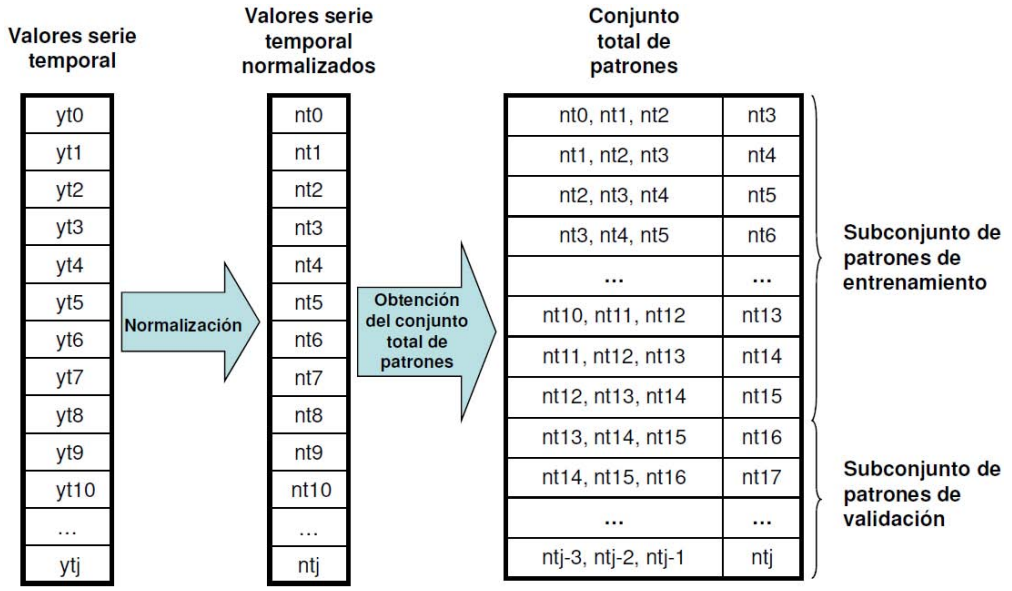


Figura 3.2: Proceso de obtención, a partir de los valores de una serie temporal, de los subconjuntos de entrenamiento y validación para una RNA con tres entradas.

3.3. Función de Evaluación

Como ya comentamos en la sección 3.2, se ha hecho uso de dos conjuntos de patrones para llevar a cabo el proceso de aprendizaje. Estos son, el subconjunto de entrenamiento, usado para modificar los pesos de las conexiones, y un subconjunto de validación para evitar que se diera un posible caso de sobreaprendizaje y medir la capacidad de generalización de los posibles candidatos.

A la hora de especificar como calcular el valor de adecuación de cada individuo o RNA, se ha tenido en cuenta el estudio llevado a cabo por Schaffer et al. [123] donde se afirmaba que aquellas funciones de adecuación en las que se usaba el error de validación en lugar del de entrenamiento obtenían mejores resultados. Esto se ha podido corroborar experimentalmente como se podrá ver más adelante en los resultados mostrados en la sección 4.2.2. Por ello, nuestra función de evaluación será el mínimo error de validación obtenido a lo largo del proceso de aprendizaje como se puede observar en la ecuación 3.2.

$$función\ de\ evaluación\ =\ error\ mínimo\ de\ validación \tag{3.2}$$

Respecto al uso del *Error Cuadrático Medio* (del inglés MSE) como método para medir el error dado por la función de evaluación, nuestro objetivo principal es la reducción de errores extremos que pueden afectar a predicciones de varios pasos

adelante. Además, se llevó a cabo una experimentación preliminar para comprobar que el valor de adecuación medido con MSE generaliza mejor y lleva a mejores predicciones que los llevados a cabo con el *Error Medio Absoluto* (EMA).

Por ello, se plantearán dos nuevas funciones de evaluación. En estas nuevas funciones de evaluación, se hará uso de un nuevo conjunto de patrones. Este nuevo conjunto no se utilizará ni para el proceso de aprendizaje, ni para detener dicho proceso de aprendizaje, estableciendo así el fenotipo.

Tan sólo para obtener el valor de adecuación como una combinación lineal del error de validación obtenido con el subconjunto de validación y el error de este nuevo subconjunto de patrones adicionales. Ese nuevo subconjunto se conocerá como subconjunto de validación II.

Se estudiará si el uso de este nuevo subconjunto validación II dentro de la función de evaluación permite obtener una mejor predicción. Para llevar a cabo esta tarea, y como se acaba de indicar, ahora se utilizarán tres subconjuntos:

- entrenamiento
- validación
- validación II

Recordemos que el subconjunto de test estará formado por los valores futuros de la serie temporal y por lo tanto desconocidos para la RNA y será éste el conjunto a predecir por el sistema.

Los subconjuntos de entrenamiento y validación siguen teniendo la misma funcionalidad y el nuevo subconjunto validación II será empleado para complementar la función de evaluación.

Este conjunto de validación II se obtendrá del mismo modo que se hizo en la sección 3.2, del total de los patrones ordenados establecidos. Se tomarán los primeros $x\,\%$ (donde x es un parámetro del sistema representado por **Prct_tr**) patrones para generar el subconjunto de entrenamiento, los próximos $y\,\%$ (siendo y otro parámetro del sistema conocido como **Prct_valid**) para formar el subconjunto de validación y el resto de patrones para el subconjunto validación II (figura 3.3).

El porcentaje (o parámetros del sistema **Prct_tr** y **Prct_valid**) usado para dividir el conjunto total de patrones generando así estos tres subconjuntos, dependerá de si se dispone de alguna información previa sobre la serie temporal. Si no existe información previa alguna sobre los datos de la serie temporal, tan sólo los

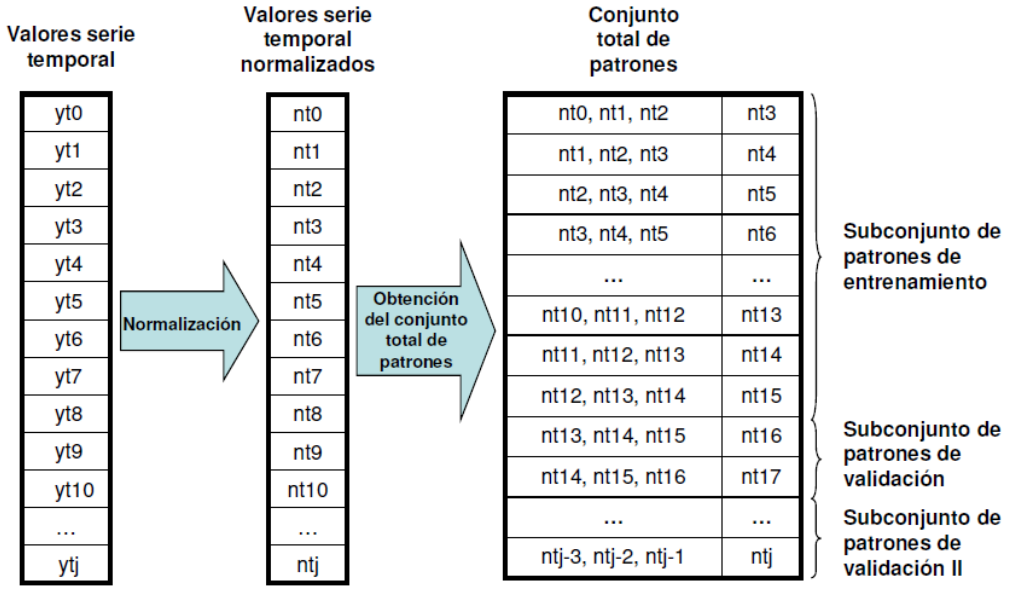


Figura 3.3: Proceso para obtener los subconjuntos de entrenamiento, validación y validación II, para el caso de una RNA con tres nodos de entrada.

valores, los subconjuntos de entrenamiento, validación y validación II serán, el *70 %*, *15 %* y *15 %* respectivamente del conjunto total de patrones.

Pero si hubiera alguna información previa sobre qué representan los datos de la serie temporal, entonces la medida podría tener una periodicidad inherente de días, semanas, meses o años, etc. (relacionado con eventos naturales o actividades humanas) como ocurre en la serie Passengers, donde se mide el número de pasajeros de una aerolínea mensualmente durante varios años. Si éste es el caso y hubiera cualquier periocidad en los datos, el mapeado de entrada salida será aprendido mejor si esa periocidad no se rompe, es decir, si los datos de entrenamiento corresponden a un número entero de horas, días, semanas, meses o años.

Lo mismo ocurre con la estimación de la capacidad de generalización dada por el subconjunto de validación. Esa estimación será más representativa si los datos de validación y de validación II también corresponden a un número entero de semanas, meses o años. En este caso, deberemos seleccionar unos valores para los parámetros de partición de los subconjuntos de patrones de acuerdo con la periocidad dada por la serie temporal.

Una vez que se han generado estos tres subconjuntos, el proceso de aprendizaje será alterado ligeramente. De nuevo se entrena la RNA y se guarda la arquitectura de dicha RNA cuando el error de validación es mínimo durante el proceso de entrenamiento. Pero ahora, el error del subconjunto de validación II se obtiene

una vez que el entrenamiento ha terminado. Así, el subconjunto de validación II no interviene, en ningún modo, en el proceso de entrenamiento o aprendizaje de la RNA. También ahora, no sólo el error de validación será usado como valor de adecuación sino que también se usará el de validación II, ya que la nueva función de evaluación consistirá en la combinación lineal del error del subconjunto de validación y el de validación II.

Se debe decidir cuanto peso se quiere asignar a los errores de validación (α) y validación II (β). Para ello se hará uso de dos parámetros más del sistema denominados **Prct_Fitness_Valid** y **Prct_Fitness_ValidII** que corresponderán a (α) y (β) respectivamente. En este trabajo, se han seleccionado tres pares distintos de coeficientes para dichos errores durante la experimentación:

- $\alpha = 1{,}0$ y $\beta = 0{,}0$
- $\alpha = 0{,}5$ y $\beta = 0{,}5$
- $\alpha = 0{,}0$ y $\beta = 1{,}0$

La ecuación 3.3 representa como quedaría la nueva función de evaluación.

$$función\ de\ evaluación\ =\ (\alpha \cdot valid\ error) + (\beta \cdot valid\ II\ error) \quad (3.3)$$

3.4. Teoría de Conjuntos o *"Ensembles"*

La sabiduría de los grupos [171] o *"Ensembles"* es utilizada con frecuencia para mejorar el rendimiento de los modelos resultantes para las tareas de clasificación y regresión, ya que se observó que un conjunto de individuos predictores se comporta mejor que un único individuo predictor, de media [175, 176]. Un *"Ensemble"* consiste entonces en un conjunto de modelos tomados de una sola clase, en este caso RNA.

Al hacer uso de un *"Ensemble"*, es necesario responder a dos preguntas:

- ¿Cómo se genera el *"Ensemble"*?
- ¿Cómo se combinan los resultados del *"Ensemble"* en un único resultado?

Para responder a la primera pregunta debemos recordar que como se comentó en la sección 3.6.1, nuestra aproximación tan solo hace uso de la mejor RNA de

la última generación, ignorando toda la información aportada por otras RNA. Sin embargo, una población de RNA contiene más información que cualquier RNA tomada individualmente de dicha población. Esta información puede ser usada para mejorar el rendimiento y la fiabilidad. Se han realizado diversos estudios sobre combinación de RNA para mejorar la capacidad de generalización y la fiabilidad [177, 178, 179]. Para maximizar el efecto de combinar múltiples RNA sería preferible seleccionar aquellas con gran diversidad, ya que no existe mucha ventaja en combinar RNA que generalizan casi por igual [176]. Necesitamos entonces un conjunto de RNA que generalicen bien obteniendo un error mínimo, debiendo además presentar algún grado de diversidad [180, 181, 19].

Otro dato a tener en cuenta es el número de individuos que se seleccionan. Yao propone en [130], donde trabaja también con RNAE, seleccionar todos los individuos de la última generación. En este trabajo, se ha optado por no llevar a cabo la selección realizada por Yao ya que en ocasiones, algunos individuos se repiten o son muy similares. Además, aquellos individuos peores de la última generación también son excluidos de la selección para escoger tan solo a los más precisos. Por lo tanto seleccionaremos aquellos más precisos y con diferentes topologías, es decir, aquellos con más diferencia entre número de nodos en la capa de entrada y en la oculta [182].

Por otro lado, como podremos observar más adelante en la sección 3.5, al hacer uso de la validación cruzada, un único individuo estará representado por tantas arquitecturas como número de subconjuntos se hayan utilizado. En este caso específico, es la validación cruzada en sí la que nos aportará las *n* RNA que pasarán a formar el *"ensemble"*.

Una vez que ya conocemos el proceso para seleccionar las RNA que formarán parte del *"Ensemble"* debemos responder a la segunda pregunta, ¿cómo combinamos los resultados?. Existen varias maneras de combinar dichas predicciones para obtener una única. La más sencilla sería llevar a cabo la media de todas ellas, otro modo es calcular la mediana en lugar de la media.

Una manera de considerar las diferencias existentes entre individuos sin involucrar un excesivo coste computacional extra es usar el valor de adecuación para computar un peso para cada individuo. Más tarde este peso se tendrá en cuenta para calcular la combinación de predicciones, dando una mayor importancia a aquellas predicciones con un peso más elevado. En esta tesis doctoral se plantean dos métodos de combinación teniendo en cuenta los valores de evaluación de los individuos. El primero consiste en calcular los pesos de manera exponencial como muestra la ecuación 3.4. Donde f_i representa el valor de adecuación de ese individuo específico y *max* y *min* se refieren al máximo y mínimo de todos los inversos de los valores de evaluación, es necesario calcular el inverso ya que nuestro valor

de adecuación es mejor cuanto menor es).

$$\begin{aligned} min &= \frac{1}{f_{min}} \\ max &= \frac{1}{f_{max}} \\ peso_i &= exp(\frac{\frac{1}{f_i} - min}{max - min}) \end{aligned} \tag{3.4}$$

La segunda opción fue presentada por Yao en [130] y consiste en un método de *Combinación Lineal Basado en Rango* (CLBR). En ella la salida o predicción final depende de una combinación lineal de las predicciones de los individuos seleccionados, de tal forma que la respuesta de cada RNA es pesada proporcionalmente a su valor de adecuación. Dadas *n* arquitecturas ordenadas por valor de adecuación con una tasa de error cada vez mayor, donde *n* es el tamaño de la población seleccionada, β es un factor escalar (establecido con experimentación preliminar), β_j el valor de adecuación de cada RNA y sus salidas $salida_1, salida_2, \ldots, salida_n$, entonces el peso para la arquitectura i-ésima es el dado por la ecuación 3.5.

$$peso_i = \frac{exp(\beta \cdot (n + 1 - i))}{\sum_{j=1}^{n} exp(\beta_j)} \tag{3.5}$$

Y la predicción final consiste en la combinación de las predicciones de cada arquitectura con su peso correspondiente como se puede observar en la ecuación 3.6.

$$salida_{Ensemble} = \sum_{j=1}^{n} peso_j \cdot salida_j \tag{3.6}$$

El objetivo de esta sección es presentar varias aproximaciones de *"Ensembles"* y estudiar cual de ellas es la más óptima para nuestro dominio.

3.5. Validación Cruzada

La validación cruzada, a veces conocida como estimación de rotación o *"cross-validation"*, es una práctica estadística que consiste en dividir una muestra de datos en subconjuntos de tal modo que el análisis se realiza inicialmente en uno de ellos, mientras que el resto de subconjuntos son retenidos para su uso posterior en la confirmación y validación de este análisis inicial. Es una técnica muy utilizada

en inteligencia artificial para validar los modelos generados a partir de un conjunto de datos o muestra. Se utiliza principalmente en entornos donde el objetivo es la predicción [183], y se quiere estimar con mayor precisión cómo de preciso se comportará un modelo predictivo en la práctica.

Una ronda de validación cruzada implica dividir una muestra de datos en subconjuntos complementarios, realizando el análisis en un subconjunto (denominado el conjunto de entrenamiento), y validando el análisis sobre el otro subconjunto (llamado conjunto de validación). Para reducir la variabilidad, se realizan múltiples rondas de validación cruzada utilizando particiones diferentes. Al final se calcula la media aritmética de las validaciones obtenidas para conseguir el ratio de error para la muestra final.

La validación simple consiste en dividir en dos conjuntos complementarios los datos de la muestra, aunque de esta manera se suele sobrestimar el modelo construido, es por ello que optamos por la validación cruzada.

La elección del número de conjuntos a dividir los datos dependerá del tamaño y características de la muestra.

La validación cruzada es importante a la hora de evitarnos la comprobación de hipótesis sugerida por los datos llamados *"errores de Tipo III"* [184], especialmente cuando las muestras son más costosas o imposibles de recoger con lo que no hay demasiadas muestras (i.e. datos de la serie temporal) [183]. Esto ocurre en la sección 4.2.3, donde las series temporales Passengers, Temperature y Dow-Jones tienen pocos elementos.

La validación cruzada se ha utilizado en varios trabajos anteriores para predecir series temporales [185, 186]. En el estudio llevado a cabo por Simo et al. [185], se mide la densidad de los términos de ruido al aplicar el filtro de *Kalman*, una herramienta estadística clásica, para predecir series temporales. Por otro lado, en [186] la validación cruzada se utiliza para validar el proceso de formación de RNA para predecir series de temporales.

En este estudio, el conjunto total de patrones se divide entre un número n de subconjuntos de patrones (de 2 a 8). Cada vez que uno de estos subconjuntos de patrones se utiliza para validar, el resto de los subconjuntos (n-1) será usado para entrenar la RNA. Este subconjunto de validación nunca será usado para entrenar la RNA en este turno y nunca se superpondrá con otros subconjuntos como ocurre en [186]. Pero en la siguiente ronda este subconjunto será parte del subconjunto de patrones de entrenamiento y uno anteriormente usado para entrenar la RNA se utilizará como subconjunto de validación. Este proceso se repetirá tantas veces como subconjuntos de patrones tengamos. La figura 3.4 muestra un ejemplo de validación cruzada con cinco subconjuntos de patrones.

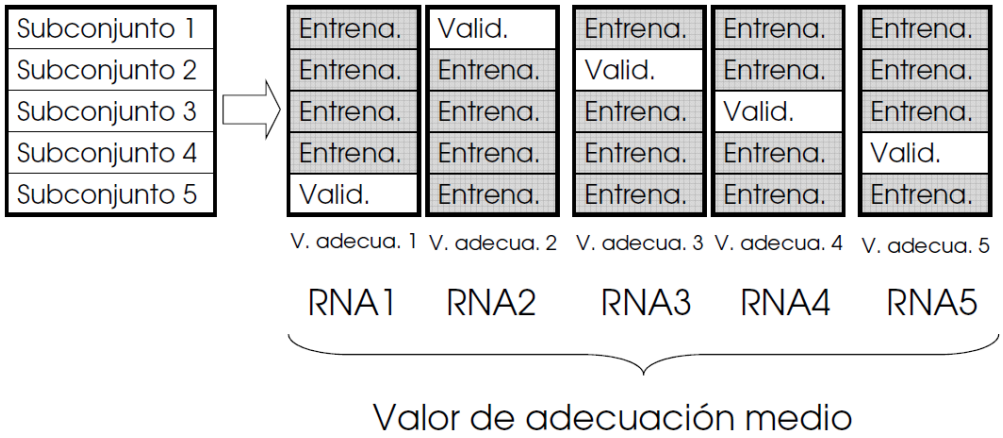


Figura 3.4: Ejemplo de validación cruzada con 5 subconjuntos de patrones.

Dependiendo del número *n* de subconjuntos con los que se lleve a cabo la validación cruzada y aunque en un principio en nuestro sistema un individuo es representado por una única RNA, cuando aplicamos validación cruzada a dicho individuo, éste obtiene como resultado *n* RNA y *n* valores de adecuación diferentes (un valor de adecuación para cada una de las RNA).

Esto ocurre ya que la misma RNA con una topología dará lugar a una arquitectura diferente (arquitectura = topología + pesos de conexiones) dependiendo de qué patrones son usados para entrenar y cuáles para validar. Puesto que en cada turno los subconjuntos usados para entrenar y validar cambian, también lo hará la arquitectura dada por el individuo como resultado.

Entonces, después de aplicar la validación cruzada a un individuo y obtener *n* valores de adecuación diferentes para este mismo individuo, el procedimiento más común es pesar de manera equitativa cada valor de adecuación dado, es decir, calcular el valor de aptitud usando el promedio entre todos los valores de adecuación obtenidos con sus arquitecturas resultantes como muestra la figura 3.4. Este valor de adecuación total nos ayudará a hacernos una idea mejor de como se comporta este individuo particular frente al problema de predicción de una serie temporal planteada.

Sin embargo, en el dominio de predicción los modelos recientes (i.e. aquellos que usan los datos más recientes de la serie temporal para ser validados) deben tener una mayor importancia si se compara con los más viejos (i.e. aquellos que usan datos menos recientes de la serie temporal para ser validados). Es por ello que al dividir el conjunto de patrones en subconjuntos, aquellos que son usados como validación y que sus patrones pertenezcan al principio de la serie temporal, tendrán asociados unos pesos menores que si pertenecieran al final de la mis-

ma. Siguiendo esta hipótesis, en este trabajo, proponemos la siguiente validación cruzada ponderada expresada en la ecuación 3.7.

$$W_j = \frac{1}{2^{n+1-j}},\ para\ todo\ j \in \{2,\ldots,n\}\ y\ W_1 = 1 - \sum_j W_j. \tag{3.7}$$

Por ejemplo, para *n* = 4, entonces $W_1 = W_2 = 0{,}125$, $W_3 = 0{,}25$ y $W_4 = 0{,}5$.

Pero cuando se alcanza la última generación del proceso global de búsqueda, su mejor individuo se debe usar para predecir los valores futuros y por lo tanto desconocidos y se hará uso de los *"Ensembles"* explicados en la sección anterior.

3.6. Algoritmos Evolutivos

Los Algoritmos Genéticos (AG) son métodos adaptativos utilizados para solventar problemas de búsqueda y optimización. Basados en el proceso genético de los organismos vivos, donde las poblaciones evolucionan en la naturaleza de acuerdo con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin [49].

Por imitación de este proceso, los AG son capaces de generar soluciones para problemas del mundo real. Los principios básicos de los AG fueron establecidos por Holland [50] en 1975, y se encuentran bien descritos en varios textos [19, 187].

La Evolución Diferencial (ED) es un algoritmo perteneciente a la clase de computación evolutiva, desarrollado y propuesto por Storn y Price [188, 61].

ED se utiliza en funciones reales multidimensionales, pero no emplea el gradiente del problema que se pretende optimizar, lo que significa que ED no requiere que el problema de optimización sea diferenciable como es requerido por los métodos clásicos de optimización, tales como descenso de gradiente. ED por lo tanto puede también ser usado en problemas de optimización que ni siquiera son continuos, son ruidosos, cambian con el tiempo, etc.

El énfasis principal de diseño es la optimización de parámetros reales. ED se basa en un operador de mutación, que añade una cantidad resultante de la diferencia de dos individuos elegidos al azar de la población actual, en contraste con la mayoría de los algoritmos evolutivos, en los que la mutación se realiza a través de un variable aleatoria.

El principal componente del algoritmo es el operador de sobrecruzamiento, que trabaja sobre la diferencia ponderada de los genes de dos individuos aleato-

rios, añadida al mismo gen de un tercer individuo elegido al azar, como se explicó con anterioridad en la sección 2.5. Este operador se aplica con una probabilidad de *CR* (de manera similar a la probabilidad del cruce en un AG, definido por el usuario), pero en este caso se aplica al menos una vez por cada individuo.

Otra parte importante del algoritmo ED es el operador de selección, que realiza la selección progresivamente durante la generación de los hijos, llevando a cabo tan solo comparaciones locales. La función *"es mejor"* se aplica para probar la condición del posible reemplazo de los padres por los hijos. Estas comparaciones locales hacen al algoritmo más eficiente, evitando la necesidad de clasificación o ranking de toda la población.

Los autores del algoritmo ED han sugerido que calculando la diferencia entre dos individuos elegidos al azar de la población, lo que el algoritmo hace en realidad es la estimación del gradiente en esa zona [188]. Esta teoría también constituye una manera más eficiente para autoadaptar el operador de mutación, de hecho, ED también es capaz de autoadaptar ambos, el tamaño y la dirección del paso, ya que ambos dependen de las soluciones actuales en la población [189].

En la literatura existen pocos trabajos donde ED sea aplicado al campo de predicción de series temporales [65, 64]. Por ello, aquí presentamos un método automático que hace uso de las ventajas de ED y de las RNA (es decir, un sistema híbrido), en el cual el usuario no necesita ser un experto para predecir todo tipo de series temporales.

Por otro lado, en los últimos años se ha desarrollado un nuevo algoritmo evolutivo llamado Algoritmo de Estimación de Distribución (AED) [53]. Los AED, también conocidos como *Algoritmos Genéticos de Modelo Probabilístico* (AGMP), son el resultado de una modificación de los AG.

Basado en AG, este algoritmo sustituye los operadores de cruce y mutación por el aprendizaje y muestreo de distribuciones de probabilidad de una selección de los mejores individuos de la generación anterior.

Mientras que en los algoritmos heurísticos de la CE las dependencias que se dan entre los genes de cada individuo se tienen en cuenta de manera implícita, en los AED estas relaciones se expresan explícitamente a través de las distribuciones de probabilidad asociadas al conjunto de individuos seleccionados en cada iteración.

Se genera entonces una primera población de individuos de forma aleatoria y a partir de ésta, la población de soluciones candidatas se crea a partir de la distribución de probabilidad obtenida de los mejores individuos de la generación anterior. Cabe destacar que los mejores individuos de una generación pasan a la siguiente directamente (elitismo).

Dado que la población no se regenera a partir de individuos, sino desde las distribuciones de probabilidad obtenidas, no existen operadores de cruzamiento ni de mutación, por lo tanto éstos no necesitan ser parametrizados como ocurre con los AG. Además, al no existir sobrecruzamiento evitamos el problema de separación dentro del cromosoma de los valores (pesos) asociados a cada conexión que llegan a un mismo nodo como se explicó en la sección 2.9.1

Sin embargo sí que seguiremos usando el mismo número de individuos por generación y el número máximo de generaciones, como criterio de parada, que se han usado hasta ahora con los AG.

Se encuentran en la literatura un reducido número de estudios híbridos, es decir combinando RNA y AED. Además, éstos han sido aplicados tan sólo a dominios de clasificación [60]. Por ello, aquí presentamos una novedosa propuesta híbrida que hace uso de las ventajas de AED y de las RNA para predecir series temporales.

3.6.1. Diseño de Redes de Neuronas Artificiales mediante Algoritmos Genéticos

Aquí se presenta la propuesta basada en AG, el sistema ADANN.

El problema de diseñar RNA puede ser visto como un problema de búsqueda dentro del espacio formado por todas las posibles RNA. Esta búsqueda se podría llevar a cabo por un AG [190] haciendo uso de sus propiedades de exploración y explotación. Por lo tanto, existen tres cuestiones cruciales a tener en cuenta:

1. El espacio de búsqueda: qué información de la RNA se establece previamente y cuál es incluida en el cromosoma.

2. Cómo se codifica la información incluida en el cromosoma, el esquema de codificación.

3. Qué es lo que se busca, traducido a la función de evaluación .

En esta aproximación (ADANN) se ha optado por usar el perceptrón multicapa como modelo computacional debido a su capacidad de aproximación, Cybenko [86]. Y dentro de este grupo, se ha optado por el perceptrón multicapa totalmente conectado con un única capa oculta y un único nodo en la capa de salida. Como algoritmo de aprendizaje ha sido seleccionado el algoritmo de retropropagación, (BP).

Para llevar a cabo el entrenamiento de las RNA se ha optado por el *Stuttgart Neural Network Simulator* (SNNS) [191], un simulador que provee un entorno flexible para el desarrollo e investigación de aplicaciones de RNA, diseñado en la Universidad de Stuttgart (Alemania). No solo es una herramienta software, si no un proyecto de investigación que desarrolló esta Universidad, a lo largo de varios años. Esta herramienta dispone de un interfaz gráfico en el que se puede diseñar y trabajar con RNA de manera sencilla y rápida. Por otro lado, dispone de una línea de comandos, así como de un conjunto de funciones a las que se puede llamar para llevar a cabo las mismas operaciones de que dispone el interfaz del sistema, entre ellas las básicas que aquí usaremos, generación de RNA y entrenamiento. Al estar programado en C es rápido, computacionalmente hablando, y se puede trabajar con varios tipos de RNA a bajo nivel gracias a las numerosas funciones y opciones de las que dispone. Es por ello que ha sido seleccionado como herramienta para operar con RNA.

Como ya se comentó en la sección 2.9, el diseño de una RNA se lleva a cabo estableciendo los parámetros de dicha RNA. Diferentes estudios han tratado por separado con el diseño de RNA abordando tres aspectos:

- Pesos de las conexiones: Valor para cada conexión presente en una RNA.
- Topología: Número de nodos en cada capa oculta, número de capas ocultas, número de nodos en la capa de entrada y salida.
- Reglas de aprendizaje: Factor de aprendizaje y valores del momento.

En esta tesis doctoral se propone la evolución de las topologías de las RNA, de los pesos de conexión e incluso de las reglas de aprendizaje, en forma de factor de aprendizaje asociado al algoritmo BP, conjuntamente. En este trabajo en el que hacemos uso de RNA perceptrón multicapa con un única capa oculta y retropropagación, lo parámetros elegidos para su codificación en el cromosoma son:

- Números de nodos de entrada.
- Número de neuronas ocultas.
- Patrón de conexiones, o lo que es lo mismo, cómo están conectados los nodos.
- Y por último, el conjunto completo de pesos de las diferentes conexiones entre nodos. Referente a la implementación del valor de los pesos de las

conexiones, se ha optado por emplear la semilla para inicializar dichos pesos de conexión al generar la RNA inicial que representa cada individuo como se verá más adelante.

Una ventaja conocida del esquema de codificación directo es que es fácil de evolucionar RNA de tamaño pequeño [144]. El esquema de codificación directo ha sido adoptado por Whitley et al. [145], por Schaffer et al. [123] y por Maniezzo [144] entre otros. Es por ello que en nuestra aproximación para diseñar RNA, se ha optado por este esquema de codificación directo para poder codificar los perceptrones multicapa de una única capa oculta y totalmente conectados. Entendemos por *gen* la información representada por un dígito decimal. En este esquema, los parámetros de la RNA comentados con anterioridad se codifican en el cromosoma como sigue:

- Dos dígitos decimales (2 genes) para codificar el número de nodos de entrada (i).
- Dos dígitos decimales (2 genes) para el número de nodos en la capa oculta (h).
- Dos dígitos decimales (2 genes) para el factor de aprendizaje (α) del algoritmo retropropagación.
- Y por último, diez dígitos decimales (10 genes) para codificar el valor de la semilla de inicialización de los pesos de las conexiones (s). Debido a que como se comentó anteriormente, se ha optado por el uso de la herramienta SNNS [191] para generar y entrenar las diferentes RNA, y en dicha herramienta, la semilla implementada es del tipo *"long int"* o entero largo. Se ha procedido pues a darle a *"s"* un valor máximo definido por ese tipo, de ahí que se le hayan asignado 10 dígitos decimales para su codificación.

De esta forma, los valores de *"i"*, *"h"*, "α" y *"s"* se obtienen del cromosoma como se indica en la figura 3.5 y en la ecuación 3.8. Como podemos observar en la figura 3.5 y en la ecuación 3.8, para calcular el número definitivo de nodos de entrada que tendrá una RNA específica, se usan dos parámetros aparte de los dos genes asignados para dicha tarea. *NTS* representa el número de elementos de los que dispone la serie temporal con la que trabajamos en ese momento, **max_entradas** es un parámetro del sistema dado por el usuario, que sirve para calcular el número de nodos en la capa de entrada y **max_ocultas** es otro parámetro empleado para calcular los nodos de la capa oculta de la RNA (su valor será siempre el doble del valor dado por **max_entradas**).

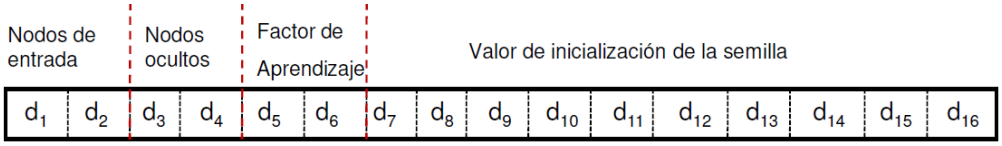


Figura 3.5: Esquema de codificación decimal donde d_j es el n-ésimo dígito decimal (0-9).

$$\begin{aligned} d_j &\in \{0,\ldots,9\} \\ i &= NTS \cdot max\_entradas \cdot \tfrac{(d1\cdot 10)+d2}{100} + 1 \\ h &= NTS \cdot max\_ocultas \cdot \tfrac{(d3\cdot 10)+d4}{100} + 1 \\ \alpha &= \tfrac{(d5\cdot 10)+d6}{100} \\ s &= \textstyle\sum_{j=7}^{16} 10^{16-j} \end{aligned} \tag{3.8}$$

A continuación veremos un ejemplo que explica como funciona la generación de una RNA a partir de un genotipo específico. Suponiendo que abordamos una serie temporal con 125 elementos (*NTS=125*), si tomáramos el número de nodos de entrada y nodos en la capa oculta como $(d_1 \cdot 10 + d_2) + 1$ y $(d_3 \cdot 10 + d_4) + 1$, respectivamente, estaríamos limitando los parámetros de la RNA a un mínimo de 1 nodo de entrada y otro en la capa oculta y un máximo de 100 en ambos. Estos valores puede resultar suficiente para series temporales como la indicada de 125 elementos. Pero si la serie temporal tuviera 2000 elementos, el número de entradas y por lo tanto de elementos máximos previos a usar para predecir un valor futuro estaría muy limitado, al igual que los ocultos. Es por ello que para no aumentar el número de genes y para que el cromosoma no sea dependiente del dominio, hemos optado por usar el número de elementos de la serie temporal (*NTS*) y dos parámetros dados por el usuario. Así si *NTS=125* y *max_entradas=0.4* (entonces *max_ocultas=0.8*), en el caso de que d_1 y d_2 valgan *"0"* nos aseguramos un mínimo de 1 nodo en la capa de entrada y la oculta. Si por el contrario, d_1, d_2, d_3 y d_4 tuvieran un valor de *"9"*, el número de nodos de entrada sería $(125 \cdot 0{,}4 \cdot (0{,}99)) + 1 = 50$ y el de ocultas, $(125 \cdot 0{,}8 \cdot (0{,}99)) + 1 = 100$.

Proceso de Búsqueda Global

El proceso de búsqueda global (AG) consiste de los siguientes pasos:

1. Se genera una primera población aleatoria de individuos (i.e. cromosomas).

2. Obtenemos el fenotipo (i.e. arquitectura de la RNA) para cada uno de los genotipos (i.e cromosomas) de la generación actual. Además, se obtendrá el valor de adecuación asociado a cada individuo. Así pues, para cada individuo de la población:

   a) Se obtiene la topología de este individuo. Para ello se genera una RNA a partir de la información dada por su cromosoma y se le asignan los pesos iniciales dados por la semilla de su cromosoma.

   b) A continuación, se obtienen los subconjuntos de entrenamiento y validación para esta RNA, usando la serie temporal a predecir como se comentó en la sección 3.2.

   c) Dada la topología de la RNA y los valores de los pesos de conexión con la información del cromosoma, ésta es entrenada haciendo uso de SNNS [191].

      El subconjunto de patrones de entrenamiento será usado para entrenar y el de validación para estimar la capacidad de generalización, obteniendo el valor de adecuación de la RNA y parando el proceso de entrenamiento antes de que se produzca sobreaprendizaje en dicho proceso.

      La arquitectura de la RNA cuando el error de validación es mínimo a lo largo del proceso de entrenamiento se guarda (i.e. *"early stopping"*). Dicha arquitectura almacenada será el fenotipo final del individuo y el error de validación pasará a ser el valor de adecuación final del individuo.

3. Al terminar de calcular los valores de adecuación de todos y cada uno de los individuos de la población actual, los operadores genéticos elitismo, selección, sobrecruzamiento y mutación se aplican con el objetivo de generar los individuos de la próxima generación, un conjunto nuevo de cromosomas como se explicará más adelante.

4. Los pasos 2 y 3 son ejecutados iterativamente hasta que se alcanza el criterio de parada (un número máximo de generaciones).

Un esquema del proceso completo se puede observar en la figuras 3.6 y 3.7. Mientras en la figura 3.6 se muestra un esquema completo del sistema, en la figura 3.7 se muestra el proceso para obtener los conjuntos de patrones generados para cada individuo y su correspondiente valor de adecuación.

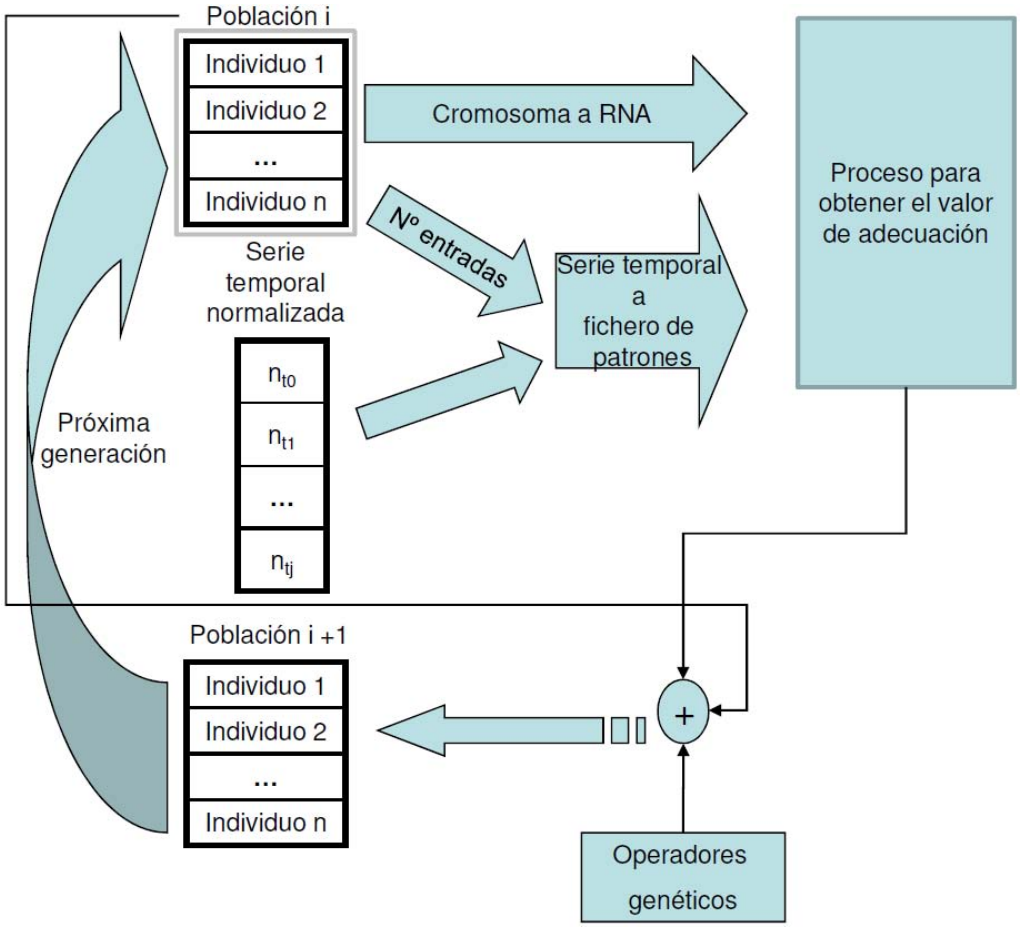


Figura 3.6: Esquema de diseño de RNA mediante AG. Sistema ADANN

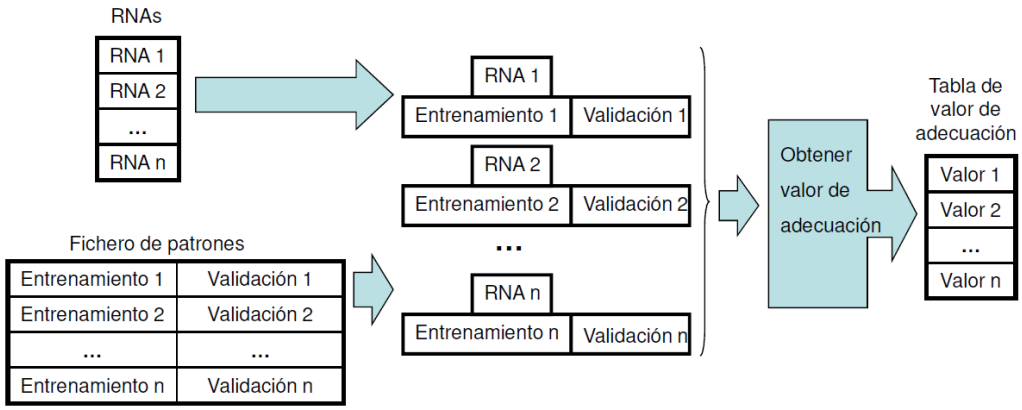


Figura 3.7: Proceso para obtener el valor de adecuación.

Función de Evaluación

Como ya indicamos con anterioridad en la sección 3.3, se ha optado por la función de evaluación en la que el subconjunto de validación tiene todo el peso a la hora de obtener un valor de generalización. Por ello, nuestra función de evaluación será el mínimo error de validación obtenido a lo largo del proceso de aprendizaje como se puede observar en la ecuación 3.2.

Parámetros del Algoritmo Genético

En cuanto al AG, se tuvieron que fijar los valores de algunos parámetros para llevar a cabo la experimentación. Para preseleccionar los valores de estos parámetros, se tuvieron en cuenta los estudios de Goldberg sobre AG [19] y Marek [192] además de la experiencia obtenida con la experimentación preliminar llevada a cabo. Como ya comentamos anteriormente los parámetros del AG a considerar son:

- Tamaño de la población. Número de individuos en cada generación, este parámetro se estableció en *50* para dar un margen suficiente de individuos a la hora de llevar a cabo la búsqueda global. Como se explica en [193] si este parámetro no es incluido en el proceso de búsqueda, la mejor solución es darle un valor lo suficientemente alto a costa de tiempo computacional en la experimentación.

- Elitismo. Parte porcentual de individuos que pasan directamente de una generación *"i"* a la siguiente *"i+1"* sin ser modificados. En nuestro sistema, seleccionamos el *10 %* de individuos con mejor valor de adecuación.

- Sobrecruzamiento. El punto o gen seleccionado como punto de corte para formar dos descendientes a partir de dos progenitores. Se elige el punto de corte aleatoriamente para ambos progenitores. Una vez divididos los progenitores en dos partes cada uno, se obtienen los descendientes intercambiando la parte izquierda del primer progenitor con la derecha del segundo y la parte derecha del primer progenitor con la parte izquierda del segundo.

- Mutación. Porcentaje de probabilidad de que un gen sea modificado de su valor actual a otro valor. Se calcula como uno dividido entre la longitud del cromosoma (*1/longitud cromosoma*, en nuestro caso $1/16 = 0{,}07$). La probabilidad de mutación se aplicará a cada uno de los genes del cromosoma de cada individuo.

- Número de generaciones. Este número indicará cuantas generaciones deben ejecutarse antes de que el proceso de búsqueda alcance el criterio de parada. En nuestro caso se considera *100* generaciones.

La tabla 3.1 muestra un resumen de los valores dados a los parámetros del sistema.

Tabla 3.1: Parámetros seleccionados para el sistema.

Parámetros	Valores
Tamaño población	50
Elitismo	10 %
Mutación	0,07
Nº generaciones	100

Predicción de las Series Temporales

Una vez que el proceso de búsqueda llevado a cabo por el AG finaliza, el mejor individuo (i.e. RNA con mejor valor de adecuación) de la última generación se emplea para predecir los *n* futuros valores desconocidos ($y_t, y_{t+1}, \ldots, y_{t+n}$) uno a uno, usando los k valores previos conocidos de la serie temporal ($y_{t-k}, \ldots, y_{t-2}$, y_{t-1}) donde k representa también el número de nodos de entrada de la RNA seleccionada.

Para llevar a cabo esta última tarea, la predicción los valores futuros, tan solo será necesaria la arquitectura almacenada de la RNA seleccionada al final del proceso de búsqueda y un archivo de patrones de formato compatible con SNNS [191] donde se incluirán los k valores previos necesarios como entrada y una salida cualquiera. Para predecir varios valores consecutivos ($y_t, y_{t+1}, \ldots, y_{t+n}$), cada vez que un nuevo valor y_t es predicho, éste es incluido en orden en el conjunto de valores de entrada anteriores y se usará para predecir el próximo y_{t+1} (predicción *1 a N* en adelante).

3.6.2. Diseño de Redes de Neuronas Artificiales mediante Algoritmo Evolución Diferencial

Como se comentó en la sección 2.5, ED a diferencia de los AE añade la diferencia pesada entre dos vectores de la población a un tercero, con lo que la distribución

de probabilidad por separado no tiene que ser utilizada, lo que hace este esquema completamente organizado. Es por ello por lo que se ha optado en esta sección por presentar un sistema en el que los AG se sustituyen por ED, para mejorar el rendimiento.

Existen varios esquemas de generación de individuos usando ED [194]. En este documento, se presenta el esquema más popular, ED/rand/1/bin, el cual se recomienda como primera opción cuando se trata de aplicar evolución diferencial a cualquier problema dado [195, 194]. A continuación podemos observar, en pseudo-código, un resumen del proceso para un ED/rand/1/bin:

1. Generar y evaluar una población inicial ($P_{n,j}$) de *n* soluciones posibles donde *j* es el número de dimensiones en el espacio de búsqueda
2. Repetir desde generación k = 0, 1, 2, . . . , hasta que se alcanza un criterio de parada
   a) Para cada vector objetivo ($\vec{x_i}$) de la población actual, donde $i \in \{1, \ldots, n\}$
   b) Se seleccionan de manera aleatoria tres miembros diferentes de la población ($\vec{x_{r0}}$, $\vec{x_{r1}}$ y $\vec{x_{r2}}$), donde *r0*, *r1* y *r2* $\in \{1, \ldots, n\}$ y *r0* $\neq$ *r1* $\neq$ *r2* $\neq$ *i*
   c) Calculamos la diferencia entre $\vec{x_{r1}}$ y $\vec{x_{r2}}$
   d) Multiplicamos la diferencia de $\vec{x_{r1}}$ menos $\vec{x_{r2}}$ ($\vec{x_{r1}}$ - $\vec{x_{r2}}$) por el factor de mutación *f* (parámetro del sistema), f·($\vec{x_{r1}} - \vec{x_{r2}}$)
   e) Se obtiene $\vec{v_i}$ añadiendo f·($\vec{x_{r1}} - \vec{x_{r2}}$) al vector base ($\vec{x_{r0}}$) para obtener la población mutada $V_{n,j}$
   f) Se obtiene $\vec{u_i}$ llevando a cabo sobrecruzamiento entre el vector objetivo $\vec{x_i}$ y el vector mutado $\vec{v_i}$
   g) Se lleva a cabo la selección del mejor individuo entre $\vec{x_i}$ y $\vec{u_i}$ para pasar a la siguiente generación

Como se puede observar en este pseudo-código, son varios los pasos que se deben llevar a cabo en los ED. En primer lugar, comenzamos con un conjunto de vectores elegidos aleatoriamente como población inicial de *n* soluciones posibles ($P_{n,j}$). Como se comentó anteriormente, se selecciona un vector objetivo ($\vec{x_i}$). Se seleccionan además un vector base ($\vec{x_{r0}}$) y otros dos vectores, ($\vec{x_{r1}}$ y $\vec{x_{r2}}$), son seleccionados aleatoriamente. Para cada $\vec{x_i}$ (individuo) de la generación actual, se forma un *"vector mutado"* ($\vec{v_i}$) haciendo uso de $\vec{x_{r0}}$, $\vec{x_{r1}}$ y $\vec{x_{r2}}$ como se muestra en la ecuación 3.9.

$$\vec{v_i} = \vec{x_{r0}} + f \cdot (\vec{x_{r1}} - \vec{x_{r2}}) \tag{3.9}$$

Donde *r0*, *r1* y *r2* son tres índices distintos escogidos al azar entre $\{1, \ldots, n\}$, y también distintos de *i*. f corresponde al factor de mutación con un valor comprendido entre 0 y 2 ($f \in [0, 2]$). La figura 3.8 muestra gráficamente como se obtendría $\vec{v_i}$ en un espacio euclídeo tridimensional.

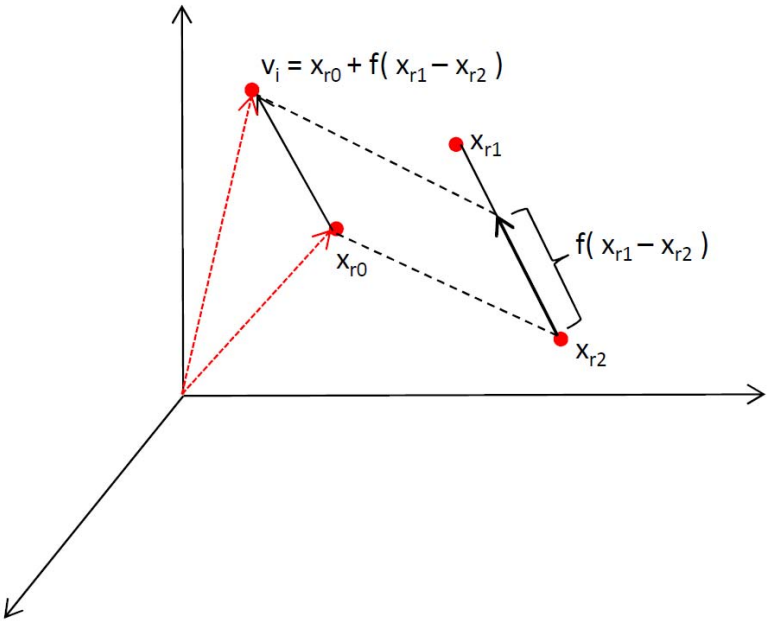


Figura 3.8: Proceso para obtener el vector mutado $\vec{v_i}$.

Una vez obtenido el vector *"mutado"* $\vec{v_i}$ se lleva a cabo el proceso de sobrecruzamiento con $\vec{x_i}$ y obtenemos así un vector ensayo $\vec{u_i}$. Para ello, para cada componente (*j*) de los vectores $\vec{x_i}$ y $\vec{v_i}$, se obtiene un número aleatorio dentro del rango [0,1] llamado $rand_j$. Suponemos además que la razón de sobrecruzamiento (*CR*, parámetro del sistema) es un punto de corte con valor $0 <= CR < 1$. Si $rand_j <= CR$, u_{ij} = v_{ij}, si no u_{ij} = x_{ij}. Para asegurarnos de que se lleva a cabo al menos algún sobrecruzamiento, se selecciona un componente de $\vec{u_i}$ (aleatoriamente) para que pase directamente desde $\vec{v_i}$.

Por ejemplo, si tuviéramos $\vec{x_i}$ y $\vec{v_i}$ como se muestra en la ecuación 3.10, entonces $\vec{u_i}$ podría ser el resultado mostrado en la misma ecuación.

$$\begin{aligned} \vec{x_i} &= (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}) \\ \vec{v_i} &= (v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5}) \\ \vec{u_i} &= (v_{i1}, x_{i2}, x_{i3}, x_{i4}, v_{i5}) \end{aligned} \tag{3.10}$$

Donde el índice 1 de $\vec{u_i}$ (v_{i1}) ha sido seleccionado aleatoriamente como sobrecruzamiento directo y definitivo desde $\vec{v_i}$. v_{i5} ha sido seleccionado desde $\vec{v_i}$ porque $rand_5$ era menor que CR y, en el resto de casos, $rand_j$ era mayor que CR. La figura 3.9 muestra un ejemplo del proceso de sobrecruzamiento.

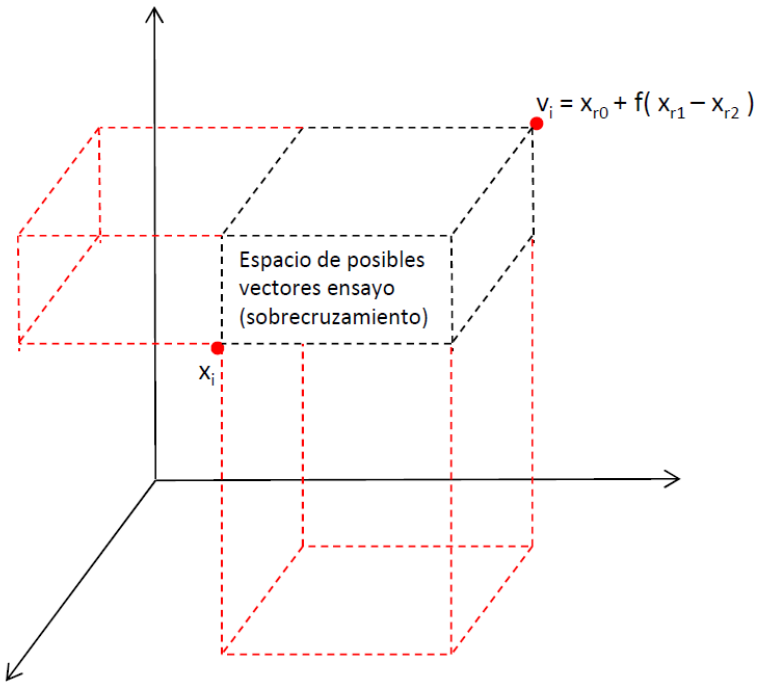


Figura 3.9: Proceso de sobrecruzamiento de $\vec{x_i}$ y $\vec{v_i}$ para formar los posibles vectores ensayo $\vec{u_i}$.

Cuando se ha obtenido el vector $\vec{u_i}$ mediante el sobrecruzamiento entre $\vec{x_i}$ y $\vec{v_i}$, se debe realizar la selección. La selección en este algoritmo es simple, si el nuevo descendiente ($\vec{u_i}$) es mejor (es decir, tiene mejor valor de adecuación) que el vector objetivo ($\vec{x_i}$) entonces $\vec{u_i}$ pasa a la siguiente generación, de lo contrario es el vector objetivo $\vec{x_i}$ el que pasará a la próxima generación. La figura 3.10 muestra el proceso completo de un ED.

La ecuación 3.11 muestra un resumen del algoritmo ED.

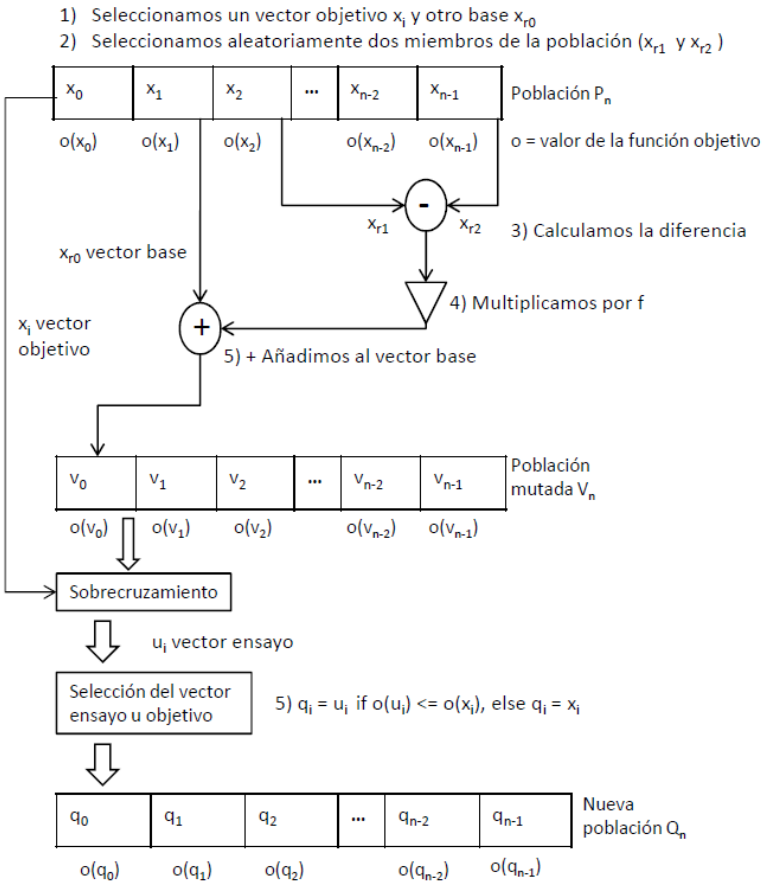


Figura 3.10: Esquema de un ED.

$$\vec{u}_{ij} = \begin{cases} \vec{x}_{r0j} + f \cdot (\vec{x}_{r1j} - \vec{x}_{r2j}) \; si \; (rand_j < CR \; or \; j = j_{rand}) \\ \vec{x}_{ij} \end{cases} \tag{3.11}$$

Para poder aplicar el algoritmo ED a nuestra aproximación fue necesario reemplazar el AG, el cual es responsable de llevar a cabo la búsqueda global en el sistema híbrido, por dicho algoritmo ED. Como podemos observar en la figura 3.11 la principal diferencia con respecto al proceso original representado por la figura 3.6 consiste en los operadores evolutivos usados.

Una vez que se conoce en detalle como funciona el algoritmo ED, podemos comenzar a estudiar las diferencias existentes con el AG.

Hemos podido observar en el pseudo-código del ED, que aunque el algoritmo ED dispone también de proceso de selección, mutación y sobrecruzamiento, éstos

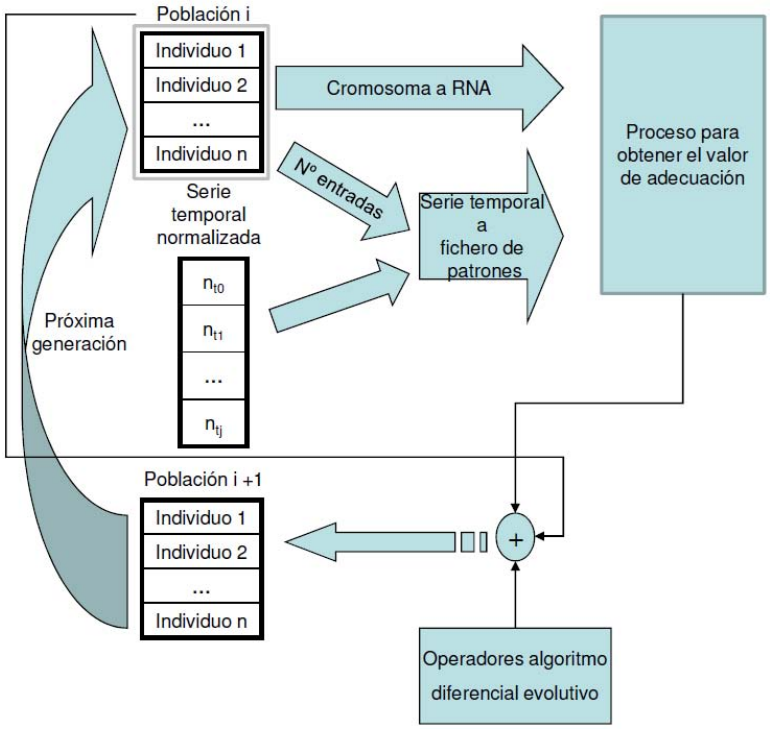


Figura 3.11: Esquema de diseño de RNA mediante algoritmo ED.

son diferentes a los utilizados por el AG.

En cuanto al operador de mutación se refiere, en el AG éste consiste en un porcentaje probabilístico (parámetro de mutación) aplicado a cada uno de los genes del cromosoma. En el AG se obtiene un número aleatorio para cada gen, y si el valor obtenido se encuentra por debajo del rango establecido por el parámetro de mutación, este gen variará su valor actual a otro.

Por contra, el factor de mutación presentado en ED consiste en un parámetro del sistema que se multiplicará sin excepciones a todas y cada una de las diferencias de dos individuos elegidos al azar de la población actual ($\vec{x_{r1}} - \vec{x_{r2}}$). Además, en el proceso de mutación, se incluye también la suma de otro individuo de la población ($\vec{x_{r0}}$). Por lo tanto, en el proceso de mutación del ED son necesarios tres individuos.

Referente al sobrecruzamiento, en el AG se seleccionan dos progenitores. Se obtiene aleatoriamente un punto de corte por el cual se dividirán ambos y con las partes opuestas de cada uno se obtendrán dos descendientes.

En ED, la primera diferencia que se puede observar es que se necesitan dos progenitores ($\vec{x_i}$ y $\vec{v_i}$), pero a diferencia del AG, tan solo se obtendrá un descen-

diente. La manera de obtener el descendiente es también muy diferente al sobrecruzamiento del AG. En ED, se hace uso de un operador de sobrecruzamiento que se aplica con una probabilidad de *CR*, pero en este caso se aplica al menos una vez por cada individuo. Para cada gen se obtiene un valor aleatorio, si éste es menor que CR, el gen del descendiente provendrá de un progenitor y en caso contrario provendrá del otro progenitor. Por lo tanto existe un cruce total y aleatorio entre progenitores.

Por último, el operador de selección difiere principalmente entre el AG y el ED en el momento en que se aplica. Mientras que en el AG se aplica al principio para seleccionar los individuos con los que trabajaremos, en ED se hace uso de él al final. Además, en ED el operador de selección consiste en la función *"es mejor"*, adoptada para probar la condición del posible reemplazo del progenitor ($\vec{x_i}$) por el descendiente ($\vec{u_i}$) obtenido mediante los operadores evolutivos anteriores. Estas comparaciones locales hacen al algoritmo más eficiente, evitando la necesidad de clasificación o ranking de toda la población. Por contra, se consume más tiempo computacional que en el AG al tener que volver a evaluar cada descendiente obtenido antes de poder compararlo con su progenitor.

3.6.3. Diseño de Redes de Neuronas Artificiales mediante Algoritmo de Estimación de Distribución

Después de haber hecho uso del algoritmo ED y como podremos comprobar en el capítulo 4, los resultados han sido mejorados cuando se comparan con los obtenidos por la versión del sistema con AG. Pero como también comentamos en la sección anterior 3.6.2, el tiempo computacional consumido es significativo. Debemos entonces hacer uso de un algoritmo evolutivo encargado de la búsqueda global del sistema que mejore los resultados y no consuma más tiempo del ya empleado por los AG. Es por ello que se plantea en esta sección el Algoritmo de Estimación de Distribución (AED).

Si quisiéramos implementar alguna de las versiones de AED con dependencias, antes deberíamos encontrar dichas dependencias y eso acarrearía un coste computacional extra sobre el sistema. Se ha optado entonces por hacer uso de UMDA *"Univariate Marginal Distribution Algorithm"*, sin ningún tipo de dependencias entre variables de acuerdo con [60]. A continuación presentamos el proceso general para un AED:

1. Generamos y evaluamos una población inicial de soluciones D_0
2. Repetimos desde generación k = 0,1,2,..., hasta alcanzar un criterio de

parada

a) Se obtiene $D_k^{Selección}$ seleccionando un subconjunto de soluciones de D_k

b) Se estima la distribución de probabilidad conjunta $P_k(x)$ a partir de $D_k^{Selección}$

c) Se generan las nuevas soluciones D_k^{Nuevo} muestreando $P_k(x)$

d) Creamos la nueva población D_{k+1} con las soluciones propuestas en $D_k^{Nuevo} \cup D_k^{Selección}$

e) D_k pasa a ser la generación actual a partir de D_{k+1}

f) Se evalúan las soluciones de la nueva generación de individuos D_k

Teniendo en cuenta el esquema anterior de un proceso de AED, a continuación presentamos un ejemplo de cómo funcionaría un UMDA decimal aplicando esta versión de AED (es decir, sin dependencias entre las variables) para un ejemplo académico de optimización con una población de 20 individuos.

Supongamos que estamos tratando de maximizar la función *Valuemax* (v(x)) definida en un espacio de búsqueda de seis dimensiones, representada en la ecuación 3.12

$$v(x) = \sum_{i=1}^{6} x_i \; con \; x_i \; \in \; [0, \ldots, 9] \tag{3.12}$$

El primer paso consiste en obtener una primera población de manera aleatoria, y a continuación cada uno de los individuos de dicha población será evaluado. Se muestra un ejemplo en la figura 3.12.

En un segundo paso los mejores *p* individuos (donde *p* es el parámetro de elitismo del sistema), aquellos con mejor valor de adecuación (en este caso mayor v(x)) son seleccionados. Para este ejemplo se han seleccionado los mejores diez individuos ($D_0^{Selección}$) como se puede observar en la figura 3.13. Estos individuos pasarán directamente a la siguiente generación (i.e. elitismo).

Después de seleccionar los mejores individuos e incluirlos en la siguiente generación, debemos expresar de manera explícita, mediante la distribución de probabilidad conjunta, las características de estos individuos seleccionados. Para ello haremos uso de la notación matemática que se puede observar en la ecuación 3.13.

$$p(x) = p(x_1, \ldots, x_6) = \prod_{i=1}^{6} p(x_i | D_0^{Selección}) \tag{3.13}$$

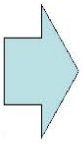
	x_1	x_2	x_3	x_4	x_5	x_6	v(x)
1	1	0	3	4	4	5	17
2	0	1	0	8	6	7	22
3	0	2	2	3	1	1	9
4	1	3	1	7	4	3	19
5	0	3	2	2	2	2	11
6	1	6	0	8	4	9	28
7	8	1	6	1	5	4	25
8	1	1	1	1	0	0	4
9	2	1	4	1	6	8	22
10	2	3	1	2	0	3	11
11	1	2	3	1	0	0	7
12	9	9	0	9	0	9	36
13	9	8	7	6	5	4	39
14	0	8	7	6	5	4	30
15	0	8	1	1	1	1	12
16	1	1	1	1	1	1	6
17	1	1	7	7	6	6	28
18	0	2	0	3	0	4	9
19	2	3	4	8	7	6	30
20	3	4	5	6	7	7	32

Figura 3.12: Población inicial D_0 para UMDA.

Por lo tanto, son necesarios sesenta parámetros (diez individuos con dimensión 6 cada uno) para especificar el modelo. Cada componente del producto, p($x_i|D_0^{Selección}$) con i = 1, ..., 6, se corresponde con la frecuencia relativa de $x_i = 1, \ldots, 9$ dentro del conjunto seleccionado de la población $D_0^{Selección}$; es decir:

- p($x_i = 0|D_0^{Selección}$)
- p($x_i = 1|D_0^{Selección}$)
- p($x_i = 2|D_0^{Selección}$)
- p($x_i = 3|D_0^{Selección}$)
- p($x_i = 4|D_0^{Selección}$)
- p($x_i = 5|D_0^{Selección}$)
- p($x_i = 6|D_0^{Selección}$)

	x_1	x_2	x_3	x_4	x_5	x_6	v(x)
1	1	0	3	4	4	5	17
2	0	1	0	8	6	7	22
3	0	2	2	3	1	1	9
4	1	3	1	7	4	3	19
5	0	3	2	2	2	2	11
6	1	6	0	8	4	9	28
7	8	1	6	1	5	4	25
8	1	1	1	1	0	0	4
9	2	1	4	1	6	8	22
10	2	3	1	2	0	3	11
11	1	2	3	1	0	0	7
12	9	9	0	9	0	9	36
13	9	8	7	6	5	4	39
14	0	8	7	6	5	4	30
15	0	8	1	1	1	1	12
16	1	1	1	1	1	1	6
17	1	1	7	7	6	6	28
18	0	2	0	3	0	4	9
19	2	3	4	8	7	6	30
20	3	4	5	6	7	7	32



	x_1	x_2	x_3	x_4	x_5	x_6	v(x)
2	0	1	0	8	6	7	22
6	1	6	0	8	4	9	28
7	8	1	6	1	5	4	25
9	2	1	4	1	6	8	22
12	9	9	0	9	0	9	36
13	9	8	7	6	5	4	39
14	0	8	7	6	5	4	30
17	1	1	7	7	6	6	28
19	2	3	4	8	7	6	30
20	3	4	5	6	7	7	32

Figura 3.13: $D_0^{Selección}$, individuos seleccionados de la población inicial.

- p($x_i = 7|D_0^{Selección}$)
- p($x_i = 8|D_0^{Selección}$)
- p($x_i = 9|D_0^{Selección}$)

Lo que significa que calcularemos que probabilidad existe en $D_0^{Selección}$ de que un gen tome un valor específico. Por ejemplo, en el caso que tratamos, la probabilidad de que x_1 = 0 equivale a calcular de cuantos *"0"* disponemos en esa variable y dividirlo entre el número de individuos, como mostramos en la figura 3.14 y en la ecuación 3.14.

$$p(x_1 = 0) = \frac{2}{10} = 0,2 \tag{3.14}$$

Llegados a este punto debemos prestar especial atención a un problema que se plantea a la hora de calcular estas probabilidades. En el caso de la figura 3.14, ocurre que la variable x_1 no toma el valor 4 para ningún individuo, al igual que pasa con los valores 5, 6 y 7. Por lo tanto, p(x_1=4) = p(x_1=5) = p(x_1=6) = p(x_1=7) = 0/10 = 0. Para evitar que este caso pueda producirse y por lo tanto evitar que la probabilidad de que algún individuo de las generaciones futuras sea nula para

	x_1	x_2	x_3	x_4	x_5	x_6	v(x)
2	0	1	0	8	6	7	22
6	1	6	0	8	4	9	28
7	8	1	6	1	5	4	25
9	2	1	4	1	6	8	22
12	9	9	0	9	0	9	36
13	9	8	7	6	5	4	39
14	0	8	7	6	5	4	30
17	1	1	7	7	6	6	28
19	2	3	4	8	7	6	30
20	3	4	5	6	7	7	32

Figura 3.14: Probabilidad de que $x_1 = 0$ en $D_0^{Selección}$.

obtener un 4, 5, 6 o un 7 en el gen x_1, se hace uso de la corrección de Laplace [53].

Dicha corrección da una mínima posibilidad a aquellos casos en los que una variable tenga en todos los individuos un único valor. Para ello, se suma *"1"* al numerador mientras que al denominador de la ecuación que calcula las probabilidades le suma un número equivalente a todos los posibles estados que puede tomar dicha variable, en nuestro caso diez estados (desde *"0"* hasta *"9"*). Así, las probabilidades para la variable x_1 quedaría como muestran la ecuación 3.15. Este proceso se repetiría para cada una de las variables x_i, $i \in \{1, 2, \ldots, 6\}$.

$$\begin{aligned}
p(x_1 = 0|D_0^{Selección}) &= 3/20 \\
p(x_1 = 1|D_0^{Selección}) &= 3/20 \\
p(x_1 = 2|D_0^{Selección}) &= 3/20 \\
p(x_1 = 3|D_0^{Selección}) &= 2/20 \\
p(x_1 = 4|D_0^{Selección}) &= 1/20 \\
p(x_1 = 5|D_0^{Selección}) &= 1/20 \\
p(x_1 = 6|D_0^{Selección}) &= 1/20 \\
p(x_1 = 7|D_0^{Selección}) &= 1/20 \\
p(x_1 = 8|D_0^{Selección}) &= 2/20 \\
p(x_1 = 9|D_0^{Selección}) &= 3/20
\end{aligned} \tag{3.15}$$

Muestreando esta distribución de probabilidad conjunta, *p(x)*, se obtiene una nueva población de individuos, D_{k+1}. Así, para cada variable x_i de cada uno de los individuos de la población $D_0^{Selección}$, se obtendrá un número al azar entre el intervalo [0,1]. Si el valor obtenido está comprendido dentro de la probabilidad dada para que esta variable tome ese valor específico, el nuevo individuo tendrá ese valor en su correspondiente variable.

En la figura 3.15 se pueden observar los 20 individuos de la siguiente generación D_{k+1}. En rojo aquellos provenientes de $D_0^{Selección}$, además de los diez nuevos obtenidos mediante el algoritmo presentados en negro.

	x_1	x_2	x_3	x_4	x_5	x_6	v(x)
1	2	3	1	1	2	6	15
2	0	1	0	8	6	7	22
3	5	4	5	6	1	7	28
4	4	3	1	2	2	9	21
5	4	3	5	6	2	8	28
6	1	6	0	8	4	9	28
7	8	1	6	1	5	4	25
8	3	3	4	5	4	3	22
9	2	1	4	1	6	8	22
10	4	3	3	2	2	3	17
11	4	6	6	5	6	5	32
12	9	9	0	9	0	9	36
13	9	8	7	6	5	4	39
14	0	8	7	6	5	4	30
15	0	3	6	1	5	5	20
16	4	5	6	2	2	2	21
17	1	1	7	7	6	6	28
18	0	7	5	5	0	1	18
19	2	3	4	8	7	6	30
20	3	4	5	6	7	7	32

Figura 3.15: Siguiente población formada por $D_0^{Selección}$ mas diez individuos nuevos, D_{k+1}.

Se trata de emplear ahora AED, en lugar de AG, para realizar la búsqueda global en el sistema híbrido. Como podemos observar en la figura 3.16 la principal diferencia con respecto al proceso original representado por la figura 3.6 consiste en los operadores evolutivos usados. En esta ocasión, nuestra aproximación hará uso de los operadores evolutivos propios del AED.

Una vez profundizado en detalle en el AED en su versión UMDA, podemos comenzar a indicar las diferencias existentes con el AG.

Tanto el AG como AED aplican elitismo, seleccionando los mejores individuos de la población actual para formar parte de la siguiente generación. Sin embargo, mientras el AG hace uso de toda la población actual para obtener los nuevos individuos que serán parte de la siguiente generación, en AED tan solo se emplean los mejores individuos previamente seleccionados (aquellos obtenidos por elitismo) para poder generar los nuevos individuos de la próxima generación.

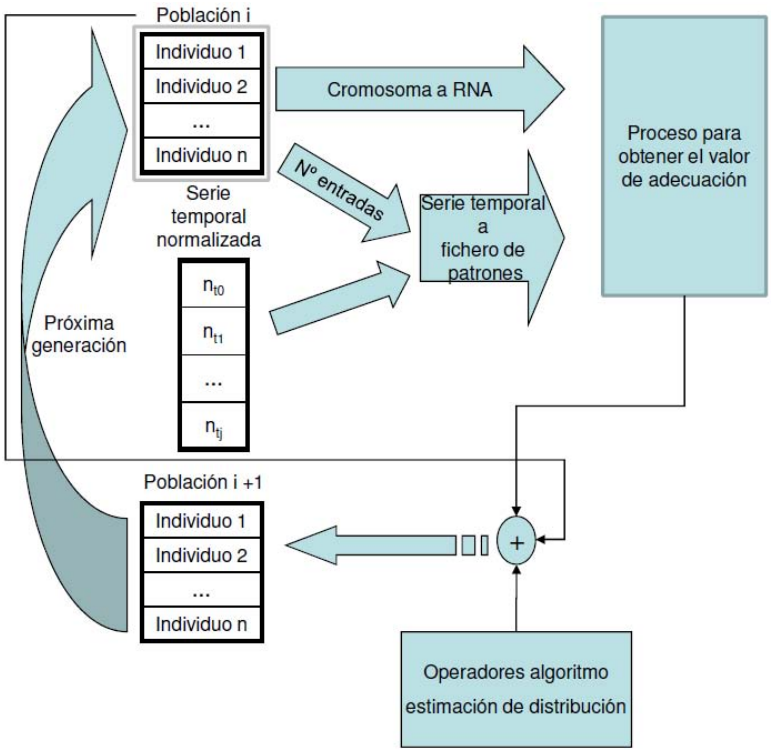


Figura 3.16: Esquema de diseño de RNA mediante AED.

En cuanto a los operadores de mutación y sobrecruzamiento se refiere, el AG dispone de ambos, no ocurriendo así para AED. En su lugar, se lleva a cabo una estimación de la distribución de probabilidad conjunta de cada uno de los genes de los individuos previamente seleccionados (aquellos obtenidos por elitismo) y muestreo de las nuevas soluciones a partir de esta probabilidad.

Al no existir sobrecruzamiento en AED, evitamos la posible interrupción causada por el sobrecruzamiento [124], manteniendo así los bloques funcionales como se explicó en la sección 2.9.1.

3.7. Ventana Deslizante y Perceptrón Multicapa Parcialmente Conectado

En esta sección se detalla el estudio que propone profundizar en el diseño de la arquitectura de las RNA utilizadas durante el proceso de búsqueda desde otros dos puntos de vista diferentes.

La primera aproximación está orientada a los datos de entrada, y trata de demostrar que la selección de instantes específicos de datos de la serie temporal juega un papel importante a la hora de obtener resultados más precisos y como la selección de éstos afecta al número de nodos de entrada de la RNA.

La segunda consiste en hacer uso de una arquitectura diferente, en esta ocasión se hará uso del perceptrón multicapa parcialmente conectado.

Para estas nuevas propuestas llevadas a cabo a nivel de diseño de las RNA, se ha optado por modificar el algoritmo de aprendizaje usado en ADANN, es decir el algoritmo de retropropagación por el *"Resilient Propagation"* (RPROP) [88, 89, 90].

Este algoritmo está considerado como uno de los algoritmos más prácticos a la hora de estimar el valor de los pesos de conexión de una RNA. El algoritmo de retropropagación modifica los valores de los pesos proporcionalmente al gradiente de la función de error, de tal manera que en aquellas regiones donde el gradiente tiende a ser plano este algoritmo avanza más lentamente. En cambio, el algoritmo RPROP difiere de la técnica clásica planteada por retropropagación ya que las derivadas parciales de la función de error sólo se usan para determinar el sentido en que deben ser corregidos los pesos de la red, pero no las magnitudes de los ajustes.

Como la derivada solamente se usa para determinar la dirección en la actualización de los pesos, RPROP tampoco se ve afectado por la saturación de los nodos de la RNA. Por lo tanto, su convergencia de entrenamiento es más rápida en comparación con otros algoritmos por lo que requiere menos esfuerzo computacional. Además son muchos y variados los trabajos que demuestran que RPROP es una buena opción como algoritmo de aprendizaje [196, 197, 198].

Para poder hacer uso de este nuevo algoritmo de aprendizaje, se ha modificado la aproximación (ADANN) en cuanto a la codificación del cromosoma se refiere.

Además, se ha seleccionado AED como algoritmo evolutivo en sustitución del AG dando lugar a un sistema denominado *Algoritmo de Estimación de Distribución para Diseñar Redes de Neuronas Artificiales* (AEDRNA). La principal razón por las que se ha optado por usar AED en lugar de AG es el hecho de que AED no necesita parametrizar el proceso de sobrecruzamiento y mutación ya que estos pasos no existen en dicho algoritmo.

En esta nueva aproximación del sistema, se sigue considerando el perceptrón multicapa completamente conectado con una única capa oculta [86, 199, 200] y un único nodo de salida. El rendimiento de RPROP puede depender de la correcta regulación de dos parámetros numéricos, conocidos como $Delta_{max} \in \mathbb{R}$ y $\alpha \in [0, 1]$. Mientras el parámetro α determina la relación entre el error de salida y la

reducción en el tamaño de los pesos (exactamente α denota el exponente negativo en base decimal de dicha relación), $Delta_{max}$ representa el límite superior del tamaño de los cambios de peso.

Por lo tanto, adoptamos de nuevo un esquema de codificación directo, que sitúa en el cromosoma:

- Dos dígitos decimales (2 genes), para codificar el número de nodos de entrada (*i*).
- Dos dígitos decimales (2 genes) para el número de nodos en la capa oculta (*h*).
- Dos dígitos decimales (2 genes) para $Delta_{max} \in \{0, 1, \ldots, 99\}$.
- Un dígito decimal (1 gen) para $\alpha = \{1, 0{,}1, 0{,}01, \ldots, 10^{-9}\}$.

De esta forma, los valores de *"i"*, *"h"*, $Delta_{max}$ y α se obtienen del cromosoma como se indica en la figura 3.17 y en la ecuación 3.16.

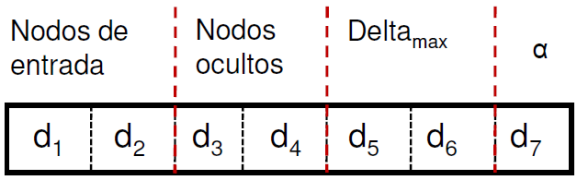


Figura 3.17: Esquema de codificación para AEDRNA.

$$\begin{aligned} d_j &\in \{0, \ldots, 9\} \\ i &= NTS \cdot max\_entradas \cdot \tfrac{(d1 \cdot 10) + d2}{100} + 1 \\ h &= NTS \cdot max\_ocultas \cdot \tfrac{(d3 \cdot 10) + d4}{100} + 1 \\ Delta_{max} &= (d5 \cdot 10) + d6 \\ \alpha &= 10^{-d7} \end{aligned} \tag{3.16}$$

En la ecuación 3.16, *NTS* representa el número de elementos de los que dispone la serie temporal con la que trabajamos en ese momento. **Max_entradas** es un parámetro del sistema dado por el usuario para calcular el número de nodos en la capa de entrada y **max_ocultas** es otro parámetro empleado para calcular los

nodos de la capa oculta de la RNA (su valor será siempre el doble del valor dado por **max_entradas**) como ya explicamos en la sección 3.6.1.

El proceso de búsqueda ahora planteado por el sistema AEDRNA es muy parecido al llevado a cabo por ADANN y consiste en los siguientes pasos (figura 3.16):

1. En primer lugar se obtiene una población aleatoria, es decir, un conjunto de cromosomas generados aleatoriamente (volvemos a fijar el tamaño de la población en 50).
2. Se obtienen los fenotipos (arquitecturas de las RNA) y se obtiene el valor de evaluación de cada individuo de la generación actual. Para obtener el fenotipo asociado a un cromosoma y su valor de adecuación :
   a) Se obtiene primero el fenotipo de un individuo de la generación actual (haciendo uso de la herramienta SNNS).
   b) Para cada RNA, los subconjuntos de patrones de entrenamiento y validación son obtenidos a partir de los datos de la serie temporal en función del número de nodos de entrada de dicha RNA, como se explicó en la sección 3.2.
   c) La RNA se entrena con RPROP (utilizando SNNS). Cuando el error de validación (el error obtenido por el subconjunto de patrones de validación) es mínimo durante el proceso de entrenamiento se almacena la arquitectura (topología y pesos de las conexiones) de la RNA.
   Por lo tanto, esta arquitectura será el fenotipo final del individuo, mientras que su valor de adecuación será el mínimo MSE de validación, obtenido durante el proceso de entrenamiento.
3. Una vez que los valores de evaluación para toda la población ya se han obtenido, se aplican los operadores de UDMA-EDA [53] (selección, estimación de la distribución de probabilidad empírica y muestreo de soluciones) con el fin de generar la población de la próxima generación, es decir, un nuevo conjunto de cromosomas.
4. Los pasos 2 y 3 se ejecutan iterativamente hasta que se alcanza un número máximo de generaciones.

3.7.1. *"Time Lags"* o Ventana Deslizante

Esta sección propone una mejora sobre la propuesta del sistema (AEDRNA) a la que pasaremos a denominar *Algoritmo de Estimación de Distribución para Diseñar Redes de Neuronas Artificiales con Ventana Deslizante* (AEDRNAVD).

Esta versión del sistema (AEDRNAVD), se centra en tres puntos claves:

- Llevar a cabo una elección más selectiva de los elementos anteriores de la serie temporal, eligiendo los que de verdad influyen en el resultado final.
- Obtener como resultado modelos más sencillos y por lo tanto más fáciles de estudiar en detalle.
- Reducir el tiempo computacional requerido en la experimentación.

Para ello AEDRNAVD busca los instantes específicos anteriores de la serie temporal haciendo uso de una ventana deslizante [14, 199, 200], especificando no solo cuántos valores anteriores tendremos en cuenta para predecir el actual sino cuáles de ellos se usan y cuáles no serán tenidos en cuenta.

Como se explicó en la sección 3.2, cada vez que se genera un nuevo individuo (RNA), los subconjuntos de patrones de entrenamiento y validación asociados a éste también deben ser obtenidos. En el ejemplo anterior de la figura 3.2, si la RNA disponía de k nodos de entrada, todos y cada uno de los k valores anteriores $(y_{t-1}, y_{t-2}, \ldots, y_{t-k})$ de la serie temporal eran usados para generar dichos subconjuntos de patrones.

Mostramos a continuación un nuevo nivel de especialización en esta sección, donde se obtienen los patrones por filtrado de intervalos cronológicos, es decir, seleccionando los instantes anteriores relevantes de la serie temporal. Dichos instantes serán usados para especificar el número de nodos de entrada de la RNA y a su vez, para generar los patrones que formen los subconjuntos de entrenamiento y validación que alimentarán la RNA.

Para llevar a cabo esta aproximación, esta información ha de incluirse en el cromosoma, lo que supone añadir más genes que codifiquen dicha información. En concreto, hemos adoptado una codificación binaria, donde cada nuevo gen (b_k) define si el correspondiente instante anterior es utilizado (con valor *"1"*) o no se tiene en cuenta (con valor *"0"*) por el sistema para generar los patrones. Por lo tanto, el tamaño de la ventana deslizante vendrá dado por el número de nodos de entrada de la RNA.

La figura 3.18 y la ecuación 3.17 muestran el esquema de codificación del cromosoma, que incluye la selección llevada a cabo por la ventana deslizante, y cómo obtener el fenotipo a partir de la información codificada en dicho cromosoma respectivamente. *NTS*, **max_entradas** y **max_ocultas** seguirán teniendo la misma funcionalidad que tenían en la aproximación del sistema AEDRNA.

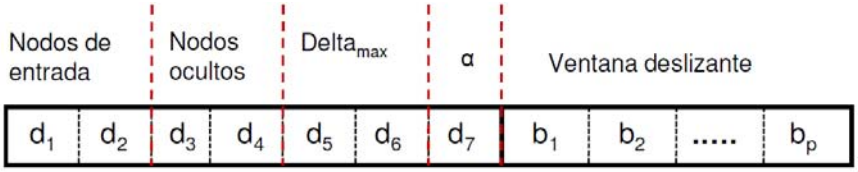


Figura 3.18: Esquema de codificación para AEDRNA con ventana deslizante (AEDRNAVD).

$$\begin{aligned} d_j &\in \{0,\ldots,9\} \\ i &= NTS \cdot max\_entradas \cdot \tfrac{(d1\cdot 10)+d2}{100} + 1 \\ h &= NTS \cdot max\_ocultas \cdot \tfrac{(d3\cdot 10)+d4}{100} + 1 \\ Delta_{max} &= (d5 \cdot 10) + d6 \\ \alpha &= 10^{-d7} \\ Ventana\ deslizante &= b_1,\ldots,b_p \in [0,1] \end{aligned} \tag{3.17}$$

El número de nodos de entrada de la RNA y por lo tanto de instantes anteriores de la serie temporal que serán usados para formar los patrones sigue siendo dado por el parámetro *i*. Sin embargo, ahora disponemos de un nueva parte del cromosoma (ventana deslizante) que con su información de tipo binario nos indicará cuales de los instantes anteriores son empleados y cuales no.

Consideremos un ejemplo similar al propuesto en la sección 3.2 y más concretamente en la figura 3.2 para explicar como funciona la ventana deslizante. Igual que en aquel ejemplo, se considera que *i* = 3, pero en esta ocasión haremos uso de la ventana deslizante con los valores iniciales b_1 = 1, b_2 = 0 y b_3 = 1.

Entonces, como se puede observar en la figuras 3.19, el número de nodos de entrada de la RNA es establecido al final no solo por el valor de *i* = 3, sino que también depende de la información aportada por la parte binaria del cromosoma, el cual tan sólo activa dos instantes anteriores, de los tres dados por *i*.

Es por ello que al final la topología de la RNA resultante (figura 3.20) no tendrá 3 nodos de entrada sino 2. Además también cabe destacar que los patrones formados tendrán el formato que muestra el primer patrón, $n_{t0}, n_{t2} \Rightarrow n_{t3}$, donde el instante intermedio es eliminado ya que b_2 = 0.

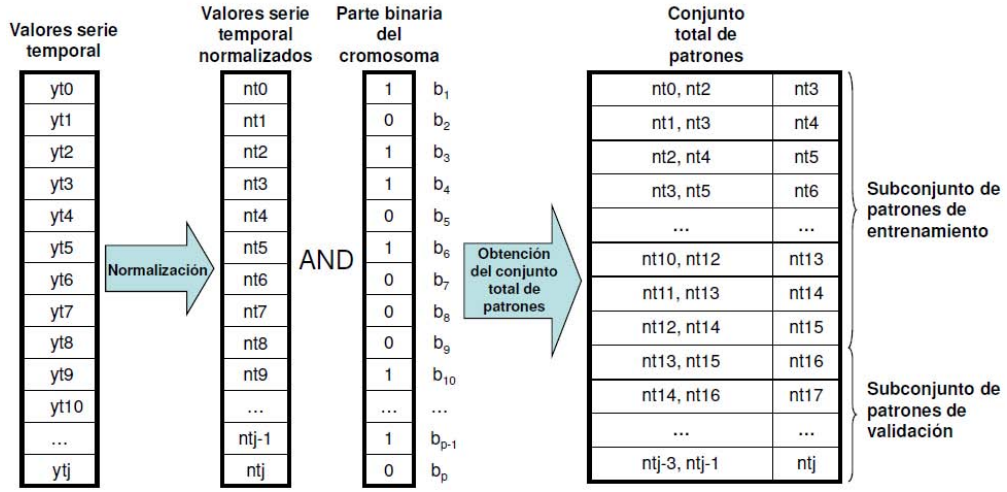


Figura 3.19: Proceso para obtener los conjuntos de entrenamiento y validación con ventana deslizante e *i=3* (AEDRNAVD).

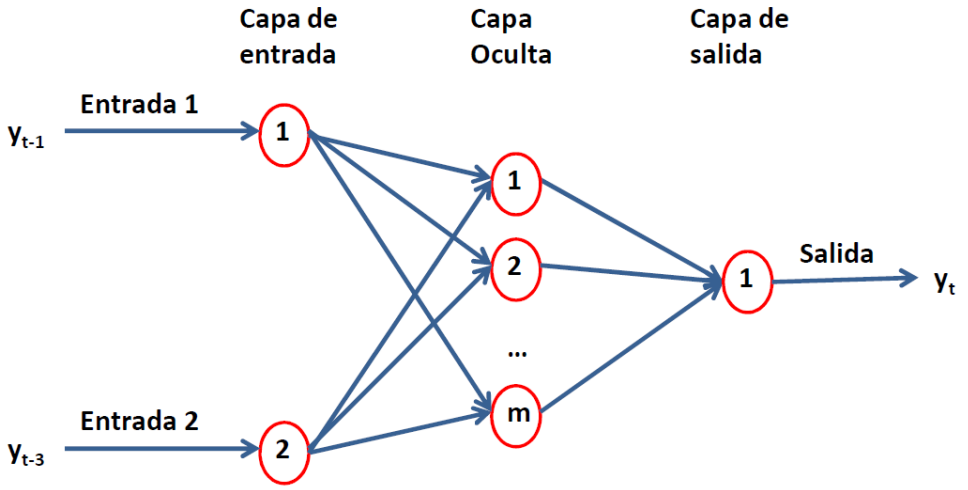


Figura 3.20: Fenotipo de la RNA obtenida como resultado con los valores de la serie temporal seleccionados como entrada.

3.7.2. Perceptrón Multicapa Parcialmente Conectado

Una vez explicado como llevar a cabo una selección en detalle de los instantes anteriores de la serie temporal, se presenta en esta sección otra aproximación basada de nuevo en la versión del sistema AEDRNA. A esta nueva versión la denominamos *Algoritmo de Estimación de Distribución para Diseñar Redes de Neuronas Artificiales Parcialmente Conectadas* (AEDRNAPC).

Con esta nueva aproximación tratamos de demostrar cómo considerando una nueva topología de las RNA basada en el perceptrón multicapa parcialmente conectado, podemos obtener mejores resultados. Damos pues la posibilidad a la búsqueda global, llevada a cabo por el AED, de eliminar aquellas conexiones entre nodos que no ayudan a obtener una mejor predicción.

Además, con el sistema AEDRNAPC se pretende hacer frente a dos puntos claves:

- Obtener como resultado modelos más sencillos y por lo tanto más fáciles de estudiar en detalle.

- Reducir el tiempo computacional consumido en la experimentación.

Un área de investigación importante se refiere a la evolución de las topologías de las RNA. Miller et al. [136] sugirió que la CE es muy buena candidata para buscar en el espacio de todas las posibles topologías.

Miller et al. [136] identificó dos aproximaciones para codificar la topología en el cromosoma: esquema de especificación fuerte (o esquema de codificación directa), donde se especifica cada conexión de la red mediante su representación binaria; y esquema de especificación débil (o esquema de codificación indirecta), donde el patrón exacto de la conectividad no está explícitamente representado, sino que se obtiene en base a la información codificada en el cromosoma aplicando determinadas reglas de desarrollo.

El esquema de codificación directa ha sido adoptado por diversos autores [119, 123]. La propuesta aquí presentada de AEDRNA *Parcialmente Conectada* (AEDRNAPC) funciona como el sistema AEDRNA presentado al comienzo de la sección 3.7, sólo que ahora podemos evolucionar las estructuras de manera más flexible. Para ello, y siguiendo el trabajo de Whitley et al. [119] y de Schaffer et al. [123], hemos adoptado un esquema de codificación binaria directa.

Como ya se tuvo que hacer en la sección 3.7.1, se ha añadido información adicional al cromosoma inicial dado por el sistema AEDRNA. Esta información adicional consiste en una nueva parte binaria que será la encargada de representar las conexiones entre nodos de las RNA. La figura 3.21 y la ecuación 3.18 muestran en detalle el nuevo cromosoma con las conexiones añadidas, éstas quedan representadas mediante el conjunto de dígitos binarios añadidos al final del cromosoma.

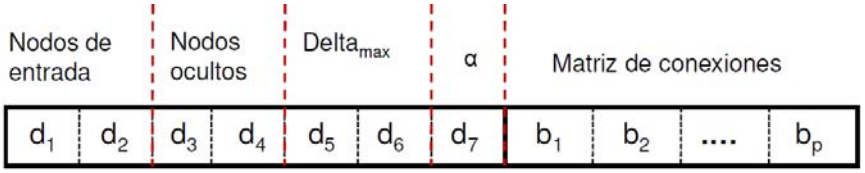


Figura 3.21: Esquema de codificación para AEDRNAPC.

$$\begin{aligned} d_j &\in \{0,\ldots,9\} \\ i &= NTS \cdot max\_entradas \cdot \tfrac{(d1\cdot 10)+d2}{100} + 1 \\ h &= NTS \cdot max\_ocultas \cdot \tfrac{(d3\cdot 10)+d4}{100} + 1 \\ Delta_{max} &= (d5 \cdot 10) + d6 \\ \alpha &= 10^{-d7} \\ Matriz\ de\ conexiones &= b_1,\ldots,b_p \in [0,1] \end{aligned} \tag{3.18}$$

NTS, **max_entradas** y **max_ocultas** seguirán teniendo la misma funcionalidad que tenían en el sistema AEDRNA.

Como podemos observar, la matriz de conexiones viene dada en el cromosoma como un vector binario de dimensión p donde p = 20.000. Este número de conexiones se ha establecido mediante experimentación previa donde se ha corroborado que es un número suficiente de conexiones para las RNA con las que nos disponemos a trabajar.

Para poder llevar a cabo cualquier interpretación de la parte binaria del cromosoma y generar así la topología de la RNA correspondiente, antes este vector de datos debe ser transformado a una estructura de datos más manejable. Debido a esto, primero deberemos transformar la parte binaria del cromosoma (información de la conectividad) en una matriz de dimensiones $F \times C$ (100×200).

En la figura 3.22 se puede observar un ejemplo de cómo transformar la codificación binaria del cromosoma en una matriz de conexiones de la RNA. Un dato importante a tener en cuenta es qué representa cada fila y columna de la matriz y por lo tanto, qué representa cada celda:

- Las *i* primeras filas representan las *i* entradas existentes en la RNA, valor obtenido del cromosoma.

- La fila representada con *O* se corresponde con la única neurona de salida.
- Las *h* columnas de la matriz representan los nodos de la única capa oculta de la que disponen las RNA.

En cada celda de la matriz se podrá encontrar o bien un *"0"* o un *"1"*, donde un *"0"* significa que no existe conexión alguna entre dos nodos y un *"1"* representa una conexión existente entre estos dos nodos.

Así, la manera de trabajar con esta matriz consiste en:

- En caso de que exista un *"1"* en una celda situada en alguna de las *i* primeras filas, este bit representa una conexión entre dicho nodo de entrada *q* y el nodo oculto representado por el número de columna *r* de la celda. Si por el contrario el bit fuera un *"0"*, no existe dicha conexión.
- En caso de que exista un *"1"* en una celda situada en la última fila representada como *O*, esto representa una conexión entre el nodo oculto dado por la la columna *r* de dicha celda y el nodo de salida *O* dado por la fila de la celda.

Por ejemplo, el bit b_3 cuyo valor es *"1"*, representa una conexión que se lleva a cabo entre el nodo de entrada 1 y el nodo oculto 3 de la RNA.

Por defecto, se ha optado por establecer una matriz de conexiones suficientemente grande con dimensiones $F \times C$ para todos los individuos a pesar de que las dimensiones reales de la RNA son limitadas por los valores *i* y *h* obtenidos del cromosoma para cada individuo. Esto es así ya que se busca establecer una misma longitud de cromosoma para todos los individuos de la generación.

Cada vez que se utiliza un nuevo individuo (genotipo) para generar una RNA (fenotipo), los valores de *i* y *h* se determinan a partir del contenido su cromosoma, con el fin de descartar las filas y columnas adicionales no deseadas, dando lugar a una matriz con las dimensiones $(i + 1) \times h$, donde *i* y *h* representan el número real de los nodos de entrada y ocultos.

3.8. Resumen

En este capítulo, se ha presentado el sistema ADANN que fue desarrollado para participar en competiciones de predicción de series temporales tales como NN3, NN5 y NNGC1.

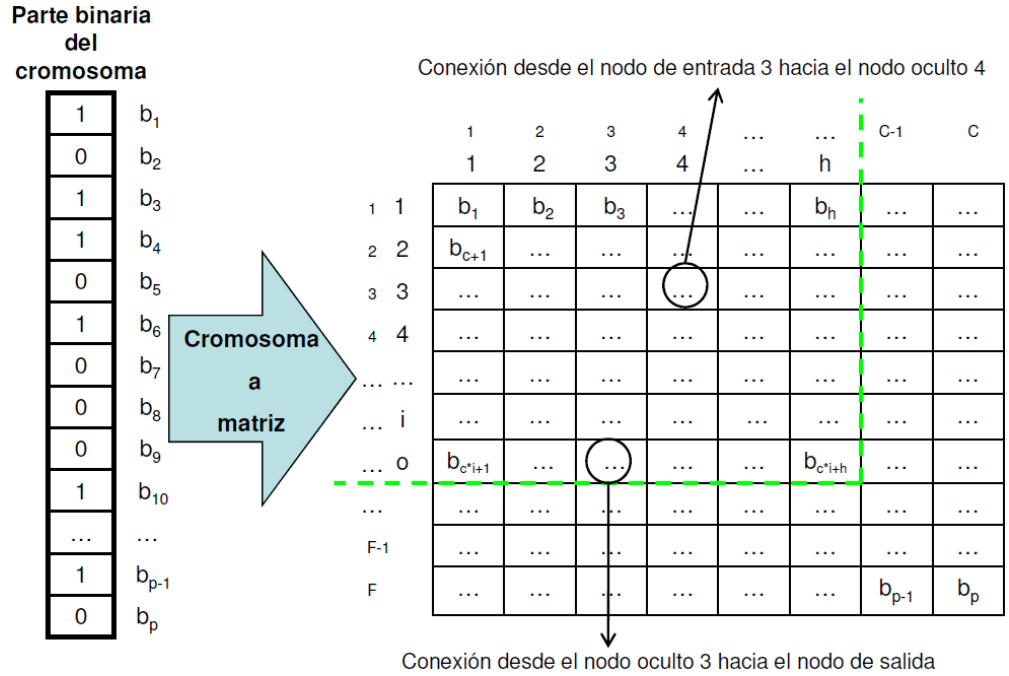


Figura 3.22: Ejemplo de cómo conseguir la matrix de conexiones desde un cromosoma dado.

También se han planteado dos posibles modelos de validación cruzada que pueden ser usados en aquellos casos en los que nuestro sistema no obtiene buenos resultados de predicción ya que los datos de entrada con los que trabaja son escasos. Además, se ha mostrado como la validación cruzada aplicada a RNAE obtiene por cada individuo tantas arquitecturas diferentes como número de conjuntos con los que se trabaja. Para poder realizar una única predicción a partir de todas estas arquitecturas, hemos planteado cuatro modelos diferentes de *"Ensembles"*.

Además, se ha estudiado el proceso de búsqueda global que se lleva a cabo mediante CE. ADANN hacía uso de AG para llevar a cabo dicha tarea. Se han propuesto también dos posibles alternativas al AG que son el algoritmo ED y AED.

Podemos confirmar que una de las ventajas principales de AED frente al AG y al algoritmo ED es que este no necesita parametrizar el proceso de sobrecruzamiento y mutación ya que estos pasos no existen en AED. En su lugar se lleva a cabo una estimación de distribución sobre el valor de los genes que forman los individuos. Es por ello que los únicos dos parámetros que necesita son el número máximo de generaciones (o lo que es igual, el criterio de parada) y el número de individuos existentes por cada una de estas generaciones.

Puesto que en AED no existe sobrecruzamiento como ocurre en los AG, evitamos el problema de separación dentro del cromosoma de los valores (pesos) asociados a cada conexión que llegan a un mismo nodo como se explicó en la sección 2.9.1

En cuanto al algoritmo ED, presenta una desventaja si lo comparamos con el AG. Dicha desventaja se presenta en el proceso de selección del algoritmo ED, ya que por cada individuo de la generación se obtiene un individuo hijo nuevo el cual hay que evaluar nuevamente para ser comparado con el padre. Así, en ED habrá que hacer una evaluación por cada individuo de la generación y otra por cada nuevo hijo obtenido, con lo que el tiempo de procesamiento se verá incrementado considerablemente, de hecho, ED consume el doble de tiempo que los AG.

Se han explicado en detalle como funcionan cada uno de estos tres algoritmos, las principales diferencias entre ellos y qué cambios llevar a cabo en el sistema para aplicarlos.

En este capítulo, presentamos también una nueva versión (AEDRNA) de la aproximación ADANN, donde en lugar de usar AG para la búsqueda global del sistema, este algoritmo es sustituido por AED. Además, a la hora de entrenar una RNA (o búsqueda local de los pesos en el sistema) ya no usamos el algoritmo de aprendizaje de retropropagación, en su lugar, RPROP es el nuevo algoritmo de aprendizaje seleccionado.

Por otro lado, se ha adoptado la popular ventana deslizante usada en predicción de series temporales mediante RNA [200] para modificar el sistema AEDRNA. En lugar de llevar a cabo un diseño manual de dicha ventana, la selección de cuantos y qué instantes anteriores de la serie temporal se realiza de un modo automático por medio del proceso evolutivo AED, a este nuevo sistema lo denominamos AEDRNAVD. Éste no requiere de conocimiento previo por parte del usuario sobre la serie temporal a predecir y por lo tanto es sencillo de usar.

Para finalizar, se ha estudiado el uso del perceptrón multicapa parcialmente conectado en lugar del totalmente conectado como modelo de diseño en nuestro sistema. Así, y mediante una representación binaria, el sistema dispone de la posibilidad de eliminar conexiones entre neuronas, pudiendo obtener modelos finales más precisos para predecir series temporales, esta propuesta es conocida como AEDRNAPC.

En el próximo capítulo se describirá la experimentación realizada para estas modificaciones y los diferentes resultados obtenidos.

Capítulo 4

Experimentación y Resultados

En este capítulo se muestra la experimentación llevada a cabo sobre la aproximación planteada y estudiada en esta tesis doctoral.

Se muestran los resultados asociados al sistema propuesto en esta tesis, con sus diferentes variantes. En primer lugar analizaremos los resultados de ADANN, el sistema con AG. Para ello estudiamos cómo tratar los datos con los que trabajan las RNA (i.e. normalizar dichos datos) y como se presentarán dichos datos a estas RNA para hacer sus predicciones mas precisas.

Entre los métodos propuestos para tratar con los datos de entrada se encuentra *"Shuffle"*. La idea principal de *"Shuffle"* consiste en introducir los patrones de entrada a las RNA de manera desordenada para que dichas RNA puedan aprender de diferentes partes de las series temporales y ser evaluadas también con diferentes partes de estas series, en lugar de aprender siempre del principio y ser evaluadas siempre con los datos del final como se venía haciendo con el sistema ADANN.

Se presentan además los resultados obtenidos con el método validación cruzada. En aquellos casos en que nuestro sistema no es capaz de llevar a cabo una predicción precisa de una serie temporal ya que ésta no dispone de datos suficientes con los que trabajar, se hará uso de validación cruzada.

Como se comentó con anterioridad en la sección 2.11, nos apoyaremos en la teoría de grupos o *"Ensembles"* para poder unificar todas y cada una de las predicciones realizadas por cada una de las arquitecturas obtenidas como resultado. Se plantean en este capítulo entonces dos modos de llevar a cabo la validación cruzada, ponderada y sin ponderar. Por otro lado, se han utilizado cuatro modelos de *"Ensembles"* diferentes para unificar las posibles salidas dadas para cada una de las dos versiones de validación cruzada. Mostraremos entonces los resultados de las ocho posibles combinaciones y explicaremos sus respectivos resultados.

En este capítulo se estudian también en detalle los resultados referentes al sistema ADANN cambiando la técnica de búsqueda (CE), de AG a ED y AED. Se lleva a cabo en este capítulo un estudio comparativo de las diferentes técnicas de CE planteadas y cómo al hacer uso de una u otra en nuestra aproximación afecta a los resultados de las predicciones.

Respecto a las diferentes variantes propuesta del sistema ADANN, primero llevamos a cabo un estudio comparativo entre ADANN, y una nueva versión de éste (AEDRNA) donde se hace uso de AED en lugar de AG como técnica evolutiva y de RPROP como algoritmo de aprendizaje en lugar del algoritmo retropropagación. A continuación pasamos a estudiar las diferencias entre AEDRNA y su versión con ventana deslizante (AEDRNAVD) y por último AEDRNA comparada con su versión con perceptrón multicapa parcialmente conectado (AEDRNAPC).

Para finalizar, se muestran además los resultados obtenidos con AEDRNAPC comparados con diferentes métodos o sistemas existentes y provenientes de distintos campos, métodos estadísticos y herramientas software comerciales entre otros.

Así, en primer lugar se realiza una comparación de las predicciones llevadas a cabo sobre un conjunto de series temporales con AEDRNAPC y dos de los métodos estadísticos más conocidos para predecir series temporales, Modelos de Componentes no Observables (tipo Harvey) [201] y ARIMA [20] de Box-Jenkins.

En segundo lugar se comparan los resultados obtenidos con AEDRNAPC con los obtenidos por una herramienta software comercial de predicción de series temporales denominada *ForecastPro* [202] (FP). Además, se mostrarán en detalle las predicciones mediante sus respectivas gráficas.

En la última sección se compararán los errores dados por las predicciones y se mostrarán las gráficas generadas por AEDRNAPC y otros tres sistemas de IC como son, Máquinas de Soporte Vectorial (MSV) y dos sistemas híbridos formados mediante lógica difusa y RNA (FRNA) y por otro lado lógica difusa y MSV (FMSV).

4.1. Series Temporales y Evaluación del Sistema

Para llevar a cabo la experimentación realizada en esta tesis, se ha hecho uso de doce series temporales. Estas series temporales son Passengers, Temperature, Dow-Jones, Quebec, Abraham12, Mackey-Glass, Ozone, IBM, Paper, Pigs, Cars y Milk [203]:

- **Passengers**: representa el número de pasajeros de una aerolínea internacional medidos en miles, mensualmente desde Enero de 1949 hasta Diciembre de 1960, la fuente es de Box & Jenkins (1976).
- **Temperature**: muestra la temperatura media mensual del aire en el castillo de Nottingham medida desde 1920 hasta 1939, la fuente es de O.D. Anderson (1976).
- **Dow-Jones**: trata sobre el valor mensual de cierre del índice industrial dowjones desde Agosto de 1968 hasta Agosto de 1981, la fuente es de Hipel y Mcleod (1994).
- **Quebec**: representa el número de nacimientos diarios en la provincia de Quebec en Canadá desde el 1 de Enero de 1977 hasta el 31 de Diciembre de 1978, la fuente es de Hipel & Mcleod (1994).
- **Abraham12**: representa la demanda mensual de gasolina en Ontario en millones de galones desde 1960 hasta 1975, la fuente es de Abraham & Ledolter (1983).
- **Mackey-Glass**: está basada en la ecuación diferencial Mackey-Glass y es considerada como punto de referencia para comparar la capacidad de generalización de diferentes métodos. Esta serie temporal es una serie caótica generada a partir de ecuaciones diferenciales ordinarias con un tiempo de retardo.
- **Ozone**: mide la concentración de ozono en Azusa medida mensualmente desde 1956 hasta 1970. La fuente es de Hipel y Mcleod (1994).
- **IBM**: corresponde al precio de cierre de las acciones de la compañía IBM, medidas diariamente desde el 17 de Mayo de 1961 hasta el 2 de Noviembre de 1962. La fuente es de Box & Jenkins (1976).
- **Paper**: son las ventas mensuales de papel escrito o impreso en Francia desde 1963 hasta 1972. La fuente es de O'Donovan (1983).
- **Pigs**: representa el número de cerdos sacrificados mensualmente en el estado australiano de Victoria, desde Junio de 1980 hasta Agosto de 1995. La fuente es de Andrews & Herzberg (1985).
- **Cars**: son los datos mensuales de ventas de coches en la provincia de Quebec, desde 1960 hasta 1968. La fuente es de Abraham & Ledolter (1983).
- **Milk**: representa la producción mensual de leche medida en libras por cabeza de ganado vacuno, desde 1962 hasta 1975. La fuente es de Cryer (1986).

A continuación mostraremos una gráfica de cada una de las series temporales propuestas. En la figura 4.1 podemos ver la serie temporal Passengers, en 4.2 está representada Temperature, en 4.3 Dow-Jones, en la figura 4.4 podemos ver la serie temporal Quebec, en 4.5 Abraham12, en 4.6 aparece Magkey-Glass, en 4.7 está representada Ozone, en 4.8 aparece IBM, 4.9 muestra la serie temporal Paper, 4.10 hace lo propio para la serie temporal Pigs, en 4.11 se detalla Cars y por último Milk en 4.12.

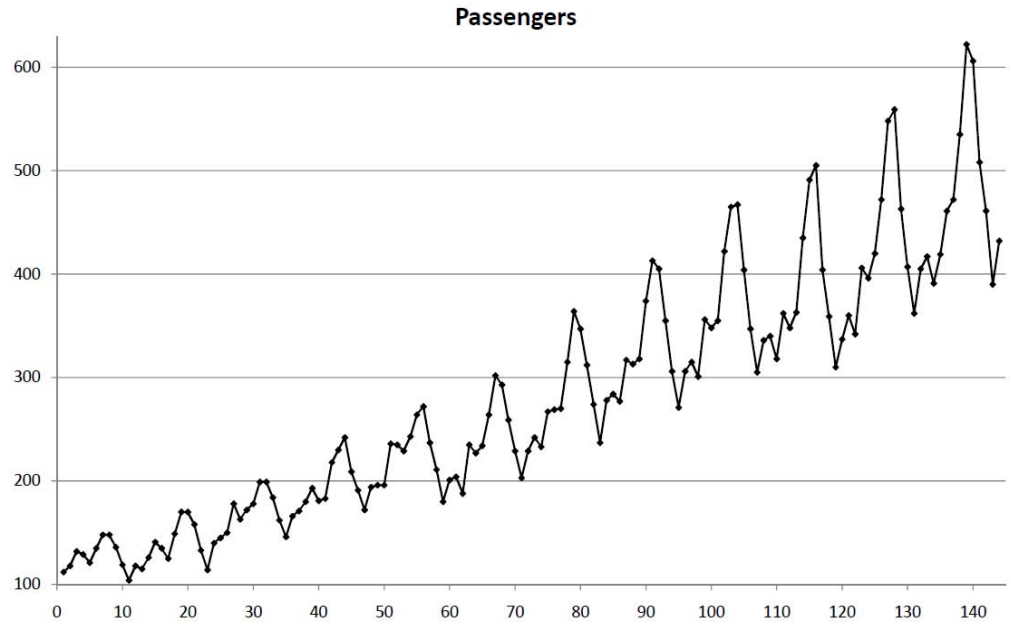


Figura 4.1: Serie temporal Passengers.

Casi todas las series temporales seleccionadas, excepto Mackey-Glass, son series temporales que provienen del mundo real. Los autores de este trabajo han considerado que los ingenieros orientados al control de procesos, manufacturación, producción industrial y de centros de investigación encontrarían más interesante este trabajo si los datos usados provinieran del mundo real. Además, los datos obtenidos del mundo real pueden sufrir por cuestiones externas tales como tormentas, terremotos, inundaciones, huelgas, guerras, avances tecnológicos, etc., lo que hace de estas series temporales más interesantes y difíciles de predecir.

La batería de pruebas que será usada en esta tesis doctoral consistirá en seis de las doce series temporales presentadas anteriormente, Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass. Las restantes seis series temporales serán usadas en experimentaciones específicas para complementar a las anteriormente citadas por razones que se detallarán más adelante.

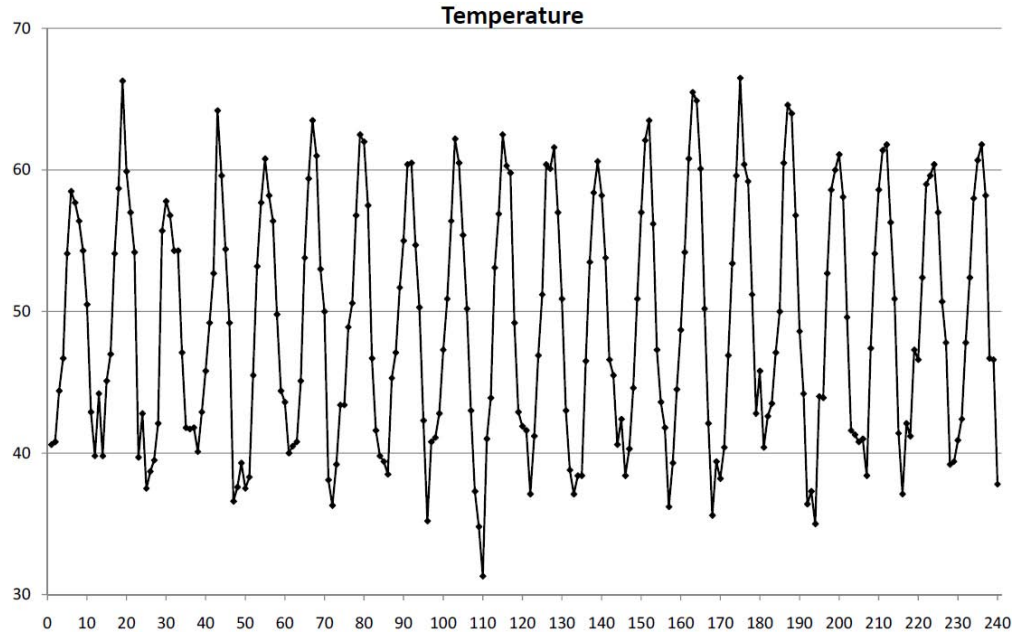


Figura 4.2: Serie temporal Temperature.

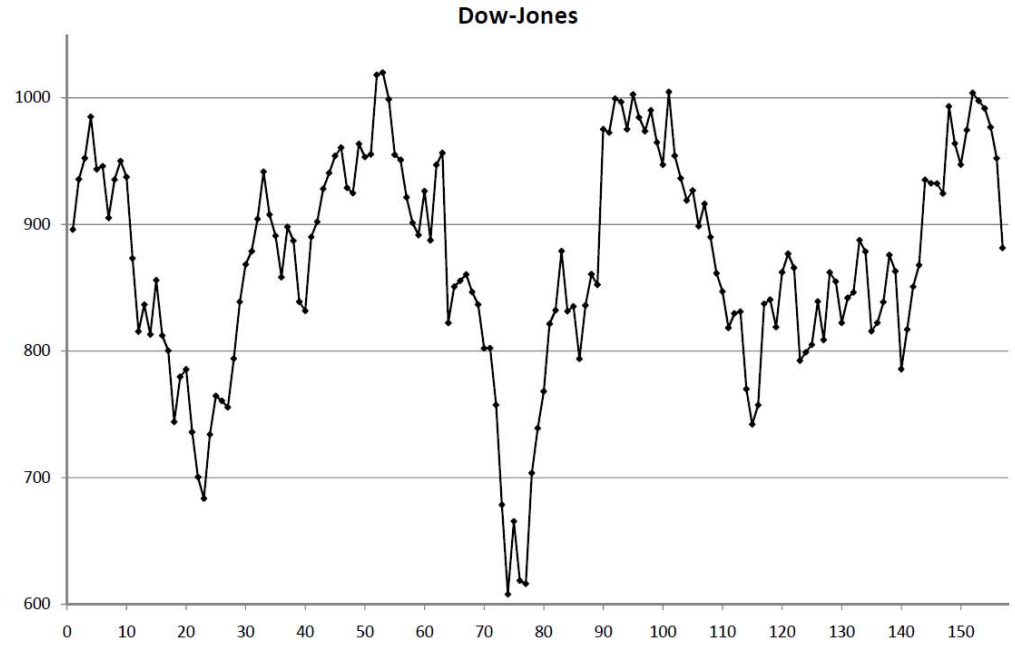


Figura 4.3: Serie temporal Dow-Jones.

Para evaluar el método de predicción presentado, cada serie temporal ha sido dividida inicialmente en dos conjuntos: muestras de entrada y muestras de salida.

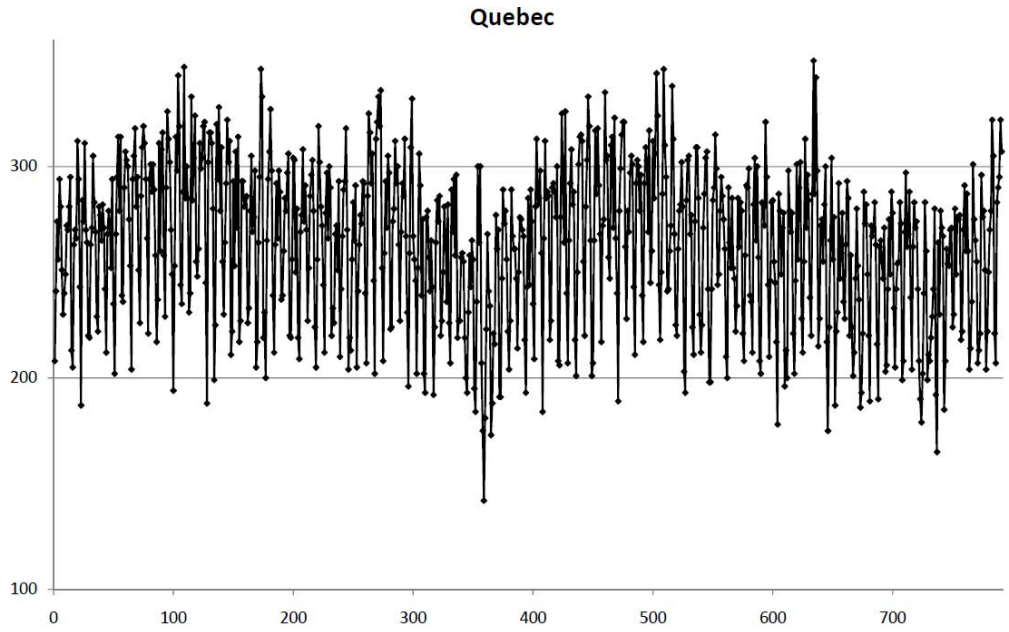


Figura 4.4: Serie temporal Quebec.

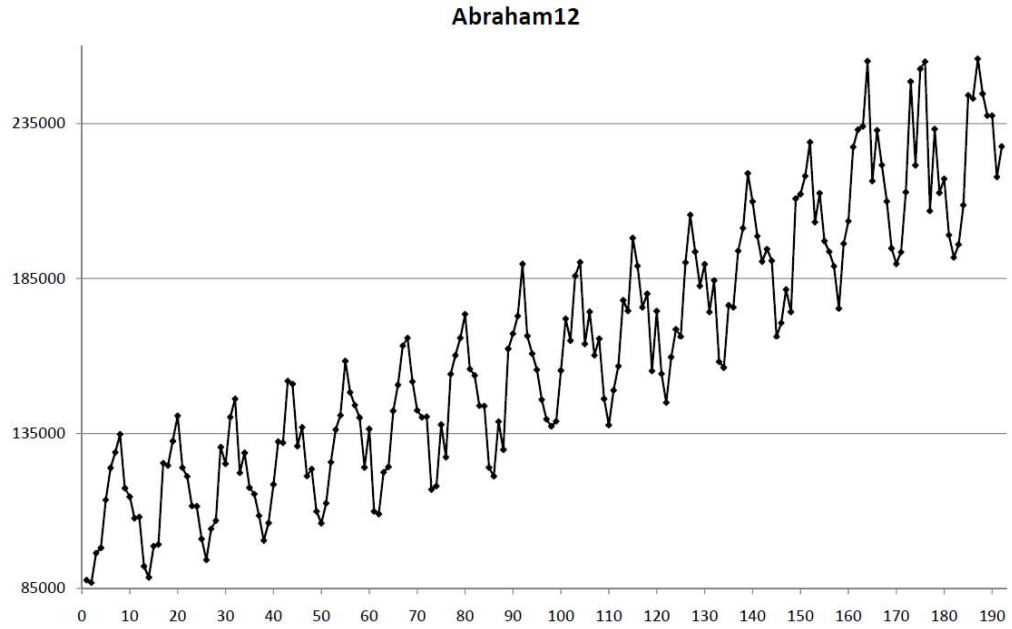


Figura 4.5: Serie temporal Abraham12.

Los datos de las muestras de entrada son usados para construir los modelos de predicción. En ADANN son estos datos los que sirven para obtener los subconjuntos de patrones de entrenamiento y validación como se explicó en la sección

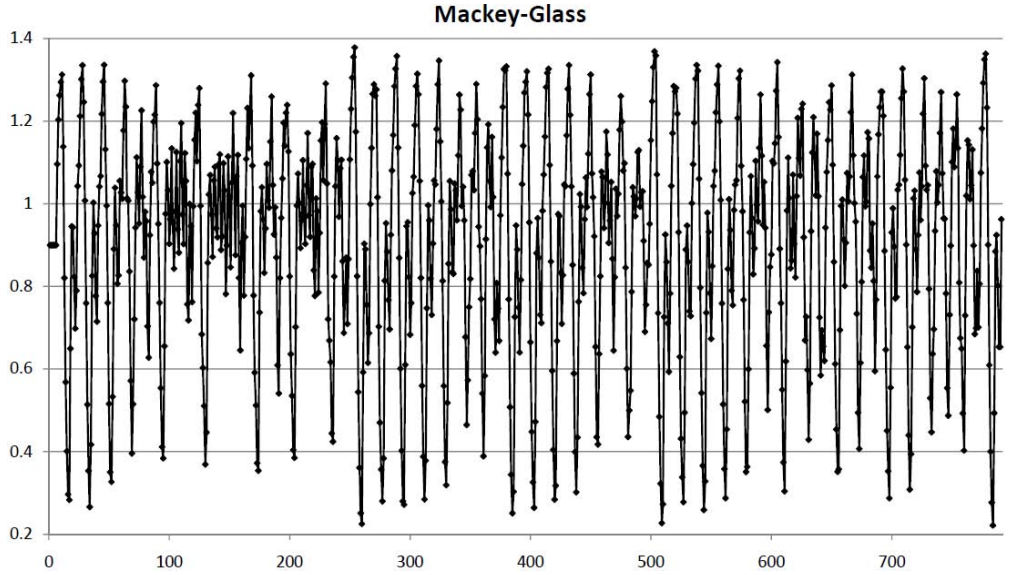


Figura 4.6: Serie temporal Mackey-Glass.

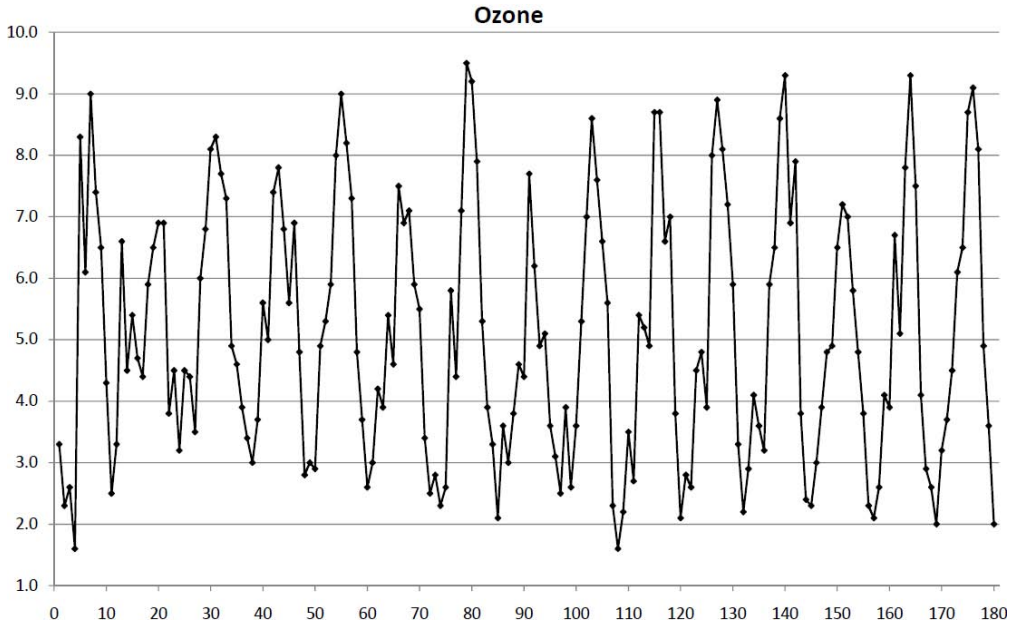


Figura 4.7: Serie temporal Ozone.

3.2. Las muestras de salida, también conocidas como conjunto de test, incluyen los valores mas recientes de la serie temporal y son usadas para evaluar las predicciones llevadas a cabo por nuestro sistema siguiendo los indicadores de error que presentaremos a continuación.

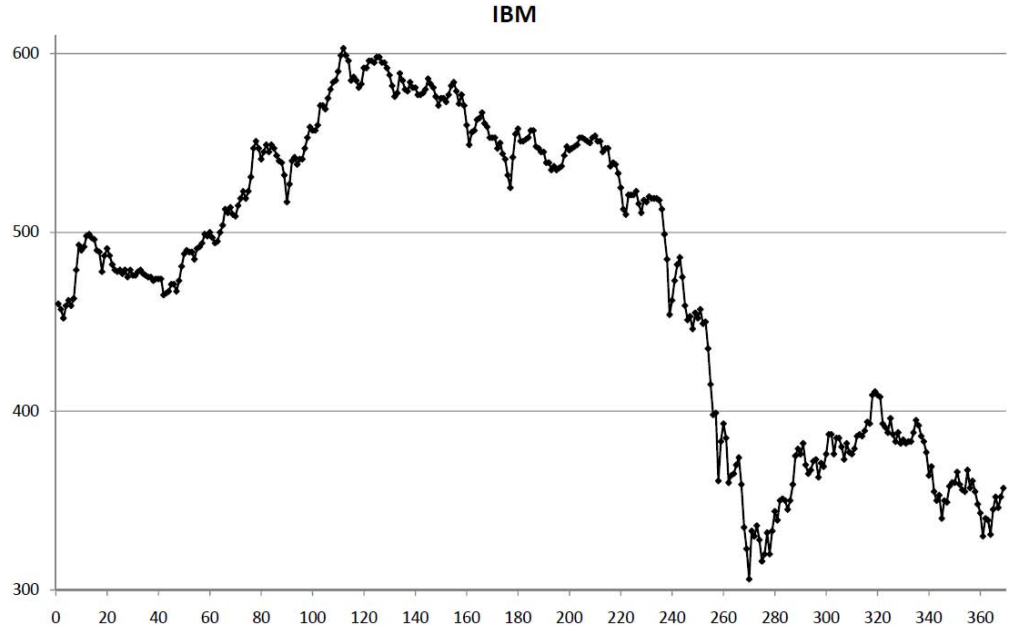


Figura 4.8: Serie temporal IBM.

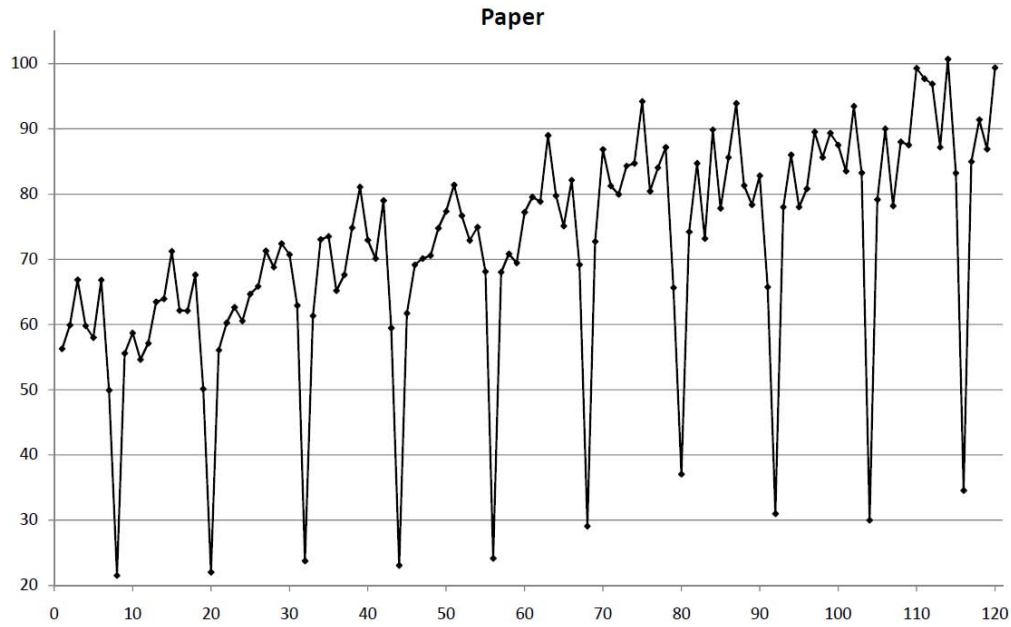


Figura 4.9: Serie temporal Paper.

La tabla 4.1 muestra el número de elementos, usados para generar un modelo, de las diferentes series temporales y los horizontes que hay que predecir para cada una, tomando como ejemplo las competiciones comentadas anteriormente

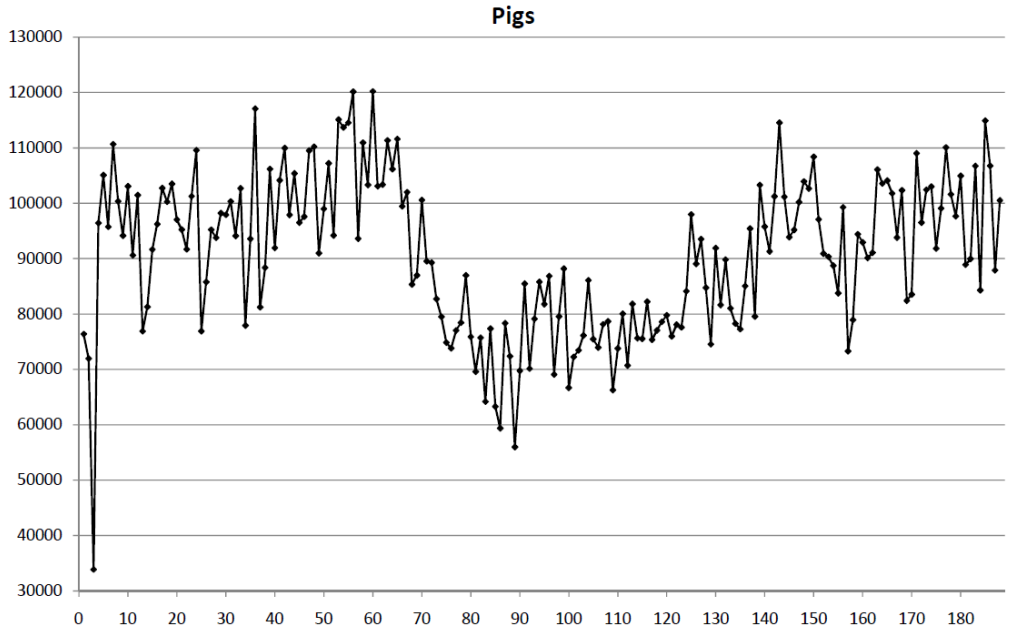


Figura 4.10: Serie temporal Pigs.

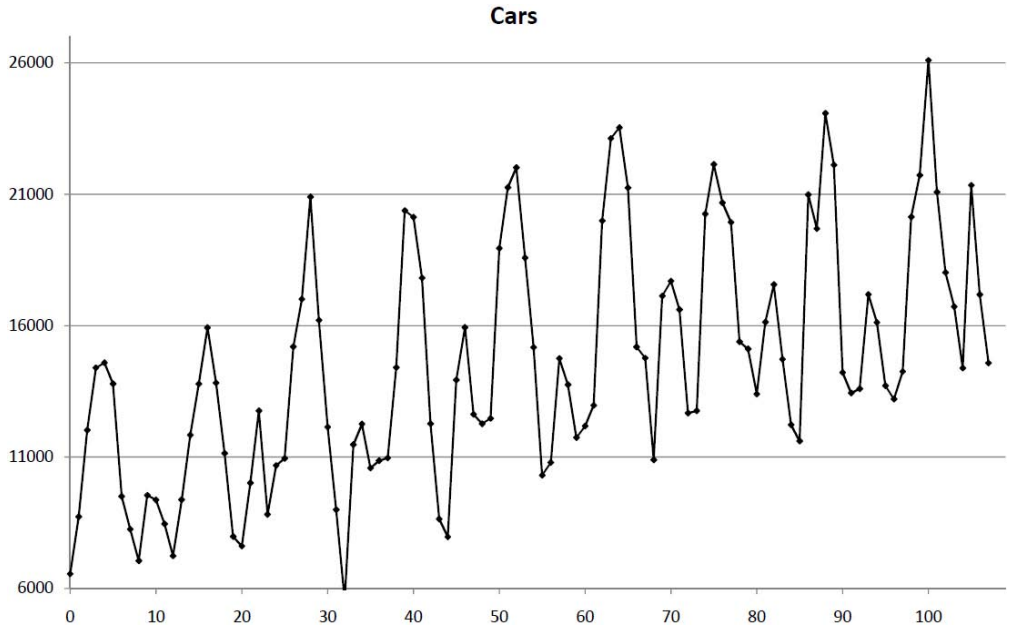


Figura 4.11: Serie temporal Cars.

(i.e. NN3, NN5 y NNGC1 [116]). En ellas, para las series temporales semejantes a Quebec y Mackey-Glass en número de elementos (series temporales largas con aproximadamente 700 u 800 elementos) había que predecir los 56 valores siguien-

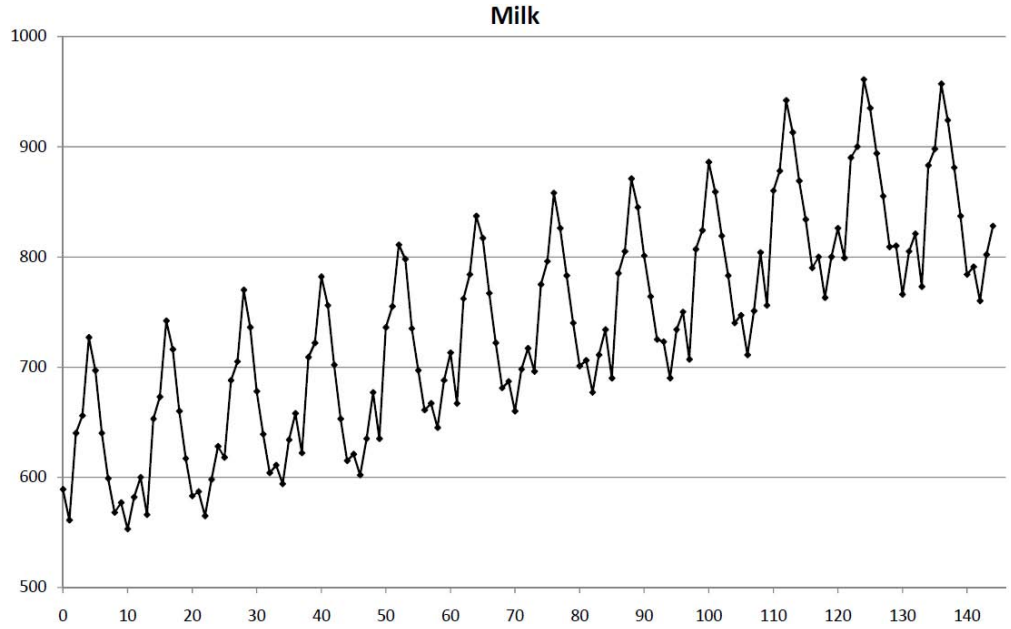


Figura 4.12: Serie temporal Milk.

tes, mientras que para las series más cortas tan solo era necesario predecir los 19 siguientes.

Tabla 4.1: Número de muestras de entrada y salida de las series temporales.

Series Temporales	Nº muestras de entrada	Nº muestras de salida
Passengers	125	19
Temperature	221	19
Dow-Jones	138	19
Quebec	735	56
Abraham12	173	19
Mackey-Glass	735	56
Ozone	161	19
IBM	350	19
Paper	101	19
Pigs	169	19
Cars	89	19
Milk	126	19

En el campo de la predicción de series temporales, los datos más antiguos tienden a ser menos relevantes que los datos más recientes. Así mismo, existen evidencias de que tener pocos datos es perjudicial a la hora de llevar a cabo una predicción. Dorn, en su revisión de las predicciones de población [204], concluyó que los predictores demográficos habían estado usando muy pocos datos. Smith y Sincich [205], haciendo uso de tres simples técnicas de extrapolación para la predicción de población de los EEUU, descubrieron que la precisión mejoraba conforme el número de años de datos incrementaba hasta diez. Aumentando las mediciones con las que trabajar más allá de diez años producía tan solo pequeñas ganancias en la predicción. Schnaars [206] concluyó que disponer de más datos no mejoraba significativamente la precisión en las extrapolaciones de las ventas anuales de productos de consumo. Armstrong demostró en [207] que cuanto mayor es el horizonte a predecir, más datos son necesarios.

Podemos observar entonces, que se han seleccionado series temporales de distinto tamaño. Aunque por norma general la comunidad investigadora dedicada a la predicción de series temporales premia el experimentar con series temporales cortas y por lo tanto que se disponga de pocos datos para predecir los futuros, en este trabajo hemos seleccionado al menos dos series temporales de mayor tamaño (Quebec y Mackey-Glass) con un mayor horizonte a predecir que el resto.

El rendimiento global de un modelo de predicción es evaluado por una medida de precisión, tales como el *Error Cuadrático Medio* (del inglés MSE) o como el *Error Simétrico Medio Porcentual Absoluto* (del inglés SMAPE) [208]. Las ecuaciones 4.1 y 4.2 muestran los errores MSE y SMAPE respectivamente:

$$MSE = \frac{1}{h} \sum_{t=T+1}^{T+h} (e_t^2) \tag{4.1}$$

$$SMAPE = \frac{1}{h} \sum_{t=T+1}^{T+h} \frac{|e_t|}{(|y_t| + |\hat{y}_t|)/2} \times 100\,\% \tag{4.2}$$

Donde $e_t = y_t - \hat{y}_t$, T es el instante actual en el que nos encontramos y h es el horizonte a predecir o el número de predicciones que se van a llevar a cabo. La variable y_t es el valor real de la serie y $\hat{y}_t$ el valor predicho.

En estas medidas, valores más bajos indican mejores predicciones. Históricamente, MSE (así como su variante de la raíz cuadrada) es una métrica muy popular dentro de la predicción de series temporales. El error SMAPE además tiene la ventaja de ser independiente de la escala. Aunque SMAPE fue originalmente propuesto en [207], la ecuación 4.2 adopta la variante propuesta en [209] ya que

esta versión no lleva a valores negativos, sino a valores comprendidos entre *0 %* y *200 %*. SMAPE es además el error usado en las competiciones de predicción de series temporales NN3, NN5 y NNGC1 [116]. Para esta tesis doctoral se ha optado por computar ambas medidas, MSE y SMAPE, a la hora de mostrar los resultados de las predicciones.

4.2. Tratamiento de los Datos

En esta sección de la tesis doctoral se explican las diferentes técnicas que se han estudiado sobre cómo tratar los datos con los que trabajamos para poder mejorar las predicciones en diferentes situaciones.

Por un lado mostraremos el único preprocesamiento de los datos que se lleva a cabo en el sistema, el reescalado de los valores conocidos de la serie temporal o proceso de normalización de datos. Este proceso se realiza una sola vez al principio de cada experimentación para cada serie temporal que se desea predecir. En este proceso, como veremos a continuación, tendremos en cuenta los posibles valores que puedan tomar los elementos futuros a la hora de reescalar los datos conocidos.

Pasaremos después a estudiar en detalle la experimentación llevada a cabo sobre las distintas funciones de evaluación presentadas en el apartado 3.3. Al final de esta experimentación se justificará cuál es la mejor función de evaluación para nuestro sistema.

A continuación, se mostrarán los resultados usando *"Shuffle"*. En este método los patrones con los que generamos los conjuntos de entrenamiento y validación se obtienen de manera desordenada del conjunto total de patrones. Mediante la experimentación pertinente podremos observar si este método ayuda a las RNA a aprender mejor que con el usado hasta el momento y presentado en la sección 3.2.

Por último, pasaremos a explicar en detalle la teoría de conjuntos (*"Ensembles"*). También nos centraremos en la validación cruzada y cómo hacer uso de ésta para poder predecir series temporales cortas (i.e. con pocos elementos) con nuestro sistema. Para ello, habrá que apoyarse en la teoría de conjuntos como veremos más adelante.

4.2.1. Normalización de los Datos

Antes de poder trabajar con RNA, los valores de las series temporales tienen que ser reescalados dentro de un rango numérico adecuado para ellas.

Diferentes procesos y curvas de aprendizaje de sistemas presentan una progresión temporal desde niveles bajos al principio, hasta alcanzar un valor máximo transcurrido un tiempo. Esta transición se produce en una región caracterizada por una fuerte aceleración intermedia, que puede ser descrita mediante la función sigmoidal. Su gráfica tiene una típica forma de *"S"* como se puede observar en la figura 4.13. Además, la función sigmoidal permite conservar la no linealidad de las RNA por lo que son adecuadas para resolver problemas en dominios no lineales como el aquí propuesto.

Se emplea entonces como función de transferencia una función sigmoidal, tanto en las posibles capas ocultas como en la de salida.

Debido a esto, el rango que se usará para normalizar los datos de la serie temporal será [0,1].

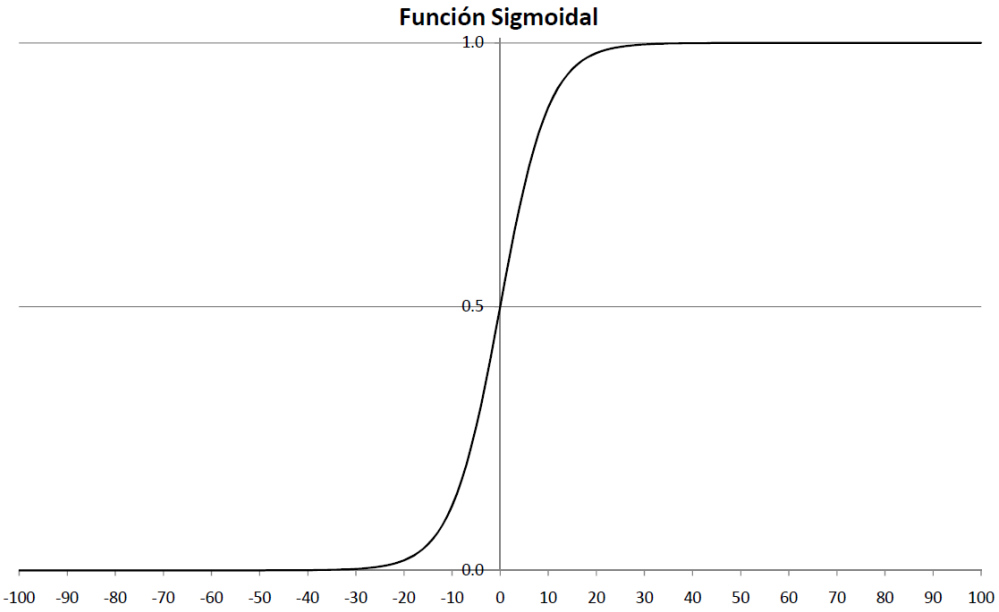


Figura 4.13: Curva logística de la función sigmoidal.

Sin embargo, a la hora de normalizar no podemos limitarnos solo al rango de valores que tienen los elementos conocidos de la serie temporal, sino que también tenemos que pensar que valores pueden tomar los elementos futuros que se van a predecir. Por lo tanto, los límites máximos y mínimos al normalizar (*max2norm* y *min2norm*) no podrán ser tan solo el máximo y mínimo de los valores conocidos de la serie temporal. Habrá que establecer un margen en caso de que los futuros elementos predichos fueran mayores o menores que los ya conocidos.

Este margen dependerá de un parámetro **Prct_inc**. En aquellos casos en los

que la serie temporal es estacionaria (no tiene tendencias) un valor de *10 %* para **Prct_inc** es más que suficiente, pero cuando trabajamos con series temporales con tendencia (ya sea creciente o decreciente), **Prct_inc** debería tener un valor de al menos *50 %*. El porqué de este mayor margen en las series con tendencia se debe a la experimentación previa que se ha llevado a cabo. Los posibles valores que puede tomar el parámetro del sistema **Prct_inc** se muestran en la tabla 4.2.

Tabla 4.2: Valor del parámetro **Prct_inc**.

Tipo de serie temporal	Valor de **Prct_inc**
Estacionaria	10 %
No Estacionaria	50 %

Así, en caso de que los valores de una serie temporal crezcan o decrezcan, el sistema podrá tener un margen suficientemente grande al predecir para hacer frente a posibles valores futuros mayores o menores que los ya conocidos y que estos estén también dentro del rango numérico [0,1]. En 4.3 se muestra cómo obtener los límites máximos y mínimos al normalizar (*max2norm* y *min2norm*).

$$\begin{aligned} max2norm &= max + (Prct\_inc \cdot (max - min)) \\ min2norm &= min - (Prct\_inc \cdot (max - min)) \end{aligned} \tag{4.3}$$

Una vez que se obtienen todos los valores normalizados de la serie temporal, se pasa a generar los conjuntos de patrones de los que va a hacer uso la RNA (entrenamiento y validación). El conjunto de test será aquel formado por los futuros valores de la serie temporal que tienen que ser predichos y por lo tanto desconocidos para la RNA.

4.2.2. Función de Evaluación

Como ya comentamos en la sección 2.9.1, se ha estudiado que aquellas funciones de evaluación en las que se usa tan sólo el error de validación generalizan mejor [123]. En este apartado, intentaremos demostrarlo presentando tres funciones de evaluación diferentes y mostrando empíricamente cual es la mejor opción.

Para llevar cabo la experimentación ha sido seleccionada una batería de series temporales formada por Passengers, Temperature, Dow-Jones, Quebec, Abraham12, y Mackey-Glass.

Al hacer uso de esta función de evaluación, se puede observar en las columnas de la tabla 4.3 y de la tabla 4.4 los resultados obtenidos para los pesos establecidos:

- $1{,}0$ para validación y $0{,}0$ para validación II
- $0{,}5$ para validación y $0{,}5$ para validación II
- $0{,}0$ para validación y $1{,}0$ para validación II

En este último caso el valor de adecuación será equivalente al dado únicamente por el error de validación II y el primero de los casos equivale a la función de evaluación que se usó inicialmente.

Los valores predichos son comparados con los valores reales de las respectivas series temporales y se usarán los errores anteriormente presentados (MSE y SMAPE) para evaluar las pruebas que se llevarán a cabo a continuación. En la tabla 4.3 se muestran los errores SMAPE obtenidos para cada serie temporal y en la tabla 4.4 los errores MSE asociados a sus respectivas predicciones. Estos resultados son obtenidos a partir de la predicción llevada a cabo por el mejor individuo de la última generación que proporciona el sistema ADANN como solución.

Tabla 4.3: Error SMAPE obtenido con diferentes funciones de evaluación.

Series Temporales	1,0 - 0,0	0,5 - 0,5	0,0 - 1,0
Passengers	**3,05**	3,82	5,37
Temperature	**3,79**	3,97	4,05
Dow-Jones	4,76	**4,71**	5,47
Quebec	12,12	**11,62**	11,84
Abraham12	**5,53**	7,08	7,97
Mackey-Glass	8,67	8,78	**6,16**
Media	**6,32**	6,66	6,81
Mediana	**5,14**	5,89	5,81

Como podemos observar en las tablas 4.3 y 4.4, la función de evaluación propuesta por Schaffer et al. [123] ($1{,}0$ para validación y $0{,}0$ para validación II) obtiene mejores resultados en tres de las seis series temporales usando la medida de error SMAPE e iguala a la función de evaluación *"mixta"* ($0{,}5$ para validación y $0{,}5$ para validación II) cuando medimos el error con MSE. Además, en términos generales esta función de evaluación (1,0 - 0,0) obtiene de media y como mediana

Tabla 4.4: Error MSE obtenido con diferentes funciones de evaluación.

Series Temporales	1,0 - 0,0	0,5 - 0,5	0,0 - 1,0
Passengers	$\mathbf{0{,}59}\times\mathbf{10}^{-3}$	$0{,}77\times10^{-3}$	$0{,}14\times10^{-2}$
Temperature	$\mathbf{0{,}31}\times\mathbf{10}^{-2}$	$\mathbf{0{,}28}\times\mathbf{10}^{-2}$	$0{,}34\times10^{-2}$
Dow-Jones	$0{,}10\times10^{-1}$	$\mathbf{0{,}10}\times\mathbf{10}^{-1}$	$0{,}13\times10^{-1}$
Quebec	$0{,}21\times10^{-1}$	$\mathbf{0{,}22}\times\mathbf{10}^{-1}$	$0{,}21\times10^{-1}$
Abraham12	$\mathbf{0{,}17}\times\mathbf{10}^{-2}$	$0{,}29\times10^{-2}$	$0{,}37\times10^{-2}$
Mackey-Glass	$0{,}36\times10^{-2}$	$0{,}44\times10^{-2}$	$\mathbf{0{,}18}\times\mathbf{10}^{-2}$
Media	$\mathbf{0{,}66}\times\mathbf{10}^{-2}$	$0{,}71\times10^{-2}$	$0{,}78\times10^{-2}$
Mediana	$\mathbf{0{,}33}\times\mathbf{10}^{-2}$	$0{,}36\times10^{-2}$	$0{,}35\times10^{-2}$

mejores resultados en ambos errores, solo igualada por la función *"mixta"* (0,5 - 0,5) en la media calculada con MSE.

Si nos fijamos en aquellas series temporales consideradas como largas o con un grupo nutrido de elementos (Quebec y Mackey-Glass), podemos observar que no queda del todo claro que función de evaluación debería ser usada. En Mackey-Glass se consigue una mejora al usar la función de evaluación en donde sólo se tiene en cuenta el error del conjunto validación II (0,0 - 1,0) consiguiendo un error SMAPE de *6,16 %*. Mientras que para Quebec el mejor resultado es obtenido con la función *"mixta"* (0,5 - 0,5).

Queda demostrado y corroborado que la mejor función de evaluación para este dominio es la propuesta en [123] y consiste en el error dado por un único conjunto, el de validación, siendo esta función la que será usada a lo largo de esta tesis doctoral.

4.2.3. *"Shuffle"*

"Shuffle" (del inglés barajar) se refiere a un método propuesto aplicado al sistema ADANN. Como ya se explicó en la sección 3.2 del capítulo anterior, para entrenar una RNA obtenida a partir de su correspondiente cromosoma, se genera primero su conjunto total de patrones y luego se divide en dos subconjuntos diferentes, entrenamiento y validación. El subconjunto de patrones de entrenamiento será usado para entrenar la RNA y el de validación para estimar la capacidad de generalización de dicha RNA.

Hasta ahora, los subconjuntos de entrenamiento y validación se obtienen de

manera secuencial, es decir, la primera parte ($x\,\%$) del conjunto total de patrones se usan para entrenar y el resto de los patrones para formar el subconjunto de validación, procedimiento que se mostró en la figura 3.2.

Sin embargo, en este nuevo método el proceso para dividir el conjunto total de patrones consistirá en obtener dichos subconjuntos de entrenamiento y validación de una manera aleatoria. Así, diferentes puntos de la serie temporal serán usados para entrenar la RNA y además otros puntos de la serie temporal, tomados también al azar, serán usados para validar dicha RNA (figura 4.14). Por ello, con la finalidad de intentar mejorar las predicciones que se lleven a cabo, en *"Shuffle"* se tomará el $x\,\%$ del conjunto total de patrones para formar el subconjunto de entrenamiento de una forma aleatoria y el subconjunto de validación será el resto de patrones, tomados también de forma aleatoria.

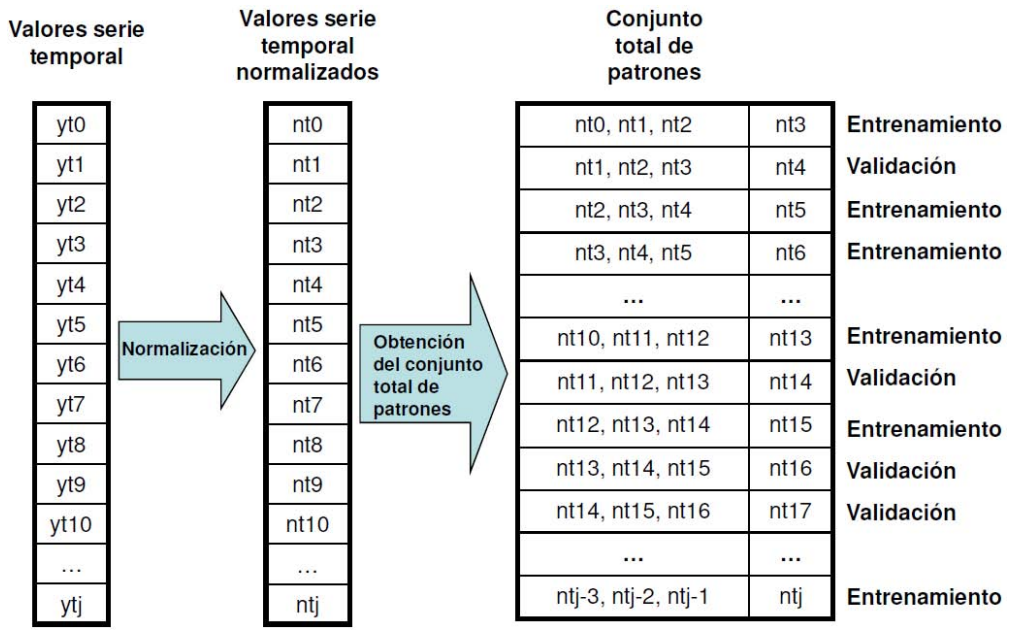


Figura 4.14: Proceso para obtener los subconjuntos de entrenamiento y validación con *"Shuffle"*.

Experimentación y resultados

Al igual que en la sección anterior, para llevar cabo la experimentación con "Shuffle" haremos uso de las series temporales Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass.

Se van a comparar ambos modos de generar los subconjuntos de entrenamien-

to y validación, el secuencial y el aleatorio (*"Shuffle"*). Los valores predichos son comparados con los valores reales de las respectivas series temporales y se usan los errores anteriormente presentados (MSE y SMAPE) para evaluar las pruebas que se llevarán a cabo.

En la tabla 4.5 se muestran los resultados obtenidos para cada una de las series temporales. En dicha tabla se pueden observar dos columnas principales, una primera donde se presentan los datos sin aplicar *"Shuffle"* y la segunda aplicándolo. Dentro de cada una de estas columnas principales se han generado dos más para presentar los resultados con los dos errores seleccionados (SMAPE y MSE). Como ya se comentó con anterioridad, estos resultados son obtenidos a partir de la predicción llevada a cabo por la mejor de las RNA de la última generación que proporciona el sistema ADANN como solución.

Tabla 4.5: SMAPE y MSE para las predicciones de Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con y sin *"Shuffle"*.

	Sin *"Shuffle"*		Con *"Shuffle"*	
	% SMAPE	MSE	% SMAPE	MSE
Passengers	**3,05**	$\mathbf{0{,}59\times10^{-3}}$	8,75	$0{,}29\times10^{-2}$
Temperature	**3,79**	$\mathbf{0{,}31\times10^{-2}}$	4,88	$0{,}42\times10^{-2}$
Dow-Jones	**4.76**	$\mathbf{0{,}10\times10^{-1}}$	5.83	$0{,}15\times10^{-1}$
Quebec	12,12	$0{,}21\times10^{-1}$	**9,21**	$\mathbf{0{,}13\times10^{-1}}$
Abraham12	5,53	$0{,}17\times10^{-2}$	**4,98**	$\mathbf{0{,}16\times10^{-2}}$
Mackey-Glass	8,67	$0{,}36\times10^{-2}$	**1,81**	$\mathbf{0{,}16\times10^{-3}}$
Media	6,32	$0{,}66\times10^{-2}$	**5,91**	$\mathbf{0{,}61\times10^{-2}}$
Mediana	**5,14**	$\mathbf{0{,}33\times10^{-2}}$	5,40	$0{,}35\times10^{-2}$

Se puede ver en la tabla 4.5 que el uso de *"Shuffle"* no consigue unas mejoras muy significativas aunque estas existan. Tan solo en la mitad de los casos planteados, el uso de *"Shuffle"* mejora los resultados.

Por otro lado, si observamos la tabla 4.5 y calculamos la media aplicando *"Shuffle"* tan solo a aquellas series temporales con un mayor número de elementos (Quebec y Mackey-Glass) y no para las series temporales cortas (Passengers, Temperature, Dow-Jones y Abraham12), el error SMAPE medio que se obtendría es de *4,69 %* miestras que la mediana sería de *4,27 %*.

Cabe destacar la importante mejora obtenida en Mackey-Glass con un error SMAPE de *1,81 %* usando este nuevo método, lo que significa que los valo-

res predichos son casi idénticos a los reales. En general, la media obtenida con *"Shuffle"* es mejor que cuando no se aplica dicho método, aunque no así la mediana.

Si observamos las series temporales cortas Passengers, Temperature y Dow-Jones, los resultados obtenidos con *"Shuffle"* son peores.

Una posible explicación a este comportamiento es por tener un número tan limitado de elementos (de 125 a 221). Si se dispone de menos elementos, menor número de patrones se podrán formar. Entonces si disponemos de pocos patrones y la obtención de los subconjuntos de patrones de entrenamiento y validación se lleva a cabo de una manera aleatoria y desordenada, muy pocos de los patrones usados para ajustar las conexiones de los pesos de las RNA corresponderán a valores consecutivos de las series temporales. Por lo tanto, la relación entre las entradas y salidas puede ser más difícil de aprender cuando existen un menor número de patrones para el aprendizaje y éstos apenas son consecutivos. Entonces la idea de aprender de puntos diferentes de la serie temporal es más difícil de llevar a cabo.

Sin embargo, Abraham12 tan solo dispone de 173 elementos conocidos y el uso de *"Shuffle"* ha mejorado su predicción. No conocemos una explicación lógica o patrón de semejanza entre estas series temporales para poder justificar porqué el uso de *"Shuffle"* obtiene mejores resultados para algunas series temporales y no para otras.

Por lo tanto, y aunque el uso de *"Shuffle"* ha mejorado el resultado en algunos casos, optamos por no seguir usándolo durante el resto de esta tesis doctoral ya que desconocemos el patrón a seguir para seleccionar a qué series temporales se podría aplicar y a cuales es mejor no hacerlo. Sería necesario un estudio más profundo, posponiendo esta cuestión a trabajos futuros.

4.2.4. Validación cruzada y *"Ensembles"*

El objetivo de esta sección es intentar mejorar la predicción de aquellas series temporales que no disponen de un gran número de elementos. Para solventar este problema haremos uso de la validación cruzada.

Para llevar a cabo la experimentación con validación cruzada vamos a utilizar las series temporales Passengers, Temperature y Dow-Jones, ya que son aquellas cuyas predicciones no mejoraron con *"Shuffle"* y disponen de pocos elementos. Además, y considerando que tres series temporales no es una batería de prueba suficientemente grande, se ha optado por añadir otras tres series temporales con poco elementos (IBM, Ozone y Paper) [203].

Los valores predichos son comparados con los valores reales de las respectivas series temporales y hacemos uso de las medidas de error SMAPE y MSE para evaluar las pruebas que se llevarán a cabo.

Cabe destacar que la experimentación se ha llevado a cabo usando los dos tipos de validación cruzada explicadas con anterioridad, ponderada y sin ponderar. Además, se han utilizado los cuatro tipos de *"Ensembles"* (media, mediana, exponencial y CLBR) para obtener una única salida unificada. Por lo tanto, tendremos que analizar ocho tablas, cuatro correspondientes a los *"Ensembles"* con validación cruzada sin ponderar y otras cuatro tablas con dichos *"Ensembles"* pero esta vez usando validación cruzada ponderada. Se pretende así estudiar dos cuestiones: cuál de los métodos de *"Ensembles"* es mejor para nuestro dominio; y cuál de las validaciones cruzadas obtiene mejores resultados, la ponderada o sin ponderar.

Las tablas muestran en sus columnas el número de subconjuntos en los que se han dividido los patrones para realizar la validación cruzada (mínimo 2 y máximo 8). Cada fila representa el error SMAPE obtenido por cada serie temporal. Se muestra en la última fila de la tabla la media y la mediana de cada serie temporal según el número de subconjuntos usado. La columna etiquetada con un *"0"*, representa el error obtenido sin hacer uso de validación cruzada, es decir, el error obtenido por la aproximación del sistema explicada en la sección 3.6.1.

Las siguientes tablas muestran:

- Tabla 4.6: representa la validación cruzada sin ponderar usando la media como método de combinación.
- Tabla 4.8: representa la validación cruzada sin ponderar usando la mediana como método de combinación .
- Tabla 4.10: muestra el método de validación cruzada sin ponderar usando la exponencial como método de combinación.
- Tabla 4.12: es para la validación cruzada sin ponderar usando CLBR como método de combinación.
- Tabla 4.14: representa la validación cruzada ponderada usando la media como método de combinación.
- Tabla 4.16: representa la validación cruzada ponderada usando la mediana como método de combinación.
- Tabla 4.18: muestra la validación cruzada ponderada usando la exponencial como método de combinación.

- Tabla 4.20: representa la validación cruzada ponderada usando CLBR como método de combinación.

Estas mismas combinaciones de resultados y en ese mismo orden pero haciendo uso del error MSE se recogen en las tablas 4.7, 4.9, 4.11, 4.13, 4.15, 4.17, 4.19 y finalmente 4.21. Cabe destacar que para cada una de las posibles combinaciones, se muestra primero su error SMAPE seguido de su error MSE.

Tabla 4.6: Validación cruzada sin ponderar usando la media como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	18,28	13,53	9,71	7,48	13,32	3,33	6,36
Temperature	4,31	**3,39**	**3,43**	**3,63**	**3,73**	**3,96**	**3,83**	**3,56**
Dow-Jones	6,66	**6,28**	6,81	7,67	7,19	**5,14**	7,04	**6,19**
Ozone	16,60	**14,06**	**14,30**	**14,80**	**14,46**	**15,11**	**14,08**	**16,21**
IBM	5,10	**4,69**	**3,18**	**4,04**	**3,46**	**3,39**	**2,10**	10,80
Paper	8,15	8,80	9,76	8,56	9,17	8,43	10,81	9,61
Media	7,32	9,25	8,50	8,07	7,58	8,23	**6,87**	8,79
Mediana	5,88	7,54	8,28	8,11	7,33	6,78	**5,43**	7,98

Tabla 4.7: Validación cruzada sin ponderar usando la media como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	1,02	0,64	0,35	0,23	0,64	0,07	0,18
Temperature ($\times 10^{-2}$)	0,36	**0,25**	**0,25**	**0,25**	**0,29**	**0,30**	**0,34**	**0,25**
Dow-Jones ($\times 10^{-2}$)	2,24	**1,83**	**2,11**	2,84	2,27	**1,18**	**2,15**	**1,55**
Ozone ($\times 10^{-2}$)	1,25	**0,89**	**0,90**	**0,99**	**1,02**	**1,03**	**0,93**	**1,12**
IBM ($\times 10^{-2}$)	0,44	**0,37**	**0,15**	**0,24**	**0,18**	**0,17**	**0,07**	1,32
Paper ($\times 10^{-2}$)	0,31	0,32	0,40	**0,30**	0,37	0,32	0,40	0,35
Media ($\times 10^{-2}$)	0,77	0,78	**0,74**	1,25	**0,72**	**0,60**	**0,66**	0,79
Mediana ($\times 10^{-2}$)	0,40	0,63	0,52	0,67	**0,33**	0,48	**0,37**	0,73

En las tablas, se indica en negrita aquellos valores donde se ha obtenido mejor resultado al hacer uso de validación cruzada, que sin usar validación cruzada (i.e.

Tabla 4.8: Validación cruzada sin ponderar usando la mediana como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	18,28	13,57	9,28	5,57	12,57	3,69	6,16
Temperature	4,31	**3,39**	**3,49**	**3,45**	**3,77**	**3,97**	**3,94**	**3,45**
Dow-Jones	6,66	**6,28**	6,78	7,79	7,64	**4,96**	8,20	7,18
Ozone	16,60	**14,06**	**14,01**	**14,58**	**15,30**	**14,06**	**14,78**	**16,15**
IBM	5,10	**4,69**	**3,30**	**3,95**	**3,35**	**3,55**	**2,81**	**3,42**
Paper	8,15	8,80	8,81	8,27	9,04	8,41	9,40	8,69
Media	7,32	9,25	8,33	7,89	7,45	7,92	**7,14**	7,51
Mediana	5,88	7,54	7,79	8,03	6,60	6,68	6,07	6,67

Tabla 4.9: Validación cruzada sin ponderar usando la mediana como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	1,02	0,65	0,33	0,16	0,59	0,07	0,17
Temperature ($\times 10^{-2}$)	0,36	**0,25**	**0,25**	**0,24**	**0,29**	**0,31**	**0,34**	**0,25**
Dow-Jones ($\times 10^{-2}$)	2,24	**1,84**	**2,18**	2,93	2,58	**1,09**	2,90	2,30
Ozone ($\times 10^{-2}$)	1,25	**0,89**	**0,83**	**0,95**	**0,96**	**0,92**	**0,98**	**1,11**
IBM ($\times 10^{-2}$)	0,44	**0,37**	**0,16**	**0,29**	**0,17**	**0,19**	**0,13**	**0,16**
Paper ($\times 10^{-2}$)	0,31	0,32	0,33	**0,29**	0,37	0,32	0,32	**0,31**
Media ($\times 10^{-2}$)	0,77	0,78	**0,73**	0,83	0,75	**0,57**	**0,77**	**0,71**
Mediana ($\times 10^{-2}$)	0,40	0,63	0,49	**0,31**	**0,33**	0,45	**0,33**	**0,28**

columna 0), con el número de subconjuntos descrito por su columna correspondiente y para la serie temporal descrita por su fila. En ocasiones también se puede observar el nombre de una serie temporal en negrita. Si es así, esa serie temporal habrá obtenido mejores resultados haciendo uso de validación cruzada en todos los casos, siendo indiferente en cuantos subconjuntos dividamos los datos.

Al observar las tablas con más detenimiento la primera conclusión que se puede sacar es que el método de validación cruzada ponderada, es decir, aquel que da más importancia a los subconjuntos de datos más recientes de la serie temporal, obtiene en general mejores resultados que el no ponderado.

Tabla 4.10: Validación cruzada sin ponderar usando la exponencial como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	18,94	12,93	10,99	6,97	12,60	**3,06**	5,96
Temperature	4,31	**3,39**	**3,50**	**3,59**	**3,76**	**3,82**	**3,81**	**3,51**
Dow-Jones	6,66	**6,40**	6,89	8,56	**5,18**	**5,43**	7,31	**6,31**
Ozone	16,60	**15,45**	**14,42**	**14,54**	**15,79**	**15,27**	**13,96**	**16,18**
IBM	5,10	**3,92**	**3,17**	**3,95**	**3,51**	**3,41**	**2,17**	11,12
Paper	8,15	8,30	9,36	8,38	9,38	8,32	10,86	9,90
Media	7,32	9,40	8,38	8,34	7,43	8,14	**6,86**	8,83
Mediana	5,88	7,35	8,12	8,47	6,07	6,87	**5,56**	8,10

Tabla 4.11: Validación cruzada sin ponderar usando la exponencial como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	1,08	0,60	0,43	0,21	0,58	**0,06**	0,16
Temperature ($\times 10^{-2}$)	0,36	**0,25**	**0,26**	**0,25**	**0,29**	**0,27**	**0,33**	**0,24**
Dow-Jones ($\times 10^{-2}$)	2,24	**1,92**	**2,24**	3,42	**1,29**	**1,30**	2,38	**1,64**
Ozone ($\times 10^{-2}$)	1,25	**1,13**	**0,94**	**0,98**	**1,01**	**1,08**	**0,94**	**1,12**
IBM ($\times 10^{-2}$)	0,44	**0,18**	**0,15**	**0,23**	**0,18**	**0,17**	**0,07**	1,44
Paper ($\times 10^{-2}$)	0,31	**0,27**	0,36	**0,29**	0,38	0,32	0,41	0,36
Media ($\times 10^{-2}$)	0,77	0,80	**0,75**	0,93	**0,56**	**0,62**	**0,69**	0,82
Mediana ($\times 10^{-2}$)	0,40	0,67	0,48	**0,36**	**0,33**	0,45	**0,37**	0,74

Si nos fijamos por ejemplo en las tablas 4.12 y 4.20, al usar validación cruzada ponderada, un número mayor de series temporales obtienen en todos los casos mejores resultados de predicción comparado con el método tradicional sin validación cruzada (i.e. columna 0), sin importar el número de subconjuntos con el que trabajemos.

Además, se puede observar también como en la tabla 4.20 existen más casos donde la media y la mediana (últimas dos filas de la tabla) obtenidas con validación cruzada ponderada son mejores que la media o la mediana obtenidas con el método de validación cruzada sin ponderar mostrados en la tabla 4.12.

Tabla 4.12: Validación cruzada sin ponderar usando CLBR como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	16,28	11,10	6,77	4,51	9,13	4,19	2,79
Temperature	4,31	**3,38**	**3,41**	**3,62**	**3,73**	**3,95**	**3,84**	**3,53**
Dow-Jones	6,66	**6,19**	6,72	7,58	7,11	**5,09**	6,96	**6,12**
Ozone	16,60	**14,02**	**14,36**	**14,37**	**15,34**	**15,02**	**13,64**	**15,79**
IBM	5,10	**4,65**	**3,18**	**4,03**	**3,45**	**3,38**	**2,10**	10,78
Paper	8,15	**7,85**	8,45	**7,92**	**8,09**	**7,51**	8,82	**8,03**
Media	7,32	8,73	7,87	7,38	**7,04**	7,35	**6,59**	7,84
Mediana	5,88	7,02	7,58	7,17	**5,81**	6,30	**5,57**	7,07

Tabla 4.13: Validación cruzada sin ponderar usando CLBR como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	0,84	0,46	0,20	0,10	0,35	0,07	0,47
Temperature ($\times 10^{-2}$)	0,36	**0,25**	**0,25**	**0,25**	**0,29**	**0,29**	**0,33**	**0,25**
Dow-Jones ($\times 10^{-2}$)	2,24	**1,78**	**2,09**	2,80	**2,18**	**1,13**	**2,12**	**1,49**
Ozone ($\times 10^{-2}$)	1,25	**0,89**	**0,90**	**0,97**	**1,01**	**0,99**	**0,95**	**1,10**
IBM ($\times 10^{-2}$)	0,44	**0,35**	**0,15**	**0,24**	**0,18**	**0,17**	**0,07**	1,33
Paper ($\times 10^{-2}$)	0,31	**0,25**	**0,29**	**0,22**	**0,26**	**0,20**	**0,25**	**0,20**
Media ($\times 10^{-2}$)	0,77	**0,72**	**0,69**	0,78	**0,67**	**0,52**	**0,63**	0,80
Mediana ($\times 10^{-2}$)	0,40	0,59	**0,37**	**0,24**	**0,27**	**0,32**	**0,29**	0,78

Aunque por lo general en casi todos los casos existen tres series temporales (Temperature, Ozone e IBM) en los que siempre se mejoran los resultados sin importar el número elegido para dividir el conjunto de patrones, también es cierto que en el caso de Passengers es prácticamente imposible mejorar sus resultados. Esto podría estar justificado porque Passengers, junto a Paper, son las dos únicas series temporales con tendencia (ascendente). Y si para Passengers es imposible mejorar los resultados, Paper tan solo obtiene mejoras claras con validación cruzada ponderada y método de combinación CLBR.

Si juntáramos todos los errores SMAPE medios obtenidos con cada método

Tabla 4.14: Validación cruzada ponderada usando la media como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	13,37	13,30	8,07	13,87	9,61	8,37	7,73
Temperature	4,31	**3,65**	**3,67**	**3,65**	**3,72**	**3,78**	**3,72**	**3,47**
Dow-Jones	6,66	7,09	7,12	**4,47**	**5,03**	**6,17**	**5,66**	**4,96**
Ozone	16,60	**15,41**	**16,18**	**15,30**	**13,91**	**15,84**	**15,88**	**15,36**
IBM	5,10	**2,82**	**2,49**	**2,34**	**3,31**	**2,41**	**2,19**	**2,12**
Paper	8,15	8,17	9,16	8,59	8,73	8,94	7,83	9,09
Media	7,32	8,42	8,65	**7,07**	8,10	7,79	**7,28**	**7,12**
Mediana	5,88	7,63	8,14	6,27	6,88	7,55	7,01	6,34

Tabla 4.15: Validación cruzada ponderarada usando la media como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	0,64	0,64	0,25	0,69	0,34	0,30	0,25
Temperature ($\times 10^{-2}$)	0,36	**0,28**	**0,27**	**0,29**	**0,29**	**0,27**	**0,26**	**0,24**
Dow-Jones ($\times 10^{-2}$)	2,24	2,30	2,34	**0,85**	**1,22**	**1,60**	**1,55**	**1,18**
Ozone ($\times 10^{-2}$)	1,25	**1,13**	**1,15**	**0,94**	**1,02**	**1,02**	**0,99**	**1,13**
IBM ($\times 10^{-2}$)	0,44	**0,10**	**0,10**	**0,07**	**0,17**	**0,08**	**0,06**	**0,06**
Paper ($\times 10^{-2}$)	0,31	**0,26**	0,36	**0,29**	**0,31**	0,32	**0,27**	**0,30**
Media ($\times 10^{-2}$)	0,77	0,78	0,81	**0,44**	**0,61**	**0,60**	**0,57**	**0,52**
Mediana ($\times 10^{-2}$)	0,40	0,46	0,50	**0,29**	0,50	**0,33**	**0,28**	**0,27**

y para cada serie temporal en una única tabla, el resultado sería la tabla 4.22. En ella "CV_{sp}" representa el método de validación cruzada sin ponderar y "CV_p" el ponderado, seguidos por una coma del método de *"ensemble"* utilizado. Además, por columnas tenemos el número de subconjuntos usados en la validación cruzada, desde 0 (sin validación cruzada) hasta 8. La última columna representa la media de los errores SMAPE de cada método para todos los subconjuntos de validación cruzada (de 2 a 8). La última fila representa la media de los errores obtenidos con cada número de subconjuntos.

Podemos observar en la tabla 4.22 en negrita aquellos errores que son menores

Tabla 4.16: Validación cruzada ponderada usando la mediana como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	13,37	13,20	9,03	14,15	9,66	7,98	7,13
Temperature	4,31	**3,65**	**3,62**	**3,67**	**3,72**	**3,71**	**3,75**	**3,37**
Dow-Jones	6,66	7,09	7,16	**3,97**	**5,18**	**5,56**	**5,88**	**5,70**
Ozone	16,60	**15,41**	**15,03**	**15,45**	**13,74**	**16,49**	**15,63**	**15,51**
IBM	5,10	**2,82**	**2,71**	**2,07**	**2,77**	**2,01**	**2,16**	**2,59**
Paper	8,15	8,17	8,45	8,70	8,45	8,81	8,25	8,81
Media	7,32	8,42	8,36	**7,15**	8,00	7,71	**7,28**	**7,19**
Mediana	5,88	7,63	7,80	6,33	6,81	7,18	6,93	6,41

Tabla 4.17: Validación cruzada ponderada usando la mediana como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	0,64	0,63	0,30	0,71	0,34	0,26	0,21
Temperature ($\times 10^{-2}$)	0,36	**0,28**	**0,27**	**0,30**	**0,32**	**0,26**	**0,28**	**0,24**
Dow-Jones ($\times 10^{-2}$)	2,24	2,30	2,39	**0,70**	**1,23**	**1,34**	**1,73**	**1,60**
Ozone ($\times 10^{-2}$)	1,25	**1,13**	**1,04**	**0,96**	**0,95**	**1,07**	**1,03**	**1,09**
IBM ($\times 10^{-2}$)	0,44	**0,10**	**0,12**	**0,07**	**0,12**	**0,06**	**0,06**	**0,09**
Paper ($\times 10^{-2}$)	0,31	**0,26**	0,34	**0,29**	**0,29**	0,33	**0,29**	**0,28**
Media ($\times 10^{-2}$)	0,77	0,78	0,79	**0,43**	**0,60**	**0,56**	**0,60**	**0,58**
Mediana ($\times 10^{-2}$)	0,40	0,46	0,48	**0,30**	0,51	**0,33**	**0,28**	**0,26**

que el *7,3 %* obtenido sin aplicar validación cruzada (columna 0).

Como ya comentamos con anterioridad y como se corrobora en la tabla 4.22 el método basado en validación cruzada ponderada obtiene de media siempre mejores resultados que su versión no ponderada.

Después de haber comprobado que el método de validación cruzada ponderada resulta más adecuado que el no ponderado, debemos definir un método de combinación o *"ensemble"*.

Llegados a este punto, existe un método de combinación que sobresale sobre

Tabla 4.18: Validación cruzada ponderada usando la exponencial como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	18,94	12,79	6,88	13,81	9,83	8,38	8,04
Temperature	4,31	**3,39**	**3,64**	**3,73**	**3,75**	**3,85**	**3,64**	**3,52**
Dow-Jones	6,66	**6,40**	7,29	**4,28**	**4,84**	**6,04**	**5,14**	**5,28**
Ozone	16,60	**15,45**	17,09	**15,37**	**14,29**	**15,59**	**15,00**	**15,36**
IBM	5,10	**3,92**	**2,67**	**2,29**	5,23	**2,42**	**2,20**	**2,45**
Paper	8,15	8,30	9,16	8,64	8,91	9,14	8,08	9,24
Media	7,32	9,40	8,77	**6,87**	8,47	7,81	**7,07**	**7,32**
Mediana	5,88	7,35	8,22	**5,58**	7,07	7,59	6,61	6,66

Tabla 4.19: Validación cruzada ponderada usando la exponencial como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	1,08	0,60	0,19	0,69	0,35	0,29	0,26
Temperature ($\times 10^{-2}$)	0,36	**0,25**	**0,26**	**0,29**	**0,30**	**0,28**	**0,26**	**0,25**
Dow-Jones ($\times 10^{-2}$)	2,24	**1,92**	2,42	**0,83**	**1,10**	**1,52**	**1,25**	**1,35**
Ozone ($\times 10^{-2}$)	1,25	**1,13**	1,26	**0,99**	**1,08**	**0,99**	**0,96**	**1,11**
IBM ($\times 10^{-2}$)	0,44	**0,18**	**0,12**	**0,07**	**0,38**	**0,08**	**0,06**	**0,08**
Paper ($\times 10^{-2}$)	0,31	**0,27**	0,34	**0,30**	0,32	0,34	**0,29**	**0,31**
Media ($\times 10^{-2}$)	0,77	0,80	0,83	**0,44**	**0,64**	**0,59**	**0,51**	**0,56**
Mediana ($\times 10^{-2}$)	0,40	0,67	0,47	**0,29**	0,53	**0,34**	**0,29**	**0,28**

el resto, este es la *Combinación Lineal Basada en Rango* (CLBR) [130]. Si nos fijamos en la tabla 4.22, el mejor resultado obtenido en la columna *"Método media"* es el dado por el método CLBR, donde tiene como media general un resultado de *7,0 %*.

Por último, con respecto al número de subconjuntos más adecuado con el que debería ser lanzado el sistema con validación cruzada, si nos fijamos en la última fila (*"Conjuntos media"*) de la tabla 4.22 donde se ha calculado la media de dichos subconjuntos, la columna con el número de subconjuntos 7 es la que mejor resultado obtiene de media (*7,0 %*).

Tabla 4.20: Validación cruzada ponderada usando CLBR como método de combinación y error SMAPE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers	3,18	11,45	10,73	5,08	10,16	5,45	4,07	3,39
Temperature	4,31	**3,66**	**3,67**	**3,64**	**3,72**	**3,75**	**3,69**	**3,45**
Dow-Jones	6,66	6,95	6,96	**4,46**	**4,92**	**6,07**	**5,58**	**4,91**
Ozone	16,60	**15,31**	**16,12**	**15,20**	**13,76**	**15,59**	**15,66**	**15,44**
IBM	5,10	**2,83**	**2,49**	**2,34**	**3,30**	**2,42**	**2,19**	**2,11**
Paper	8,15	**7,89**	8,41	**7,52**	**7,84**	**7,53**	**7,51**	8,82
Media	7,32	8,02	8,06	**6,37**	**7,28**	**6,80**	**6,45**	**6,35**
Mediana	5,88	7,42	7,68	**4,77**	6,38	**5,76**	**4,82**	**4,15**

Tabla 4.21: Validación cruzada ponderada usando CLBR como método de combinación y error MSE.

Series Temporales	0	2	3	4	5	6	7	8
Passengers ($\times 10^{-2}$)	0,06	0,50	0,46	0,12	0,42	0,13	0,10	**0,06**
Temperature ($\times 10^{-2}$)	0,36	**0,28**	**0,27**	**0,29**	**0,29**	**0,26**	**0,26**	**0,24**
Dow-Jones ($\times 10^{-2}$)	2,24	**2,21**	2,26	**0,85**	**1,17**	**1,55**	**1,50**	**1,14**
Ozone ($\times 10^{-2}$)	1,25	**1,13**	**1,17**	**0,94**	**1,05**	**0,99**	**1,00**	**1,08**
IBM ($\times 10^{-2}$)	0,44	**0,10**	**0,10**	**0,07**	**0,17**	**0,08**	**0,06**	**0,06**
Paper ($\times 10^{-2}$)	0,31	**0,21**	**0,27**	**0,20**	**0,21**	**0,20**	**0,18**	**0,23**
Media ($\times 10^{-2}$)	0,77	**0,73**	**0,75**	**0,41**	**0,55**	**0,53**	**0,51**	**0,46**
Mediana ($\times 10^{-2}$)	0,40	**0,39**	**0,36**	**0,24**	**0,35**	**0,23**	**0,22**	**0,23**

Pero si se quiere mantener un compromiso entre tiempo computacional y la exactitud en los resultados, recomendamos para este caso específico hacer uso de 4 subconjuntos para ahorrar coste computacional ya que la diferencia de tiempo entre 4 y 7 subconjuntos es prácticamente el doble y el resultado varía tan solo de *7,0 %* para 4 subconjuntos a *7,4 %* para 7 subconjuntos.

Por consiguiente, el método óptimo es el que emplea validación cruzada ponderada con CLBR como *"ensemble"* y hace uso de 4 subconjuntos para no consumir mucho tiempo computacional como ocurre con 7 subconjuntos.

Tabla 4.22: Resultados generales (media % SMAPE).

	Conjuntos								$Método_{media}$
Método	0	2	3	4	5	6	7	8	
CV_{sp}, **media**	7.3	9.3	8.5	8.1	7.6	8.2	**6.9**	8.8	8.2
CV_{sp}, **mediana**	7.3	9.3	8.3	7.9	7.5	7.9	**7.1**	7.5	7.9
CV_{sp}, **exponencial**	7.3	9.4	8.4	8.4	7.4	8.1	**6.9**	8.8	8.2
CV_{sp}, **CLBR**	7.3	8.7	7.9	7.4	**7.1**	7.4	**6.6**	7.8	7.5
CV_{p}, **media**	7.3	8.4	8.7	**7.1**	8.1	7.8	**7.3**	**7.1**	7.8
CV_{p}, **mediana**	7.3	8.4	8.4	**7.2**	8.0	7.7	**7.2**	**7.2**	7.7
CV_{p}, **exponencial**	7.3	9.4	8.8	**6.9**	8.5	7.8	**7.1**	**7.3**	8.0
CV_{p}, **CLBR**	7.3	8.0	8.1	**6.4**	**7.3**	**6.8**	**6.5**	**6.4**	**7.0**
$Conjuntos_{media}$	7.3	8.9	8.4	7.4	7.7	7.7	**7.0**	7.6	- -

Conclusiones

En este apartado de la tesis doctoral se han presentado dos métodos para intentar mejorar la predicción inicial dada por la primera aproximación del sistema (ADANN) explicada en el capítulo 3. Uno denominado (*"Shuffle"*) y otro para series temporales cortas o lo que es igual, con pocos elementos, basado en validación cruzada.

En esta sección se empezó estudiando qué función de evaluación es la que debería ser usada en nuestro sistema y se demostró empíricamente que la mejor opción es aquella función que hace uso tan sólo del conjunto de validación para obtener la capacidad de generalización de los individuos.

En este capítulo además, se ha presentado el primero de los enfoques de esta tesis doctoral. Este enfoque está orientado a los datos de entrada dados por la serie temporal que intentamos predecir y cómo tratar dichos datos para que podamos obtener unos resultados mejores. Para ello, se optó por llevar a cabo un nuevo método llamado *"Shuffle"* con el que barajar los patrones de entrada de las RNA y poder usar puntos ordenadas de forma aleatoria de la serie temporal para aprender y validar en lugar de usar el comienzo de la serie para aprender y el final de ésta para validar como se hace habitualmente. Los resultados no fueron del todo positivos, porque aunque mejorábamos en tres de las series temporales usadas como batería de pruebas, en otras tres los resultados obtenidos eran peores. Después de observar con detenimiento dichas series, no se ha podido llegar a una conclusión certera de porqué se da este comportamiento.

Por otro lado, ante el problema que se plantea en la predicción de series que tienen muy pocos elementos, como las de la competición internacional NNGC1 [116] con cerca de 30 elementos, se optó por llevar a cabo una validación cruzada y obtener así el máximo partido a dichos elementos. Para ello se dividió el conjunto de patrones en un número de subconjuntos (entre 2 y 8). Llegados a este punto apareció otro problema, si una misma topología de RNA se entrena con diferentes conjuntos de patrones, la arquitectura final resultante es diferente, dando lugar a una predicción diferente para cada arquitectura. Para solucionar este problema en el que una misma RNA con n subconjuntos de patrones genera n arquitecturas y predicciones diferentes, se hizo uso de la combinación de todas y cada una de las salidas dadas por estas arquitecturas para obtener una única solución (en este caso predicción) final. Se ha demostrado que la versión ponderada de validación cruzada obtiene mejores resultados y en cuanto al mejor método de combinación o *"Ensembles"*, se sugiere CLBR haciendo uso de 4 subconjuntos.

4.3. Algoritmos Evolutivos

En esta sección se muestran los resultados obtenidos por los diferentes algoritmos evolutivos explicados en la sección 3.6 y que han sido aplicados al sistema propuesto en esta tesis doctoral (ADANN).

Cada aproximación evolutiva (AG, ED y AED) ha sido ejecutada en cinco ocasiones, con un criterio de parada de 100 y 200 generaciones para cada una de las series temporales. Aquí se mostrarán los resultados medios de estas cinco ejecuciones.

En esta sección llevamos a cabo la predicción para cada una de las seis series temporales de la batería de pruebas presentada en la sección 4.1 y ya usada con anterioridad en las secciones 4.2.2 y 4.2.3. Para medir los errores obtenidos en la predicción se ha vuelto a optar por SMAPE (ecuación 4.2) y MSE (ecuación 4.1). A continuación mostramos los errores asociados a cada una, comparando con el conjunto de valores de test o valores futuros reales.

Los resultados se muestran en las tablas 4.23, 4.24, 4.25 y 4.26. Las columnas de estas tablas representan las series temporales y sus respectivos errores SMAPE y MSE asociados a las tres aproximaciones evolutivas usadas (AG, ED y AED), además de mostrar la media y mediana de dichos errores.

Podemos obtener diferentes conclusiones de estas cuatro tablas sobre la experimentación llevada a cabo. Como se puede observar al lanzar los experimentos para 100 generaciones en las tablas 4.23 y 4.25, aplicar ED en lugar de AG logra

Tabla 4.23: SMAPE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AG, ED y AED durante 100 generaciones.

Series Temporales	AG	ED	AED
Passengers	**3,18**	3,36	3,57
Temperature	4,31	**3,91**	3,98
Dow-Jones	**6,66**	8,19	6,81
Quebec	**12,64**	13,74	13,27
Abraham12	5,53	5,24	**4,87**
Mackey-Glass	8,67	**5,99**	6,27
Media	6,83	6,73	**6,46**
Mediana	6,09	5,61	**5,57**

Tabla 4.24: SMAPE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AG, ED y AED durante 200 generaciones.

Series Temporales	AG	ED	AED
Passengers	3,15	**3,12**	3,22
Temperature	4,24	**3,91**	3,94
Dow-Jones	6,31	5,81	**5,26**
Quebec	12,12	13,68	**10,96**
Abraham12	4,93	4,93	**4,49**
Mackey-Glass	8,04	3,74	**1,81**
Media	6,46	5,86	**4,94**
Mediana	5,62	4,42	**4,21**

un mejor resultado (SMAPE/MSE) en tres (Temperature, Abraham12 y Mackey-Glass) de las seis series temporales presentadas. En el caso de Mackey-Glass la mejora llega al *2,6 %*.

AED obtiene también mejores resultados que el AG en tres (Temperature, Abraham12 y Mackey-Glass) de las seis series temporales cuando los experimentos son lanzados para 100 generaciones. Mackey-Glass sigue siendo la serie en la que se obtiene una ganancia más pronunciada ya que su mejora del error con respecto al AG es de *2,4 %*.

En cuanto a ED comparado con AED, ED obtiene mejores resultados en las

Tabla 4.25: MSE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AG, ED y AED durante 100 generaciones.

Series Temporales	AG	ED	AED
Passengers	$\mathbf{0{,}61\times10^{-3}}$	$0{,}65\times10^{-3}$	$0{,}69\times10^{-3}$
Temperature	$0{,}36\times10^{-2}$	$\mathbf{0{,}29\times10^{-2}}$	$0{,}31\times10^{-2}$
Dow-Jones	$\mathbf{0{,}22\times10^{-1}}$	$0{,}32\times10^{-1}$	$0{,}24\times10^{-1}$
Quebec	$\mathbf{0{,}25\times10^{-1}}$	$0{,}27\times10^{-1}$	$0{,}26\times10^{-1}$
Abraham12	$0{,}18\times10^{-2}$	$0{,}17\times10^{-2}$	$\mathbf{0{,}16\times10^{-2}}$
Mackey-Glass	$0{,}36\times10^{-2}$	$\mathbf{0{,}18\times10^{-2}}$	$0{,}19\times10^{-2}$
Media	$\mathbf{0{,}95\times10^{-2}}$	$0{,}11\times10^{-1}$	$\mathbf{0{,}95\times10^{-2}}$
Mediana	$0{,}36\times10^{-2}$	$\mathbf{0{,}23\times10^{-2}}$	$0{,}25\times10^{-2}$

Tabla 4.26: MSE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AG, ED y AED durante 200 generaciones.

Series Temporales	AG	ED	AED
Passengers	$\mathbf{0{,}58\times10^{-3}}$	$\mathbf{0{,}58\times10^{-3}}$	$0{,}61\times10^{-3}$
Temperature	$0{,}35\times10^{-2}$	$\mathbf{0{,}29\times10^{-2}}$	$0{,}32\times10^{-2}$
Dow-Jones	$0{,}20\times10^{-1}$	$0{,}17\times10^{-1}$	$\mathbf{0{,}15\times10^{-1}}$
Quebec	$0{,}21\times10^{-1}$	$0{,}27\times10^{-1}$	$\mathbf{0{,}17\times10^{-1}}$
Abraham12	$0{,}17\times10^{-2}$	$0{,}14\times10^{-2}$	$\mathbf{0{,}13\times10^{-2}}$
Mackey-Glass	$0{,}31\times10^{-2}$	$0{,}64\times10^{-3}$	$\mathbf{0{,}16\times10^{-3}}$
Media	$0{,}83\times10^{-2}$	$0{,}82\times10^{-2}$	$\mathbf{0{,}62\times10^{-2}}$
Mediana	$0{,}33\times10^{-2}$	$\mathbf{0{,}21\times10^{-2}}$	$0{,}22\times10^{-2}$

series temporales Passengers, Temperature y Mackey-Glass.

Podemos observar también que aunque AG obtiene mejores resultados que ED y AED para 100 generaciones en tres de los seis casos, si prestamos atención a la media y mediana (últimas filas de ambas tablas), AED es el mejor método seguido de cerca por ED y por último AG.

Si la experimentación se lleva a cabo hasta las doscientas generaciones como muestra la tabla 4.24 y 4.26, se puede observar una mejora generalizada y notable en casi todas las series temporales.

En este caso, ED ha conseguido evolucionar y mejorar los resultados para cinco de las seis series temporales, siendo Temperature la única que no mejora su resultado con respecto a ED lanzado para 100 generaciones. En el caso de AED para 200 generaciones, todas las series temporales obtienen mejoras con respecto a los experimentos de AED lanzados para 100 generaciones.

Si comparamos los resultados obtenidos por ED y AED contra AG para 200 generaciones, ED supera a AG en cuatro (Passengers, Temperature, Dow-Jones y Mackey-Glass) de las seis series temporales. Tan solo en Quebec y Abraham12 el AG sigue siendo mejor o mantiene el resultado dado por ED. Debemos prestar una especial atención a el error obtenido por Mackey-Glass ya que su SMAPE es de *3,74 %* comparado con el *8,04 %* obtenido por el AG, estando los valores predichos por ED muy próximos a los valores reales de la serie temporal.

Esta mejora considerable obtenida con ED después de haber lanzado los experimentos durante doscientas generaciones, podría ser explicada porque en ED una mayor variación en la población (ya que las soluciones aún no convergen) lleva a una búsqueda más variada sobre el espacio de soluciones [194]. Es por ello que aunque ED puede consumir más tiempo para alcanzar una mejor solución, al final la encuentra.

AED también mejora significativamente a AG en cinco (Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass) de las seis series temporales. Tan solo la predicción llevada a cabo por el AG en Passengers es todavía mejor que la de AED, aunque en este caso los errores de sus predicciones están muy próximos.

Debemos prestar atención de nuevo a Mackey-Glass donde esta vez su error SMAPE de predicción es de *1,81 %*, estando esta predicción todavía más cerca de lo que ya la estaba la hecha por ED a los valores reales de la serie temporal. Además el error medio obtenido por AED es *1,5 %* mejor que el obtenido por el AG.

Estos buenos resultados después de lanzar los experimentos durante doscientas generaciones se deben a que AED, a diferencia del AG, evita la convergencia prematura [53]. Por lo tanto, cuando el AG queda bloqueado en un mínimo local del que no puede o le cuesta salir, AED no y sigue buscando una mejor solución o individuo si se le da más tiempo, o en este caso más generaciones para poder buscar.

En general ED, y AED en especial, mejoran los resultados dados por el AG para 100 generaciones. Si se lleva a cabo la experimentación durante más generaciones (i.e. 200), estos dos métodos obtienen mejoras mayores con respecto al AG, haciendo de estos dos métodos evolutivos una muy buena opción para ser usados al llevar a cabo un búsqueda global en un sistema híbrido para diseñar RNA para

predecir series temporales.

A la hora de medir la precisión de cada uno de los métodos presentados, se ha optado por obtener y mostrar la desviación estándar de cada uno de ellos. Como podemos observar en la tabla 4.27, el método más preciso de los tres presentados es además el que obtiene una menor media de error en los resultados, AED.

Tabla 4.27: Desviación estandar para AG, ED y AED.

	AG	ED	AED
Desv. Estand.	3,24	3,94	**3,17**
Media	6,46	5,86	**4,94**

Para poder llevar a cabo las pruebas estadísticas necesarias y poder medir así la diferencia estadística significativa entre diferentes métodos, es preciso tener en cuenta si los datos siguen una distribución normal o no. En caso de tener una distribución normal se puede hacer uso de alguna de las diferentes pruebas paramétricas que existen como el *"t de Student"*, prueba t-Student, o Test-T [210]. Si no fuera ese el caso, es necesario elegir entre una de las diferentes pruebas no paramétricas (Chi cuadrado, prueba exacta de Fischer, test de Wilcoxon, etc.). Para poder contrastar la normalidad de un conjunto de datos dado se ha optado por hacer uso del test de ShapiroWilk [211], considerado uno de los test más potentes para el contraste de normalidad, sobre todo cuando se trabaja con un número de muestras pequeño (menos de 30). Al mostrar este test que los datos dados por estos modelos no son normales, haremos uso de la prueba no paramétrica conocida como test de Wilcoxon [212] ya que permite trabajar con una muestra pequeña de datos.

El test de Wilcoxon parte de la hipótesis H0 de que los valores medios obtenidos son iguales. Al ejecutarse, éste da como resultado un valor a la variable *"P-value"* entre 0 y 1. Si el valor *"P-value"* es pequeño (menor de 0,05) negamos entonces la hipótesis H0, llegando a la conclusión de que es poco probable que la diferencia entre las medias de la muestras sea una coincidencia, con lo que las poblaciones tienen diferentes medias y por lo tanto son estadísticamente significativas.

El test de Wilcoxon ha sido lanzado por parejas sobre los diferentes métodos. Podemos observar en la tabla 4.28 el valor *"P-value"* obtenido para cada par de métodos. Según lo observado en dicha tabla, no existe evidencia clara de que haya una diferencia estadística significativa entre AG y ED o entre ED y AED. Sin embargo, según los criterios convencionales, podemos afirmar que entre AG y AED la diferencia es considerado como no absolutamente estadísticamente significati-

va, estando muy próxima a serlo, siendo *"P-value"* = 0,062.

Tabla 4.28: P-value para AG, ED y AED.

	AG	ED	AED
AG	X	0,418	0,062
ED	0,418	X	0,156
AED	0,062	0,156	X

Resultados Gráficos

Para poder tener una mejor idea sobre las predicciones de cada una de las series temporales presentadas con cada método y como de cerca están éstas a los valores reales, a continuación mostramos una figura por cada serie temporal donde aparecen sus valores reales y las predicciones llevadas a cabo con cada técnica evolutiva. La figura 4.15 muestra una ampliación de la serie passengers, 4.16 hace lo propio para Temperature, 4.17 representa los datos de Dow-Jones, 4.18 para Quebec, Abraham12 en 4.19 y por último Mackey-Glass en 4.20.

En estas gráficas el eje *X* representa el instante en que se tomó la medida y el eje *Y* el valor de ésta en dicho instante. Los valores conocidos de la serie temporal son los representados mediante *TS(tr+val)* y los valores futuros que había que predecir son los representados por *TS(test)*.

Podemos observar que en general todas las predicciones, en mayor o menor grado, siguen la forma dada por los datos reales de las series temporales. Tan solo en la serie temporal Dow-Jones, y debido a la naturaleza caótica de la misma como se muestra en la figura 4.3, las predicciones hechas no respetan la forma aunque sí que consiguen seguir la tendencia de esta serie.

4.3.1. Conclusiones

El objetivo de esta sección consiste en mejorar el proceso de búsqueda global que se lleva a cabo mediante CE. Como ya se ha comentado anteriormente, la primera aproximación del sistema que se diseñó (ADANN) hacía uso de AG para llevar a cabo dicha tarea. En esta sección se ha estudiado dos posibles alternativas al AG que son ED y AED. Cada una de ellas fueron explicadas en detalle para más tarde llevar a cabo la experimentación necesaria y comprobar así si estas dos nuevas opciones de CE mejoran al AG en resultados.

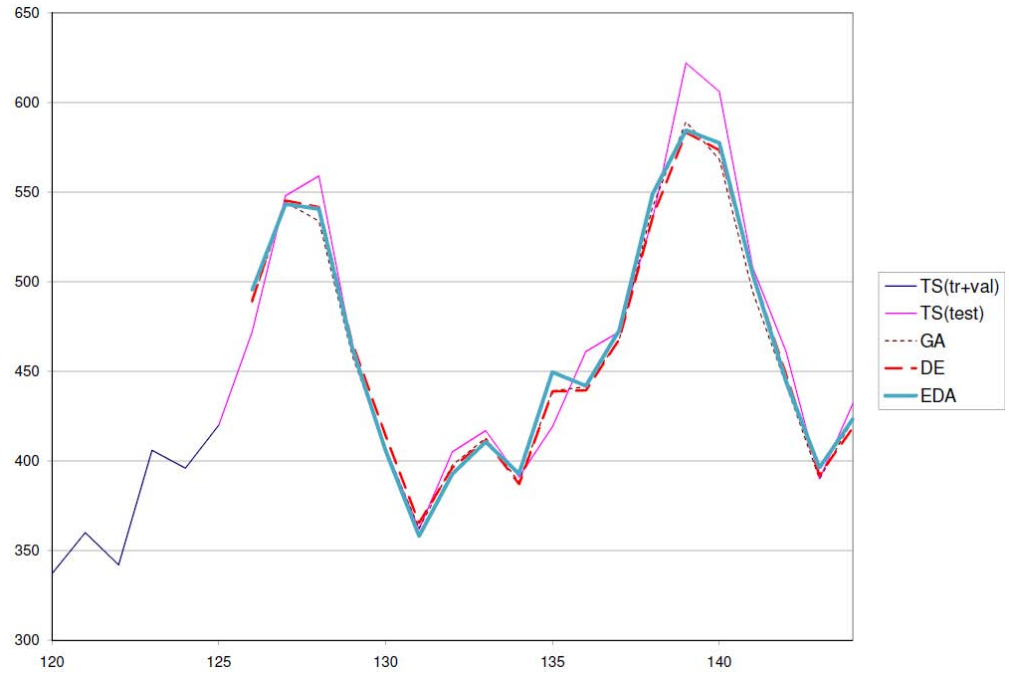


Figura 4.15: Zoom de la predicción de Passengers con AG, ED y AED.

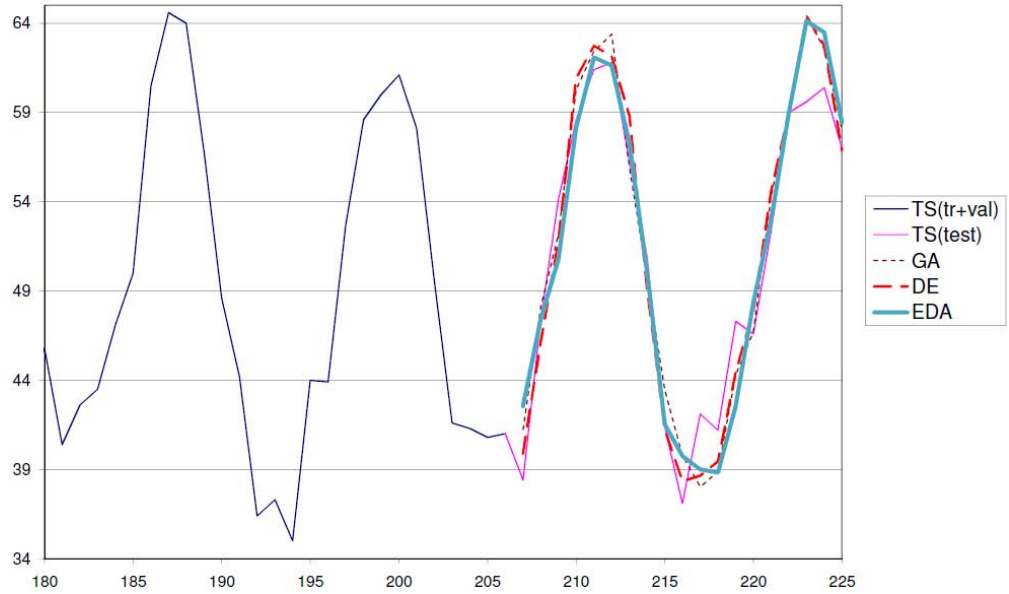


Figura 4.16: Zoom de la predicción de Temperature con AG, ED y AED.

La experimentación revela que usar ED y AED en lugar del AG obtiene buenos resultados y mejoran al AG. Con tan solo cien generaciones, ED y AED mejoran

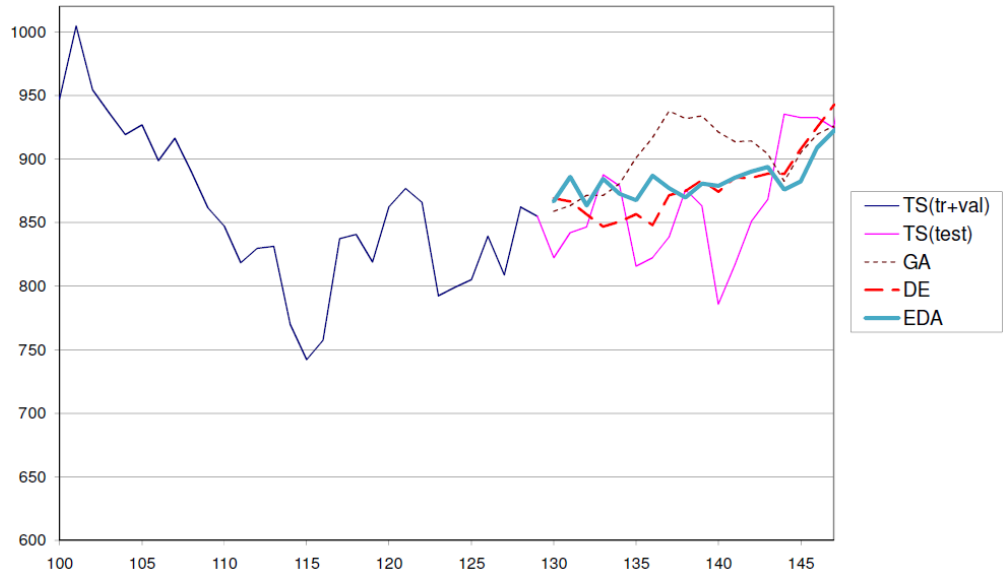


Figura 4.17: Zoom de la predicción de Dow-Jones con AG, ED y AED.

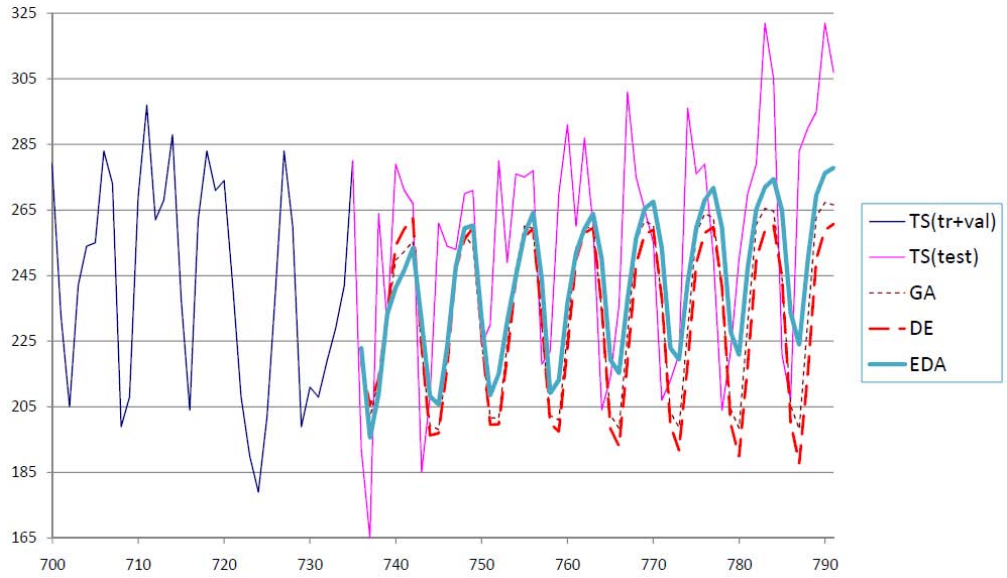


Figura 4.18: Zoom de la predicción de Quebec con AG, ED y AED.

sustancialmente los resultados obtenidos por AG. Pero si se lleva a cabo la experimentación durante 200 generaciones las mejoras observadas son mayores, a veces incluso con una ganancia del *4,3 %* como ocurre en la serie temporal Mackey-Glass usando ED y del *6,2 %* usando AED para esta misma serie.

En general, los resultados obtenidos con ED y AED hacen de estos dos méto-

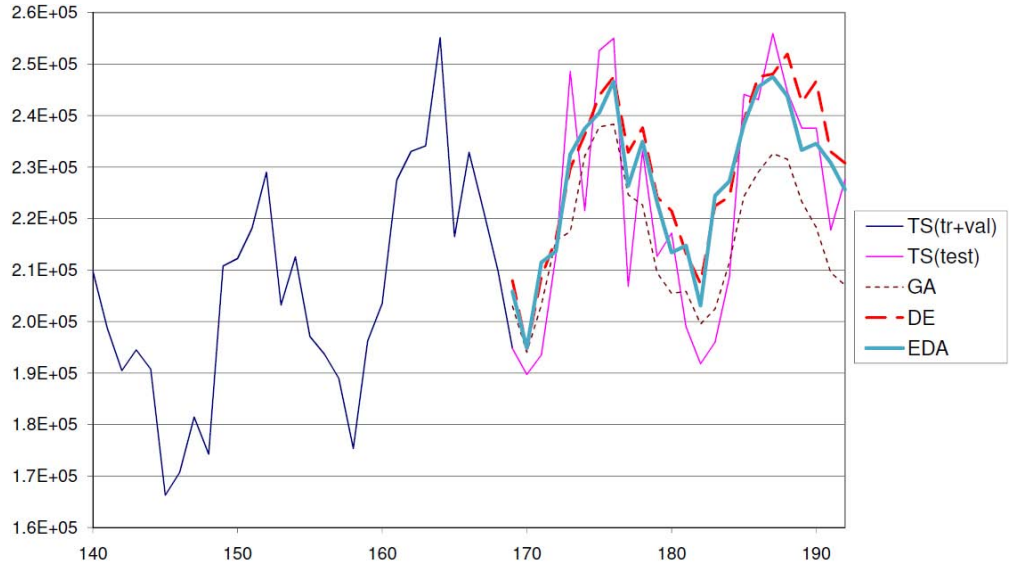


Figura 4.19: Zoom de la predicción de Abraham12 con AG, ED y AED.

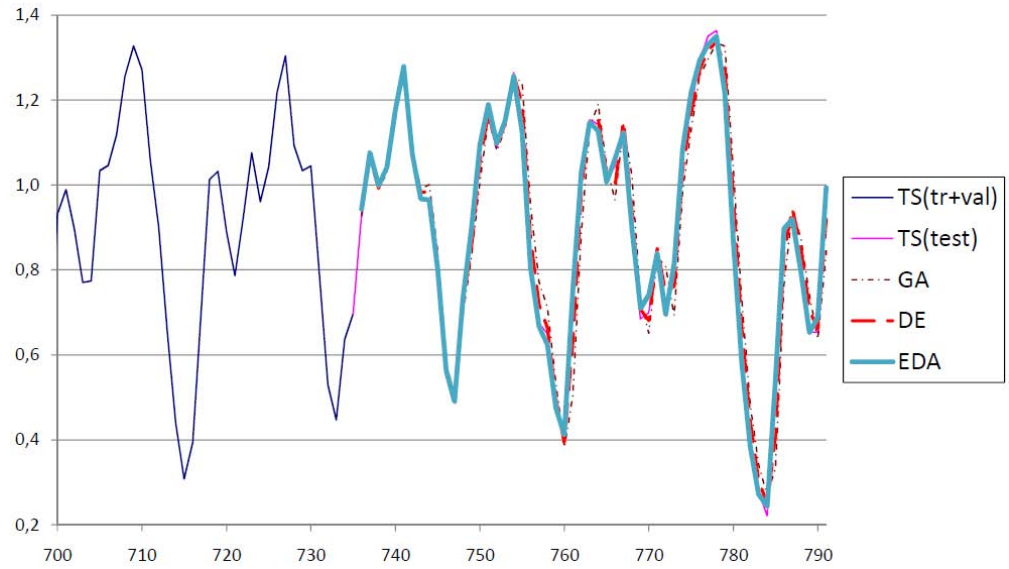


Figura 4.20: Zoom de la predicción de Mackey-Glass con AG, ED y AED.

dos unas buenas opciones a la hora de ser usados para llevar a cabo una búsqueda global en un sistema híbrido como el nuestro para el diseño de RNA.

La primera aproximación del sistema ADANN usando AG fue presentada como método automático para diseñar RNA en la competición internacional NN5, obteniendo el sexto puesto (de unos 80 participantes) con un error SMAPE de *21.9 %* para el conjunto de datos reducido (i.e. predicción de 11 series tempo-

rales). El mejor resultado y por lo tanto ganador de la competición obtuvo un error de SMAPE de *19.0 %*. La herramienta comercial de predicción Autobox [213] basada en la metodología de predicción Box-Jenkins obtuvo un error de *23.9 %*.

El sistema ADANN haciendo uso de AED fue presentado en el *Congreso Español De Informática* (CEDI) 2010 [214]. En dicho congreso se desarrolló una competición nacional de predicción de series temporales llevada a cabo en el *Simposio de Inteligencia Computacional* (SICO). ADANN obtuvo el tercer puesto en dicha competición.

4.4. Redes Totalmente Conectadas vs Ventana Deslizante vs Parcialmente Conectadas

En esta sección se muestran los resultados obtenidos por las diferentes aproximaciones presentadas en la sección 3.7. Recordemos que dichas aproximaciones son modificaciones llevadas a cabo a partir de nuestra primera versión del sistema ADANN. En el caso de AEDRNA, consistía en modificar el algoritmo de aprendizaje de retropropagación por RPROP y el AG encargado de la búsqueda global por AED. A esta nueva aproximación se le sumaron dos más partiendo de AEDRNA, AEDRNAVD en donde se hacía uso de una ventana deslizante para elegir con detalle los instantes anteriores a usar para predecir los futuros y AEDRNAPC donde la arquitectura de RNA usada es un perceptrón multicapa parcialmente conectado en lugar del totalmente conectado como ocurría en AEDRNA.

En una primera experimentación, cada aproximación ADANN y AEDRNA han sido ejecutadas en cinco ocasiones, con un criterio de parada de 100 generaciones para cada serie temporal. Aquí se mostrarán los resultados medios de estas cinco ejecuciones.

En esta sección llevamos a cabo la predicción para cada una de las seis series temporales presentadas en la sección 4.1. Para medir los errores obtenidos en la predicción se usan MSE y SMAPE, presentados en las ecuaciones 4.1 y 4.2 respectivamente. A continuación mostramos los errores asociados a cada una comparando con el conjunto de valores de test o valores futuros reales. Para ello hemos utilizado las tablas 4.29 y 4.30 donde las columnas representan las series temporales y sus respectivos errores MSE y SMAPE, asociados a los dos métodos medidos además de mostrar la media y mediana de dichos errores.

Como ya comentamos en la sección 3.7, el algoritmo de aprendizaje utilizado

Tabla 4.29: Comparación de errores SMAPE entre ADANN y AEDRNA.

Series Temporales	ADANN	AEDRNA
Passengers	**3,18**	3,39
Temperature	4,31	**3,51**
Dow-Jones	6,66	**6,28**
Quebec	12,64	**10,83**
Abraham12	5,53	**4,71**
Mackey-glass	8,67	**7,06**
Media	6,83	**5,96**
Mediana	6,09	**5,49**

Tabla 4.30: Comparación de errores MSE entre ADANN y AEDRNA.

Series Temporales	ADANN	AEDRNA
Passengers	$\mathbf{0{,}61 \times 10^{-3}}$	$0{,}65 \times 10^{-3}$
Temperature	$0{,}36 \times 10^{-2}$	$\mathbf{0{,}25 \times 10^{-2}}$
Dow-Jones	$0{,}22 \times 10^{-1}$	$\mathbf{0{,}18 \times 10^{-1}}$
Quebec	$0{,}25 \times 10^{-1}$	$\mathbf{0{,}20 \times 10^{-1}}$
Abraham12	$0{,}18 \times 10^{-2}$	$\mathbf{0{,}14 \times 10^{-2}}$
Mackey-glass	$0{,}36 \times 10^{-2}$	$\mathbf{0{,}29 \times 10^{-2}}$
Media	$0{,}94 \times 10^{-2}$	$\mathbf{0{,}75 \times 10^{-2}}$
Mediana	$0{,}36 \times 10^{-2}$	$\mathbf{0{,}27 \times 10^{-2}}$

en el sistema AEDRNA es RPROP. Al ser su convergencia de entrenamiento más rápida [88], serán necesarios menos ciclos de aprendizaje para obtener resultados parecidos a los obtenidos por el algoritmo de retropropagación y por lo tanto menos tiempo computacional.

Para demostrar esto empíricamente, en la tabla 4.31 donde se compara el tiempo computacional consumido por ADANN y AEDRNA. El computador usado para esta experimentación es un servidor HP Proliant ML350 G6.

Para cada serie temporal y método evolutivo, en esta tabla se muestra el tiempo computacional llevado a cabo dado en términos del tiempo total transcurrido en minutos. La última columna expresa en porcentaje, la razón de reducción de

tiempo computacional logrado cuando se compara ADANN y AEDRNA, donde $R_t = 1 - (tiempo_{\text{AEDRNA}}/tiempo_{\text{ADANN}})$.

Tabla 4.31: Comparación de tiempo computacional entre ADANN y AEDRNA.

Series Temporales	ADANN tiempo (min)	AEDRNA tiempo (min)	R_t
Passengers	2929	165	94 %
Temperature	4260	315	92 %
Dow-Jones	1881	161	91 %
Quebec	8591	6603	23 %
Abraham12	3055	420	86 %
Mackey-glass	5554	5649	-1,71 %

En general, AEDRNA obtiene mejores resultados en cinco de las seis series temporales, con considerables mejoras en las series Quebec y Mackey-Glass si nos fijamos en el error SMAPE. Solo en Passengers, ADANN sigue siendo mejor que AEDRNA. Al observar el valor medio y mediana obtenido de todos los experimentos, AEDRNA es la mejor opción.

En cuanto a la reducción de tiempo se refiere, AEDRNA es más rápido que ADANN en cinco de los seis casos. Aún así, se puede observar una mejora diferente dependiendo de que serie temporal observemos. Para las cuatro series temporales cortas o de pocos elementos, la reducción de coste computacional es mucho mayor. Sin embargo, en el caso de Quebec esta reducción, aunque existe, es menor que la dada para las series anteriores. Mackey-Glass sin embargo consume un *1,71 %* más de tiempo.

De nuevo, para poder medir la precisión de los métodos presentados se ha optado por obtener y mostrar la desviación estándar de cada uno de ellos. Como podemos observar en la tabla 4.32, el método más preciso y el que obtiene una menor media en los resultados es AEDRNA.

Tabla 4.32: Desviación estandar para ADANN y AEDRNA.

	ADANN	AEDRNA
Desv. Estand.	3,42	**2,80**
Media	6,83	**5,96**

Se ha vuelto a hacer uso del test de Wilcoxon sobre estos dos métodos. El va-

lor *"P-value"* obtenido para este par de métodos es de 0,061. Podemos entonces concluir que según los criterios convencionales, la diferencia entre la media de ADANN y AEDRNA se puede considerar como no absolutamente estadísticamente significativa, estando muy próxima a serlo.

4.4.1. Redes totalmente Conectadas vs Ventana Deslizante

Para llevar a cabo la experimentación se ha usado el mismo conjunto de series temporales, Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mavkey-Glass. Los valores predichos son de nuevo comparados con los valores reales de las respectivas series temporales y para medir los errores obtenidos en la predicción se ha hecho uso de MSE y SMAPE presentados en las ecuaciones 4.1 y 4.2 respectivamente.

Cada aproximación evolutiva (AEDRNA y AEDRNAVD) presentada en esta sección ha sido ejecutada en cinco ocasiones, con un criterio de parada de 100 generaciones para cada serie temporal. Se presenta pues la media de estas cinco ejecuciones. Los errores obtenidos son mostrados en la tabla 4.33 para SMAPE y la tabla 4.34 para MSE. Las dos últimas filas de cada tabla muestra las respectivas medias y medianas de los errores conseguidos con cada método.

Tabla 4.33: Comparación de errores SMAPE entre AEDRNA y AEDRNAVD.

Series Temporales	AEDRNA	AEDRNAVD
Passengers	**3,39**	3,78
Temperature	**3,51**	3,74
Dow-Jones	6,28	**5,54**
Quebec	10,83	**9,30**
Abraham 12	4,71	**4,34**
Mackey-glass	7,06	**4,93**
Media	5.96	**5,27**
Mediana	5,49	**4,63**

Como se puede observar en las tablas 4.33 y 4.34, la estrategia de ventana deslizante propuesta (AEDRNAVD) obtiene mejores resultados que la versión sin ventana (AEDRNA) en cuatro de los seis conjuntos de datos y en ambos indicadores de error. En cuanto a las series temporales Passengers y Temperature, AEDRNA sigue siendo mejor método aunque AEDRNAVD obtiene errores simi-

Tabla 4.34: Comparación de errores MSE entre AEDRNA y AEDRNAVD.

Series Temporales	AEDRNA	AEDRNAVD
Passengers	$\mathbf{0{,}65\times10^{-3}}$	$0{,}85\times10^{-3}$
Temperature	$\mathbf{0{,}25\times10^{-2}}$	$0{,}31\times10^{-2}$
Dow-Jones	$0{,}18\times10^{-1}$	$\mathbf{0{,}15\times10^{-1}}$
Quebec	$0{,}20\times10^{-1}$	$\mathbf{0{,}14\times10^{-1}}$
Abraham 12	$0{,}14\times10^{-2}$	$\mathbf{0{,}13\times10^{-2}}$
Mackey-glass	$0{,}29\times10^{-2}$	$\mathbf{0{,}11\times10^{-2}}$
Media	$0{,}75\times10^{-2}$	$\mathbf{0{,}58\times10^{-2}}$
Mediana	$0{,}27\times10^{-2}$	$\mathbf{0{,}22\times10^{-2}}$

lares. Según el error SMAPE, existe tan sólo una diferencia de 0,2 y 0,06 puntos porcentuales para Passengers y Temperature respectivamente.

Por otra parte, el resultado de error medio y mediano (representado por las últimas fila de las dos tablas), sitúa a AEDRNAVD como el mejor método para predecir series temporales en ambas métricas de error.

En la tabla 4.35 presentamos como ejemplo el mejor individuo logrado por AEDRNAVD durante una de las cinco ejecuciones que se ha llevado a cabo para cada serie temporal. La segunda columna muestra el valor tomado por *i* en el cromosoma (i.e. número de instantes anteriores considerados antes de aplicar la ventana deslizante), el cual, si estuviéramos hablando de la versión sin ventana deslizante (AEDRNA), representaría los valores anteriores de la serie temporal usados además del número de nodos de entrada en la RNA correspondiente. Pero al encontrarnos haciendo uso de la nueva versión del sistema con ventana (AEDRNAVD), se puede observar en la tercera columna los instantes anteriores, de todos los propuestos por *i*, que finalmente se han tenido en cuenta en forma de vector para cada serie temporal. En la cuarta columna se muestra cuantos instantes se han eliminado de los propuestos por *i* y en la quinta el número final y definitivo de neuronas de entrada que tendrá la RNA resultante después de borrar los no deseados. Cabe destacar que el valor dado por *entradas* representa también el número de instantes de forma discontinua que se usarán para generar los patrones.

Como se muestra en la cuarta y quinta columna (instantes borrados y número de nodos de entrada del modelo), la ventana deslizante binaria realiza una poda sustancial del número máximo de nodos de entrada (i), lo que conduce a modelos mucho más simples.

Tabla 4.35: Ejemplo de los mejores modelos de predicción con AEDRNAVD.

Series Temporales	*i*	ventana deslizante	instantes borrados	entradas
Passengers	49	{3-5,9,13,15,16,18,22,. . . ,23,24,26,28,32,33,42,44,45,46,}	31	18
Temperature	67	{3-5,7,9-12,15,17,19,21,22,28,. . . ,34-36,38,41,46-49,53-57,59-61,65,66}	32	35
Dow-Jones	41	{3,4,7,8,11,13,17,22,. . . ,25-28,30,32,34,36,37,41}	23	18
Quebec	43	{2,5,8,10,14,16,17,23,. . . ,26,29,30,34,37,41-43}	26	17
Abraham12	30	{6,8,11-13,17-19,21,23,25,27,28}	17	13
Mackey-Glass	25	{6,8,10,11,13,15-17,19-25}	10	15

La tabla 4.36 compara las características de las mejores RNA evolucionadas mediante AEDRNA y AEDRNAVD, donde cada celda muestra la media de las cinco ejecuciones que se han llevado a cabo para cada serie temporal.

Para cada serie temporal y método evolutivo, se muestra en esta tabla el número de nodos de entrada de las RNA ($ent.$) obtenidas, el número de sus neuronas ocultas ($ocul.$), número total de conexiones ($conex.$) y el tiempo computacional consumido de media dado en términos del tiempo total transcurrido en minutos.

Las dos últimas columnas muestran el porcentaje de reducción alcanzado cuando se compara AEDRNA contra AEDRNAVD, en este caso específico $R_F = 1 - (F_{\text{AEDRNAVD}}/F_{\text{AEDRNA}})$ y F es el factor de análisis (es decir, c número total de conexiones o t tiempo transcurrido en minutos).

En general, AEDRNAVD obtiene estructuras de RNA más simples y por lo tanto más fáciles de estudiar en detalle. La reducción en el número de entradas es considerable y al tratarse de perceptrones multicapa totalmente conectados, esta reducción es extrapolada también al número de conexiones (menor número de entradas $\Rightarrow$ menor número de conexiones) ya que con el nuevo sistema no se da un aumento generalizado e importante en el número de nodos ocultos. En particular, se dan altas tasas de reducción en las series temporales Passengers y Abraham12. La excepción se produce en Mackey-Glass, donde AEDRNAVD optimiza un poco el número de nodos de entrada (de trece a doce) a costa de incrementar considerablemente el número de nodos ocultos, con lo que al final el número de conexiones es mayor cuando los comparamos con AEDRNA.

Tabla 4.36: Comparación de los mejores modelos optimizados con AEDRNA y AEDRNAVD.

Series	AEDRNA				AEDRNAVD				R_c	R_t
	ent.	ocul.	conex.	tiempo	ent.	ocul.	conex.	tiempo		
Passengers	49,2	67,4	3383,4	165	23,0	72,0	1728,0	61	48,9 %	63,0 %
Temperature	63,6	64,8	4186,1	315	37,6	80,6	3111,2	199	25,7 %	36,8 %
Dow-Jones	35,8	48,8	1795,8	161	21,4	64,8	1451,5	67	19,2 %	58,4 %
Quebec	14,6	136,6	2131,0	6603	16,2	115,4	1984,9	3906	6,9 %	40,8 %
Abraham12	30,4	117,8	3698,9	270	16,0	95,4	1621,8	109	56,2 %	59,6 %
Mackey-Glass	13,0	90,4	1265,6	5649	12,0	120,4	1565,2	5613	-23,7 %	0,64 %

En cuanto a la reducción de tiempo se refiere AEDRNAVD es en todos los casos más rápido que AEDRNA requiriendo en todos los ejemplos dados mucho menos coste computacional excepto en Mackey-Glass donde la mejora es menor.

Para medir la precisión de los métodos presentados, se obtiene la desviación estándar de cada uno de ellos. Como podemos observar en la tabla 4.37, el método más preciso y el que obtiene una menor media en los resultados es AEDRNAVD.

Tabla 4.37: Desviación estandar para AEDRNA y AEDRNAVD.

	AEDRNA	AEDRNAVD
Desv. Estand.	2,80	**2,09**
Media	5,96	**5,27**

Al llevar a cabo el test de Wilcoxon sobre estos dos métodos, el valor *"P-value"* obtenido para este par de métodos es de 0,218. Podemos entonces concluir que no existe evidencia clara de que haya una diferencia estadísticamente significativa entre AEDRNA y AEDRNAVD.

4.4.2. Redes totalmente Conectadas vs Parcialmente Conectadas

Cada método (AEDRNA y AEDRNAPC) ha sido ejecutado cinco veces, con un criterio de parada de 100 generaciones para cada serie temporal. Se muestran a continuación la media de estas cinco ejecuciones. Los errores obtenidos son

mostrados en la tabla 4.38 para SMAPE y la tabla 4.39 para MSE. Al final de cada tabla se muestran también las respectivas medias de los errores conseguidos con cada método.

Para llevar cabo la experimentación se ha usado el mismo conjunto de series temporales que se ha usado en la sección 4.3 con una batería de prueba compuesta por las series temporales Passengers, Temperature, Dow-Jones, Quebec, Mavkey-Glass y Abraham12. Los valores predichos son de nuevo comparados con los valores reales de las respectivas series temporales y para medir los errores obtenidos en la predicción se ha hecho uso de SMAPE y MSE presentados en las ecuaciones 4.2 y 4.1 respectivamente.

Tabla 4.38: Comparación de errores SMAPE entre AEDRNA y AEDRNAPC.

Series	AEDRNA	AEDRNAPC
Passengers	3,39	**3,20**
Temperature	**3,51**	4,17
Dow-Jones	6,28	**5,79**
Quebec	10,83	**8,00**
Abraham 12	4,71	**4,48**
Mackey-glass	7,06	**4,03**
Media	5,96	**4,94**
Mediana	5,49	**4,32**

Tabla 4.39: Comparación de errores MSE entre AEDRNA y AEDRNAPC.

Series	AEDRNA	AEDRNAPC
Passengers	$\mathbf{0{,}65\times10^{-3}}$	$0{,}68\times10^{-3}$
Temperature	$\mathbf{0{,}25\times10^{-2}}$	$0{,}32\times10^{-2}$
Dow-Jones	$0{,}18\times10^{-1}$	$\mathbf{0{,}14\times10^{-1}}$
Quebec	$0{,}20\times10^{-1}$	$\mathbf{0{,}11\times10^{-1}}$
Abraham 12	$0{,}14\times10^{-2}$	$\mathbf{0{,}13\times10^{-2}}$
Mackey-glass	$0{,}29\times10^{-2}$	$\mathbf{0{,}83\times10^{-3}}$
Media	$0{,}75\times10^{-2}$	$\mathbf{0{,}51\times10^{-2}}$
Mediana	$0{,}27\times10^{-2}$	$\mathbf{0{,}22\times10^{-2}}$

Un análisis de las tablas, muestra que AEDRNAPC provee en general mejores predicciones cuando se compara con el método AEDRNA. El método donde usamos los perceptrón multicapa parcialmente conectadas supera al otro, totalmente

conectado, en cuatro casos al considerar el criterio MSE, y es mejor en cinco conjuntos de datos al considerar el error SMAPE.

Un estudio más detallado sobre la tabla que muestra los resultados obtenidos con SMAPE, el cual se tiene más en cuenta en el dominio de la predicción, muestra mejoras interesantes de AEDRNAPC sobre AEDRNA. Un ejemplo es la diferencia obtenida por AEDRNAPC de *1,9*, *2,8* y *3,0* puntos porcentuales para las tres últimas series temporales.

Las últimas filas de las tablas, que muestra el rendimiento en promedio y la media a través de todas las series temporales, también sitúa AEDRNAPC como el mejor método, en ambos criterios de error.

En la tabla 4.40 se comparan las características de las mejores RNA de cada serie temporal evolucionadas mediante ambas aproximaciones AEDRNA y AEDRNAPC, donde cada celda muestra el valor medio de las cinco ejecuciones llevadas a cabo para cada serie temporal. Para cada serie y método evolutivo, se muestra el número de entradas de las RNA ($ent.$), nodos ocultos ($ocul.$), el número total de conexiones ($conex.$) y el esfuerzo computacional medio, dado en términos de tiempo total transcurrido para cada método en minutos.

Las últimas dos columnas muestran la tasa de reducción alcanzada cuando comparamos AEDRNA contra AEDRNAPC. Como podemos observar, $R_F = 1 - (F_{\text{AEDRNAPC}}/F_{\text{AEDRNA}})$ y F es el factor a analizar (es decir, c número total de conexiones o t tiempo transcurrido).

Tabla 4.40: Comparación de los mejores modelos optimizados con AEDRNA y AEDRNAPC.

Series	AEDRNA				AEDRNAPC				R_c	R_t
	ent.	ocul.	conex.	tiempo	ent.	ocul.	conex.	tiempo		
Passengers	49,2	67,4	3383,4	165	49,6	71,4	1813,6	71	46,4 %	56,9 %
Temperature	63,6	64,8	4186,1	315	65,0	59,8	2011,1	114	51,9 %	63,8 %
Dow-Jones	35,8	48,8	1795,8	161	38,6	93,6	1868,8	73	-4,1 %	54,7 %
Quebec	14,6	136,6	2131,0	6603	12,2	111,2	824,8	5221	61,3 %	20,9 %
Abraham12	30,4	117,8	3698,9	270	21,2	99,4	1118,4	89	69,8 %	67,0 %
Mackey-Glass	13,0	90,4	1265,6	5649	14,6	117,8	924,6	5590	26,9 %	1,1 %

La tabla 4.40 muestra que, en general, AEDRNAPC obtiene modelos o arquitecturas de RNA más simples y por lo tanto más fáciles de estudiar. En particular, cabe destacar la gran tasa de reducción alcanzada para las series temporales Abra-

ham12 y Quebec. La excepción se produce en este caso en la serie temporal Dow-Jones, donde AEDRNAPC consigue como resultado una RNA con muchos más nodos ocultos cuando lo comparamos con AEDRNA, aunque al final el número total de conexiones es tan sólo *4 %* mayor que AEDRNA.

Además, AEDRNAPC es en todos los casos más rápida (computacionalmente hablando), requiriendo mucho menos tiempo computacional en todas las series temporales para obtener mejores resultados a excepción de Mackey-Glass donde la mejora mínima.

Para medir la precisión de los métodos presentados se emplea la desviación estándar de cada uno de ellos. Como podemos observar en la tabla 4.41, el método más preciso y el que obtiene una menor media en los resultados es AEDRNAPC.

Tabla 4.41: Desviación estandar para AEDRNA y AEDRNAPC.

	AEDRNA	AEDRNAPC
Desv. Estand.	2,80	**1,71**
Media	5,96	**4,94**

Al llevar a cabo el test de Wilcoxon sobre estos dos métodos. El valor *"P-value"* obtenido para este par de métodos es de 0,220. Podemos entonces concluir que no existe evidencia clara de que haya una diferencia estadística significativa entre AEDRNA y AEDRNAPC.

4.4.3. Conclusiones

En esta sección presentamos los resultados obtenidos por el sistema ADANN, comparado con el sistema AEDRNA.

Después de llevar a cabo la experimentación pertinente se han podido observar mejoras significativas, no solo en cuanto a los errores de predicción obtenidos, siendo las predicciones del nuevo sistema más precisas, sino en cuanto al coste computacional se refiere. En algunos casos AEDRNA ha obtenido una mejora superior al *90 %* en tiempo de ejecución.

Por otro lado, se ha adoptado la popular ventana deslizante usada en predicción de series temporales mediante RNA [200] para modificar el sistema AEDRNA. En lugar de llevar a cabo un diseño manual de dicha ventana, este diseño de los elementos anteriores de la serie temporal elegidos para predecir el actual se incluye en el método automático que forma nuestro nuevo sistema AEDRNAVD. Este no

requiere de conocimiento previo por parte del usuario sobre la serie temporal a predecir y por lo tanto es más sencillo de usar.

Se ha llevado a cabo la experimentación necesaria para poder comparar los dos métodos, el comentado con anterioridad (AEDRNA) sin ventana y la versión que hace uso de dicha ventana (AEDRNAVD). La batería de pruebas usada consta de seis series temporales distintas y las predicciones obtenidas han sido analizadas con dos criterios de error, MSE y SMAPE.

Los resultados obtenidos han sido muy satisfactorios. Al medir los errores cometidos con ambos sistemas, AEDRNAVD es sin duda mejor que AEDRNA, además se ha conseguido una reducción importante en el coste computacional en los experimentos llevados a cabo.

Resultados parecidos a los anteriores han sido logrados por el sistema propuesto en esta tesis. Nos referimos a AEDRNAPC, donde se ha optado por hacer uso del perceptrón multicapa pero parcialmente conectado, esto es, pudiendo elegir qué conexiones se llevan a cabo entre los nodos de la RNA. Gracias a esta nueva modificación y a su correspondiente codificación en el cromosoma, el sistema es capaz de eliminar aquellas conexiones que no ayudan a obtener un buen resultado, ampliando también considerablemente el espacio de búsqueda. Los resultados obtenidos con este último sistema (AEDRNAPC) comparados con AEDRNA muestran que es una buena opción a tener en cuenta no solo por obtener mejores predicciones sino que además disminuye también considerablemente el tiempo de ejecución de la experimentación.

En general, tanto AEDRNAVD como AEDRNAPC obtienen modelos para predecir series temporales más sencillos que los obtenidos por la aproximación AEDRNA.

4.5. Estudio Comparativo de Predicciones

En esta sección se muestran los resultados obtenidos con el sistema AEDRNAPC comparado con diferentes métodos o sistemas existentes y provenientes de distintos campos, métodos estadísticos y herramientas software comerciales, entre otros.

Así, en primer lugar se realiza una comparación de las predicciones llevadas a cabo sobre un conjunto de series temporales mediante este sistema propuesto en esta tesis doctoral y dos de los métodos estadísticos más conocidos para predecir series temporales, Modelos de Componentes no Observables (tipo Harvey) [201] y ARIMA [20] de Box-Jenkins.

En un segundo bloque de experimentación se comparan los resultados obtenidos por AEDRNAPC con los obtenidos por una herramienta software comercial de predicción de series temporales denominada *ForecastPro* [202] (FP). Además, se mostrarán en detalle dichas predicciones mediante sus respectivas gráficas.

En la última sección se compararán los errores dados por las predicciones y se mostrarán las gráficas generadas por AEDRNAPC y otros tres sistemas de IC como son, Máquinas de Soporte Vectorial (MSV) y dos sistemas híbridos formados por, lógica difusa y RNA (FRNA) y por otro lado lógica difusa y MSV (FMSV).

4.5.1. Métodos Estadísticos

ARIMA, del inglés *"Autoregressive Integrated Moving Average"*, fue descrito por los estadísticos George Box y Jenkins Gwilym en 1970 [20] como una manera sistemática y ordenada de dar forma a las series temporales. ARIMA es además un proceso matemático usado para predecir series temporales.

El modelo de Box-Jenkins consiste en identificar un proceso ARIMA apropiado, adaptándolo a los datos, y utilizando el modelo ajustado para el pronóstico. Una de las características más atractivas de la aproximación de Box-Jenkins para la predicción es que los procesos ARIMA son una clase muy rica de posibles modelos y por lo general es posible encontrar un proceso que proporciona una descripción adecuada de los datos.

Un modelo ARIMA consta de una combinación de tres componentes: autorregresivo (es decir, un valor de la serie temporal como una regresión lineal de los valores anteriores (parámetro a); tendencia, constante lineal cuadrática (parámetro t); media móvil, es decir, un valor de la serie temporal que es la suma de la regresión lineal de la media y el ruido blanco (parámetro q). Por lo tanto, el modelo ARIMA (a,t,q) para una serie temporal univariante es una combinación de modelos lineales de sus valores anteriores y los errores de estos.

Por otro lado, los modelos estructurales o modelos de componentes no observables (del inglés *"Unobserved Components Models"*, UCM) [201] se basan en la captura de las características explícitas de la serie temporal que está siendo estudiada, y que corresponden a los movimientos a largo plazo y sus patrones cíclicos y repetitivos, los cuales están representados por la tendencia y estacionalidad del patrón, respectivamente.

En estos modelos, tales componentes pueden variar estocásticamente en el tiempo para que puedan adaptarse a los cambios que sufre la serie temporal. Estas componentes son:

- Tendencia.
- Ciclos a largo plazo, como pueden ser las crisis económicas.
- Componente estacional: corresponde a un patrón de repetición dado a corto plazo, con periodicidad anual, mensual, semanal, etc.
- Componente irregular: corresponde a una variable aleatoria que no muestran ninguna regularidad, debidos a fenómenos de carácter ocasional como pueden ser tormentas, terremotos, inundaciones, huelgas, guerras, avances tecnológicos, etc.

Combinando estos componentes, se obtienen modelos diferentes que pueden ser usados para predecir series temporales.

Para llevar a cabo la experimentación de comparación entre los métodos estadísticos propuestos y AEDRNAPC, se han seleccionado las mismas series temporales usadas en apartados anteriores (Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass) además de volver a hacer uso de los errores SMAPE y MSE para medir la precisión de las diferentes técnicas. AEDRNAPC ha sido lanzado con un criterio de parada de cien generaciones como se ha hecho en experimentaciones previas.

A continuación mostramos las tablas 4.42 y 4.43 donde se pueden observar los errores SMAPE y MSE obtenidos por AEDRNAPC, ARIMA y UCM.

Estas dos últimas predicciones (i.e. ARIMA y UCM) han sido llevadas a cabo por el catedrático de universidad de Castilla la Mancha Diego J. Pedregal haciendo uso de un *"toolbox"* para Matlab implementado por él mismo denominado *"CAPTAIN"* [215].

El primer dato importante que se puede observar en las tablas es que no ha sido posible llevar a cabo la predicción mediante ninguno de los dos métodos estadísticos de la serie temporal Mackey-Glass. Esto es debido a que dichos métodos necesitan disponer de información previa sobre las series temporales, como puede ser la periodicidad, a contrario de lo que ocurre con AEDRNAPC, donde ningún conocimiento previo por parte del usuario es necesario.

Así pues, mientras las demás series han podido ser predichas puesto que se conocen datos sobre ellas como son la frecuencia de medida, no ocurre igual con Mackey-Glass, ya que ésta es una serie temporal caótica generada a partir de ecuaciones diferenciales ordinarias.

AEDRNAPC ha conseguido unas predicciones para Passengers y Mackey-Glass muy próximas a los datos reales con un error SMAPE de *3,20 %* y *4,03 %* respectivamente.

Tabla 4.42: SMAPE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AEDRNAPC, ARIMA y UCM.

Series Temporales	AEDRNAPC	ARIMA	UCM
Passengers	3,20	3,14	**2,37**
Temperature	4,17	3,98	**3,38**
Dow-Jones	5,79	**4,78**	**4,78**
Quebec	**8,00**	11,26	10,76
Abraham12	**4,48**	6,21	8,11
Mackey-Glass	**4,03**	- -	- -

Tabla 4.43: MSE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AEDRNAPC, ARIMA y UCM.

Series Temporales	AEDRNAPC	ARIMA	UCM
Passengers	$0{,}68\times10^{-3}$	$0{,}41\times10^{-3}$	$\mathbf{0{,}28\times10^{-3}}$
Temperature	$0{,}32\times10^{-2}$	$0{,}29\times10^{-2}$	$\mathbf{0{,}24\times10^{-2}}$
Dow-Jones	$0{,}14\times10^{-1}$	$\mathbf{0{,}11\times10^{-1}}$	$\mathbf{0{,}11\times10^{-1}}$
Quebec	$\mathbf{0{,}11\times10^{-1}}$	$0{,}19\times10^{-1}$	$0{,}19\times10^{-1}$
Abraham12	$\mathbf{0{,}13\times10^{-2}}$	$0{,}22\times10^{-2}$	$0{,}38\times10^{-2}$
Mackey-Glass	$\mathbf{0{,}83\times10^{-3}}$	- -	- -

Nuestra aproximación además ha sido capaz de mejorar los resultados obtenidos por ambos métodos estadísticos en tres de las seis series temporales, algunos con mejoras de dos puntos porcentuales como ocurre con Quebec. En cuanto a las otras tres series temporales, donde no ha mejorado a los métodos estadísticos, cabe resaltar que sus resultados están muy próximos a los de estos métodos.

Podemos observar también que tanto ARIMA como UCM obtienen el mismo error SMAPE y MSE en la serie temporal Dow-Jones, *4,78 %*, mejorando la predicción hecha por AEDRNAPC cuyo error SMAPE es de *5,79 %*.

Se puede concluir que AEDRNAPC es una buena opción a la hora de predecir series temporales cuando lo comparamos con ARIMA y UCM.

Resultados Gráficos

Para poder observar en más detalle las predicciones llevadas a cabo, a continuación se muestra una gráfica para cada serie temporal donde se representan los valores conocidos de estas (*tr+val*) y los futuros que hay que predecir (*test*), además de los valores predichos por cada uno de los sistemas comentados con anterioridad, AEDRNAPC, ARIMA y UCM. La figura 4.21 muestra esta información para Passengers, en 4.22 podemos observar Temperature, por otro lado 4.23 para Dow-Jones, 4.24 para Quebec, Abraham12 en 4.25 y por último 4.26 muestra Mackey-Glass.

En estas gráficas, el eje *X* representa el instante en que se tomó la medida y el eje *Y* el valor de ésta en dicho instante. Los valores conocidos de la serie temporal son los representados mediante *TS(tr+val)* y los valores futuros que había que predecir son los representados por *TS(test)*.

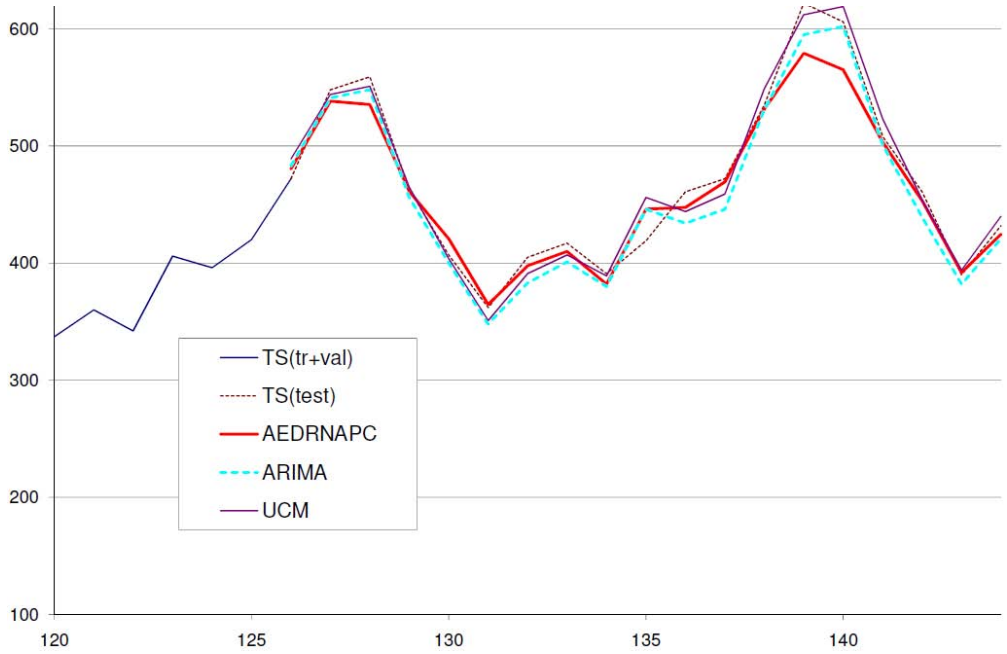


Figura 4.21: Zoom de la predicción de Passengers con AEDRNAPC, UCM y ARIMA.

Todos los métodos (i.e. AEDRNAPC, ARIMA y UCM) consiguen seguir la forma que tienen las series temporales en sus predicciones salvo en Dow-Jones.

Si nos fijamos con detenimiento en la serie temporal Dow-Jones (figura 4.23), podemos observar que tan solo AEDRNAPC ha sido capaz de seguir la forma de los valores a predecir de la serie temporal, mientras que los otros dos métodos llevan a cabo una predicción de una línea recta. Esto es debido a que esta serie

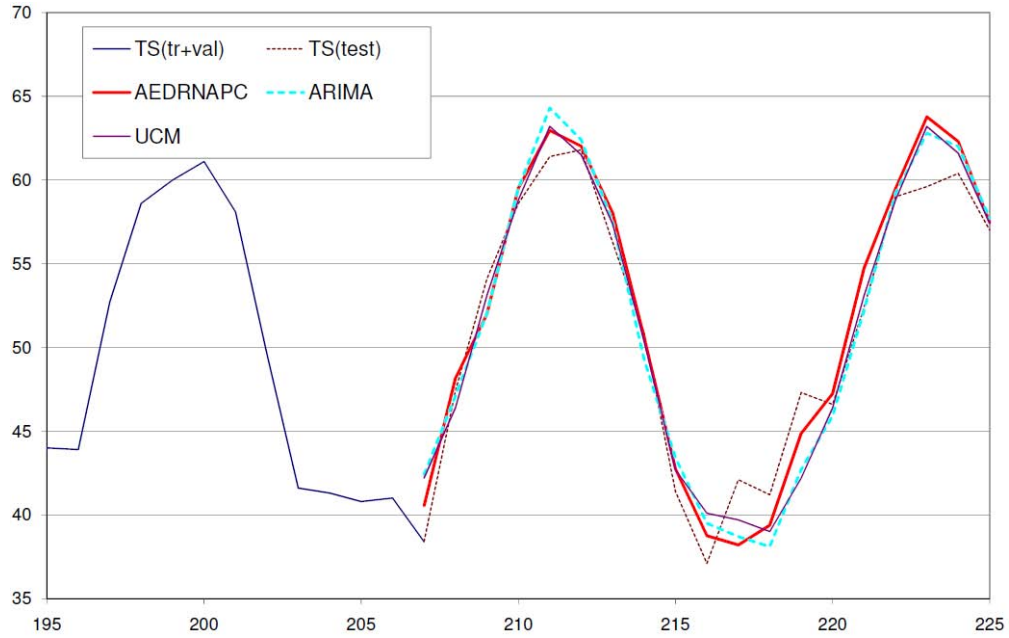


Figura 4.22: Zoom de la predicción de Temperature con AEDRNAPC, UCM y ARIMA.

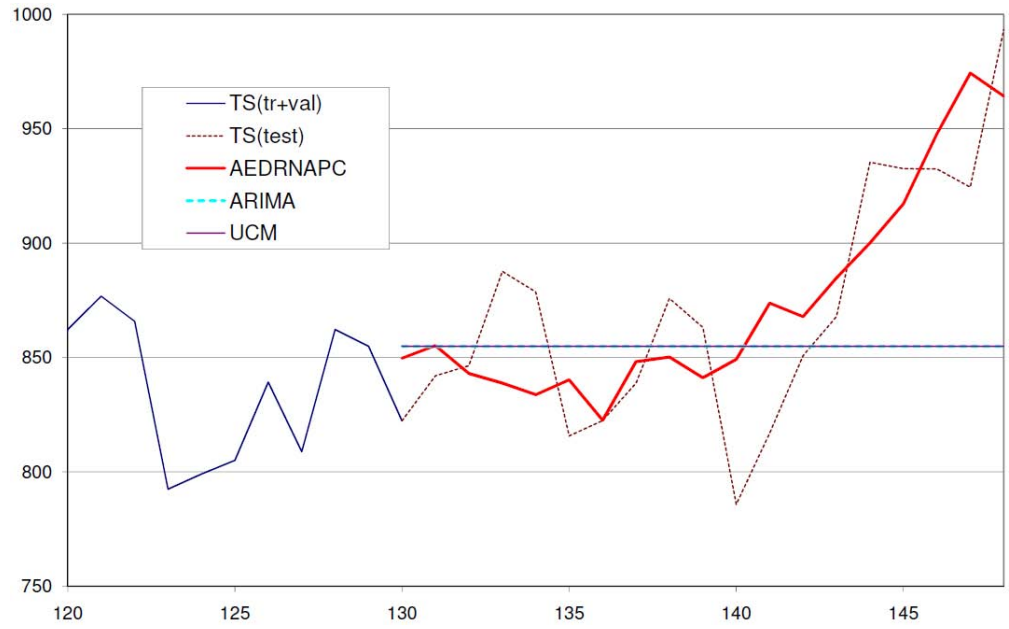


Figura 4.23: Zoom de la predicción de Dow-Jones con AEDRNAPC, UCM y ARIMA.

temporal, a diferencia de las otras, es más irregular ya que no dispone de un ciclo estacional claro. Así, ARIMA y UCM, al no detectar dicho ciclo, llevan a cabo

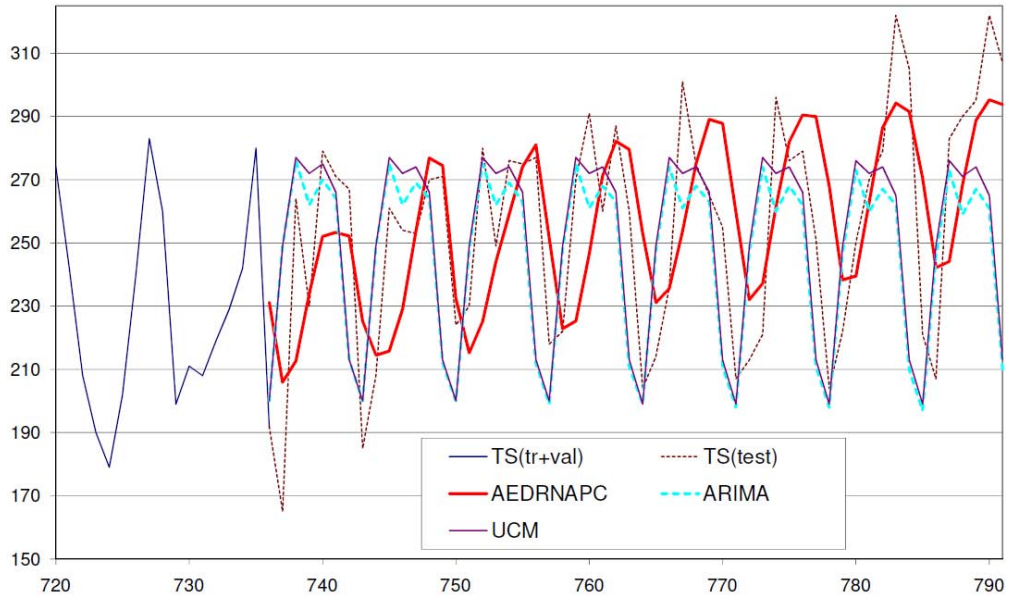


Figura 4.24: Zoom de la predicción de Quebec con AEDRNAPC, UCM y ARIMA.

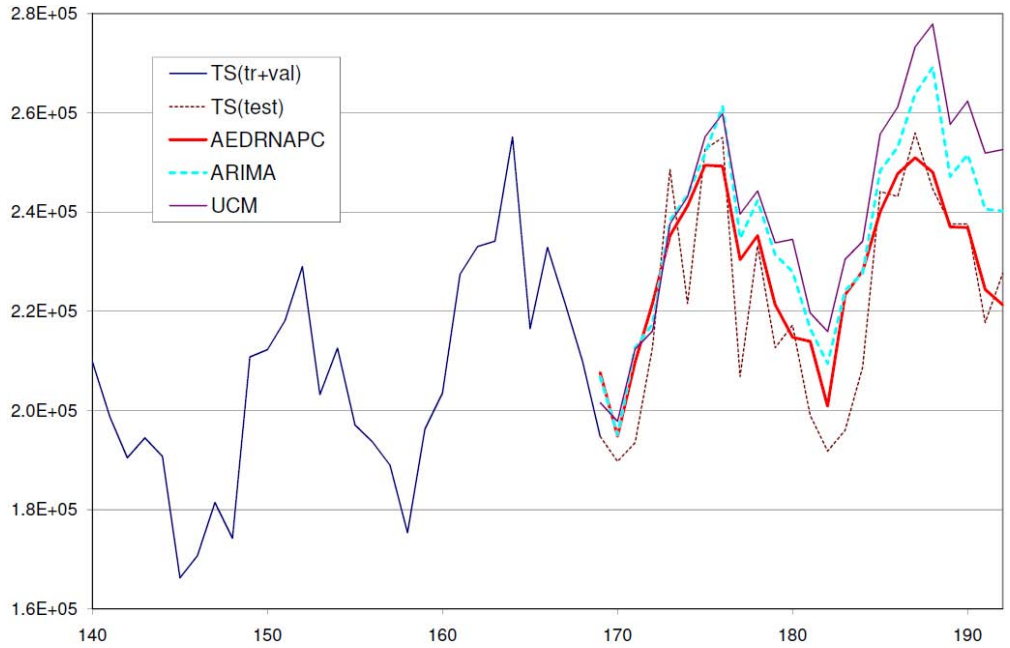


Figura 4.25: Zoom de la predicción de Abraham12 con AEDRNAPC, UCM y ARIMA.

como predicción un valor intermedio de los valores conocidos de forma continuada.

Dicha predicción, aunque gráficamente no es nada acertada (figura 4.23), ase-

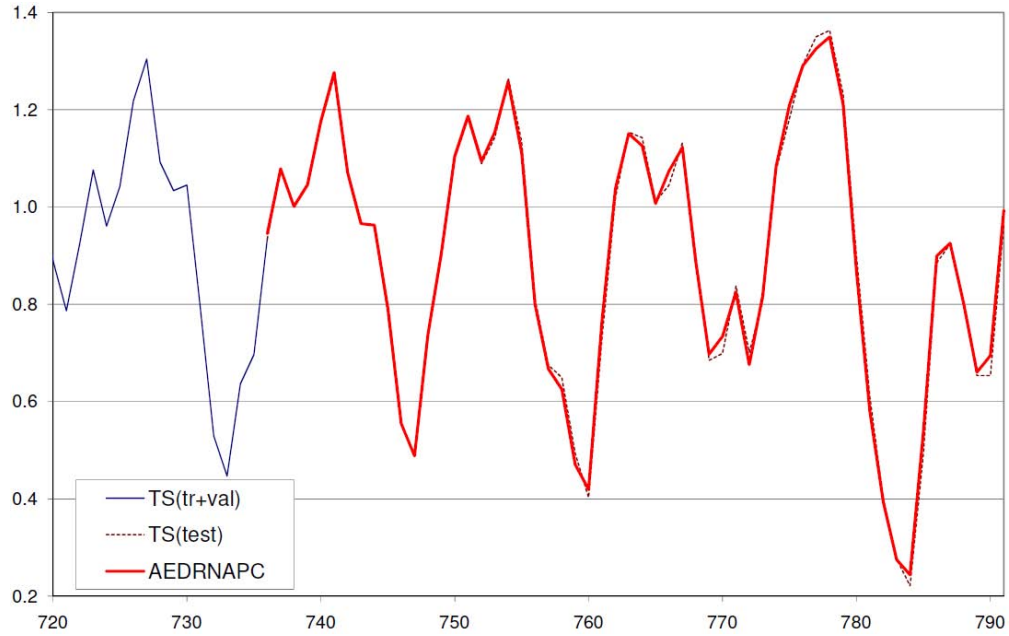


Figura 4.26: Zoom de la predicción de Mackey-Glass con AEDRNAPC, UCM y ARIMA.

gura un error menor, dando como resultado un error SMAPE de *4,78 %*, mejor que el obtenido por AEDRNAPC del *5,79 %*. Por lo tanto estos métodos estadísticos, al no poder detectar el ciclo de una serie temporal tienden a minimizar el error de la predicción mediante una predicción de un valor medio constante a pesar de que dicha predicción gráficamente no aporte información alguna a la hora de interpretarla.

4.5.2. Forecast Pro (FP)

En esta sección, se ha elegido la popular herramienta de predicción ForecastPro (FP) [202] como base de comparación. Este sistema experto permite analizar los datos, seleccionar la técnica más apropiada para pronosticar y calcular los pronósticos utilizando métodos estadísticos probados. Esta herramienta proporciona tres modos de trabajar con ella.

- Modo totalmente manual: requiere los conocimientos de un experto que seleccione el modelo o técnica estadística con la que trabajar, además de los parámetros óptimos a usar para una serie temporal específica.
- Modo semiautomático: el usuario debe seleccionar la técnica que se quiere

usar para la predicción. El sistema será el que determine los valores óptimos de los parámetros de la técnica usada para predecir una serie temporal.

- Modo totalmente automático o experto: el sistema se encargará de seleccionar la técnica usada y de buscar los mejores parámetros para llevar a cabo la predicción.

En particular, se alimenta la herramienta con los datos conocidos de las mismas seis series temporales usadas en el apartado anterior 4.5.1 y se ejecuta la función completamente automática de la herramienta para obtener las predicciones. La razón es usar un *"benchmark"* popular que puede compararse fácilmente y que no requiere de las capacidades expertas de selección de modelo por parte del usuario. El criterio de parada seleccionado para nuestro sistema evolutivo (AEDRNAPC) vuelve a ser de cien generaciones.

Las tablas 4.44 y 4.45 muestran los errores SMAPE y MSE obtenidos por AEDRNAPC y FP.

Tabla 4.44: SMAPE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AEDRNAPC y FP.

Series Temporales	AEDRNAPC	FP
Passengers	**3,20**	4,50
Temperature	4,17	**3,42**
Dow-Jones	5,79	**4,78**
Quebec	**8,00**	10,36
Abraham12	**4,48**	7,14
Mackey-Glass	**4,03**	26,15
Media	**4,94**	9,39
Mediana	**4,32**	5,96

Las predicciones obtenidas en este caso por AEDRNAPC, superan a las hechas por FP en cuatro de las seis series temporales seleccionadas.

Si tenemos en cuenta el error SMAPE, se puede observar en la tabla 4.44 mejoras de dos puntos porcentuales en Quebec e incluso de veintidós puntos en Mackey-Glass, donde FP no consigue hacer una buena predicción.

En general, AEDRNAPC obtiene una media de error que representa la mitad de la obtenida por FP, esto es debido al mal resultado obtenido por FP en Mackey-

Tabla 4.45: MSE para Passengers, Temperature, Dow-Jones, Quebec, Abraham12 y Mackey-Glass con AEDRNAPC y FP.

Series Temporales	AEDRNAPC	FP
Passengers	$\mathbf{0{,}68}\times\mathbf{10}^{-3}$	$0{,}75\times10^{-3}$
Temperature	$0{,}32\times10^{-2}$	$\mathbf{0{,}25}\times\mathbf{10}^{-2}$
Dow-Jones	$0{,}14\times10^{-1}$	$\mathbf{0{,}11}\times\mathbf{10}^{-1}$
Quebec	$\mathbf{0{,}11}\times\mathbf{10}^{-1}$	$0{,}13\times10^{-1}$
Abraham12	$\mathbf{0{,}13}\times\mathbf{10}^{-2}$	$0{,}29\times10^{-2}$
Mackey-Glass	$\mathbf{0{,}83}\times\mathbf{10}^{-3}$	$0{,}65\times10^{0}$
Media	$\mathbf{0{,}51}\times\mathbf{10}^{-2}$	$0{,}11\times10^{0}$
Mediana	$\mathbf{0{,}22}\times\mathbf{10}^{-2}$	$0{,}69\times10^{-2}$

Glass. Si nos fijamos en la mediana (última fila de la tabla), podemos observar como ésta también es favorable a AEDRNAPC frente a FP.

Podemos afirmar entonces que AEDRNAPC también se presenta como una buena opción a la hora de llevar a cabo predicciones frente a FP.

Obtenemos la desviación estándar de estos dos métodos para medir su correspondiente precisión. Como podemos observar en la tabla 4.46, el método más preciso y el que obtiene una menor media en los resultados es AEDRNAPC.

Tabla 4.46: Desviación estandar para AEDRNAPC y FP.

	AEDRNAPC	FP
Desv. Estand.	**1,71**	8,57
Media	**4,94**	9,39

Al llevar a cabo el test de Wilcoxon sobre estos dos métodos. El valor *"P-value"* obtenido para este par de métodos es de 0,160. Podemos entonces concluir que no existe evidencia clara de que haya una diferencia estadística significativa entre AEDRNAPC y FP.

Resultados Gráficos

Con el objetivo de poder observar de nuevo en más detalle las predicciones llevadas a cabo, a continuación se muestra una gráfica para cada serie temporal donde se representan los valores conocidos de estas (*tr+val*) y los futuros que hay que predecir (*test*), además de los valores predichos por cada uno de los sistemas comentados con anterioridad, AEDRNAPC y FP. En la figura 4.27 se muestra esta información para Passengers, en 4.28 podemos observar Temperature, por otro lado 4.29 hace lo propio para Dow-Jones, 4.30 para Quebec, Abraham12 en 4.31 y por último 4.32 muestra Mackey-Glass.

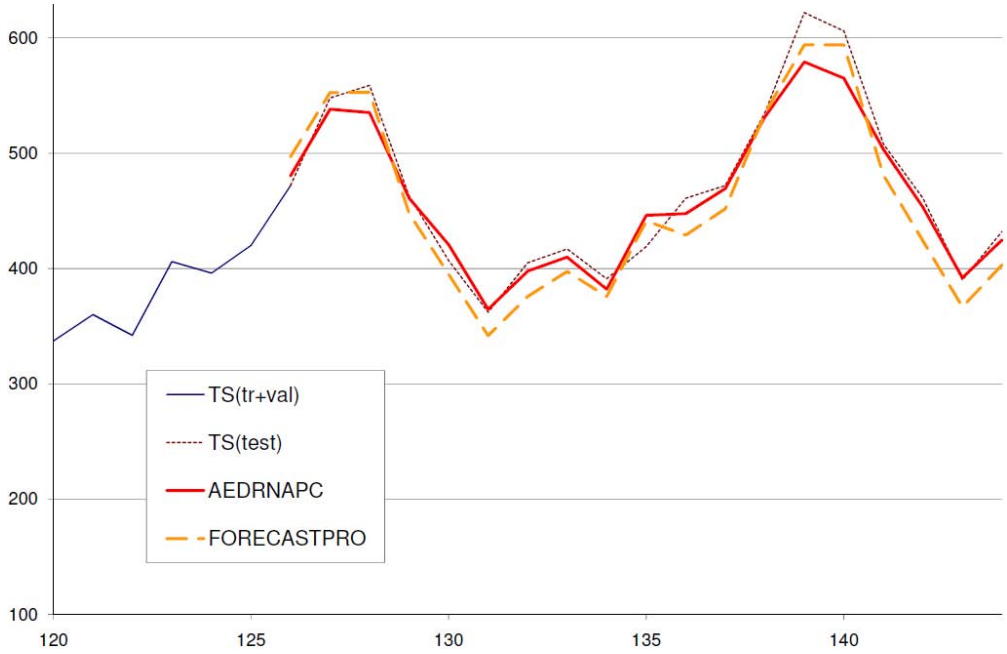


Figura 4.27: Zoom de la predicción de Passengers con AEDRNAPC y FP.

Ambos métodos (i.e. AEDRNAPC y FP) consiguen seguir la forma que tienen las series temporales en sus predicciones salvo en Dow-Jones y Mackey-Glass.

Al fijarnos con detenimiento en la serie temporal Dow-Jones, podemos observar que tan solo AEDRNAPC ha sido capaz de seguir la forma de los valores a predecir de la serie temporal, mientras que FP lleva a cabo una predicción de una línea recta. La razón es que FP hace uso de técnicas estadísticas para llevar a cabo las predicciones y como sucedía en la sección anterior, al ser Dow-Jones más irregular que las demás, esta es la mejor predicción que puede realizar FP al no existir en Dow-Jones un ciclo estacional claro, al menos con los datos proporcionados.

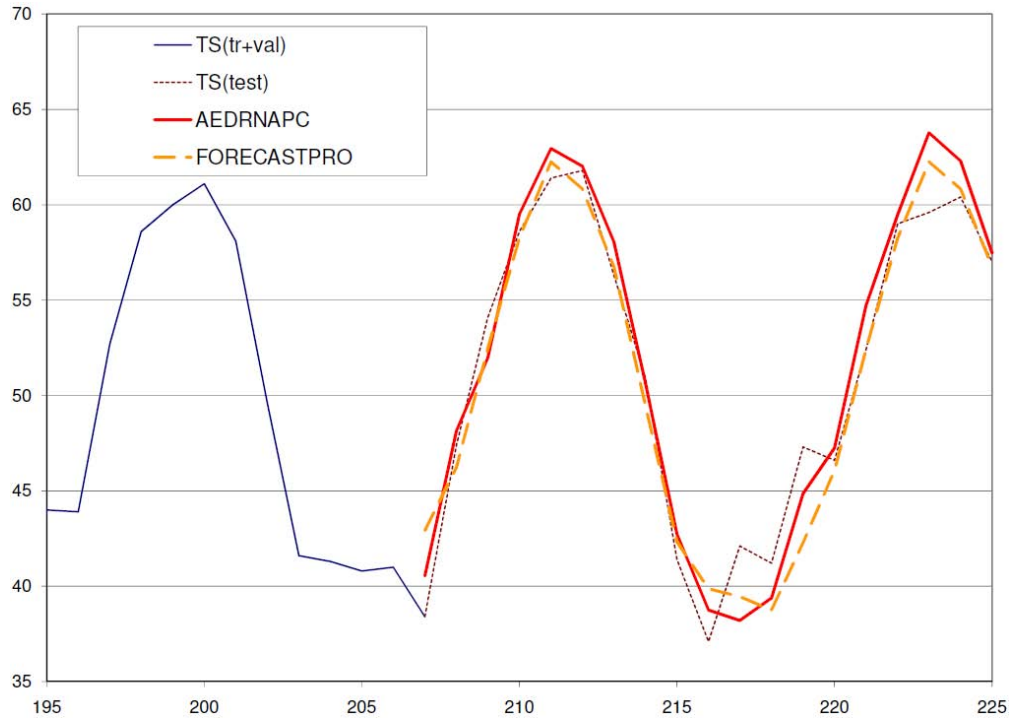


Figura 4.28: Zoom de la predicción de Temperature con AEDRNAPC y FP.

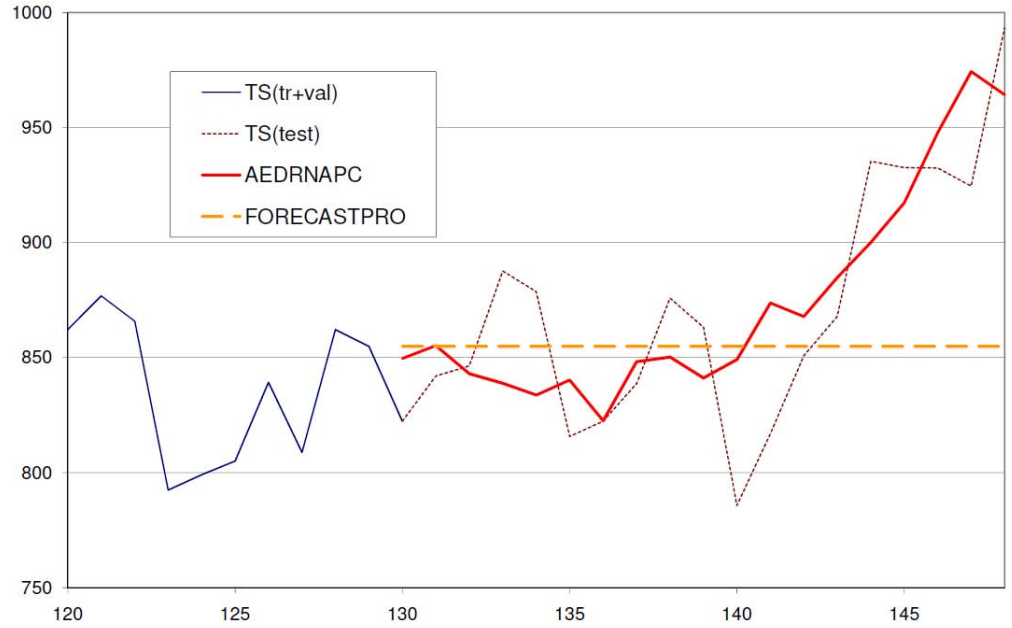


Figura 4.29: Zoom de la predicción de Dow-Jones con AEDRNAPC y FP.

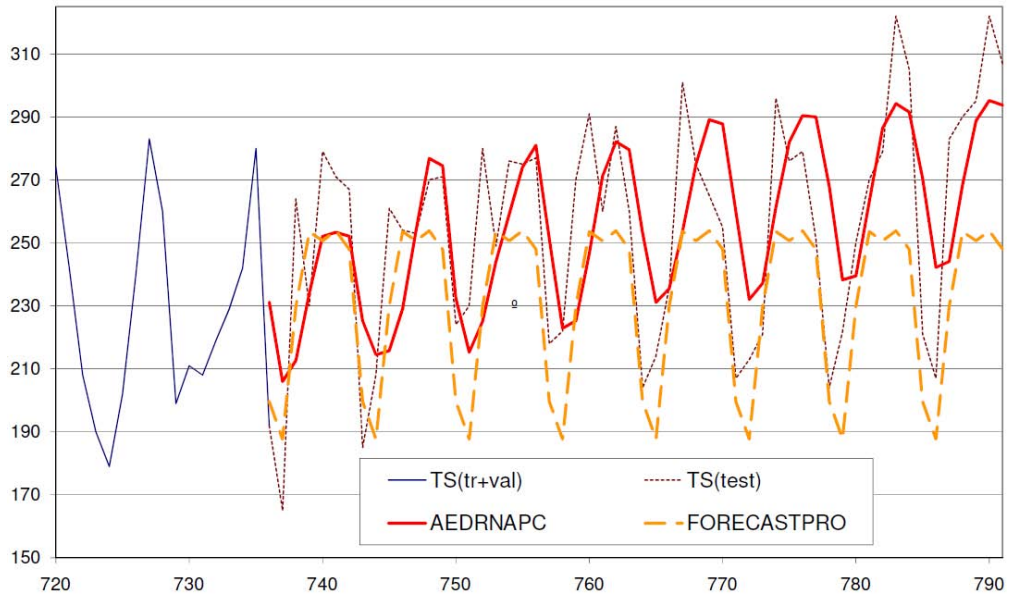


Figura 4.30: Zoom de la predicción de Quebec con AEDRNAPC y FP.

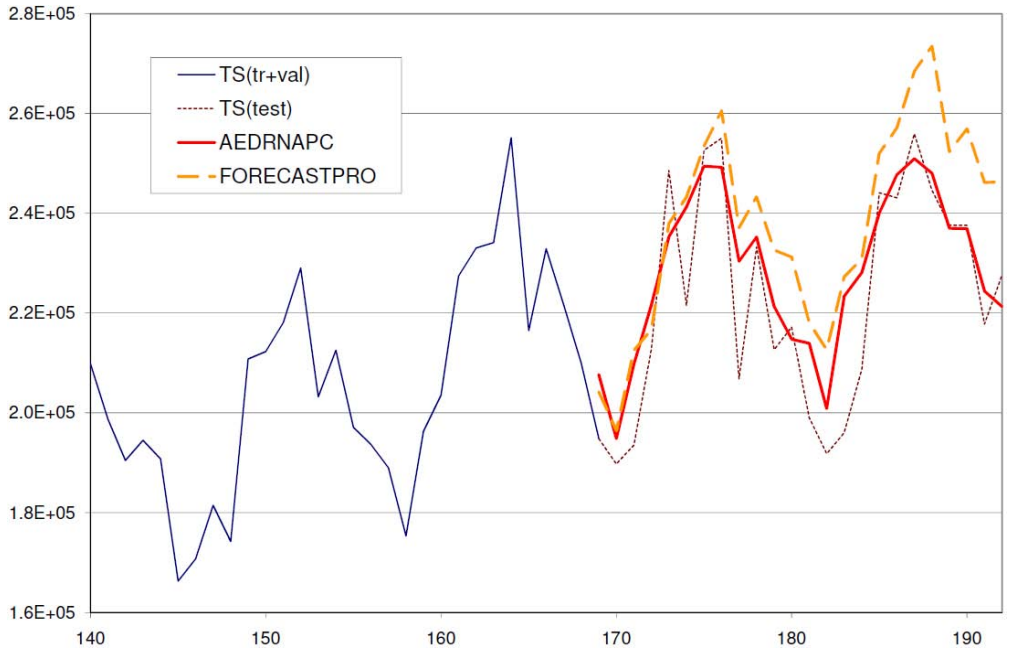


Figura 4.31: Zoom de la predicción de Abraham12 con AEDRNAPC y FP.

Esta predicción de una línea recta, aunque gráficamente no es nada acertada (figura 4.29), da como resultado un error SMAPE de *4,78 %*, mejor que el obtenido por AEDRNAPC del *5,79 %*.

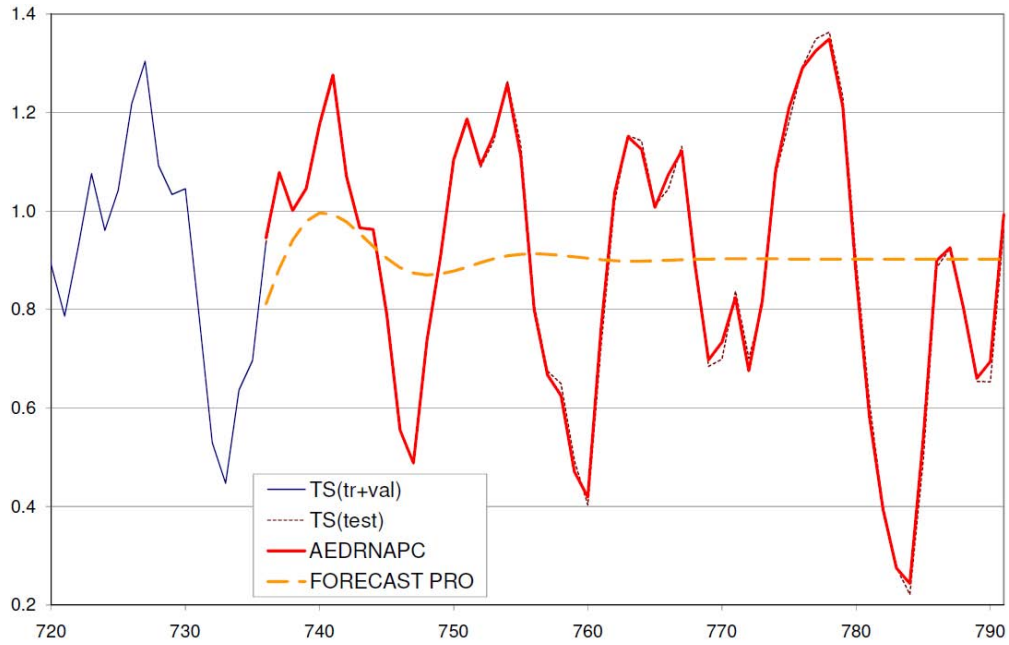


Figura 4.32: Zoom de la predicción de Mackey-Glass con AEDRNAPC y FP.

En cuanto a Mackey-Glass (figura 4.32), vuelve a repetirse lo que ocurría en el apartado anterior con métodos estadísticos. Aunque tiene un tamaño y forma parecidos a Quebec, en este caso FP no es capaz de predecir algo mejor que una línea recta. Esto puede ser debido a que Mackey-Glass es una serie temporal caótica generada a partir de ecuaciones diferenciales ordinarias como comentamos en el apartado 4.1. Por lo tanto, debido a su naturaleza, FP a riesgo de hacer una mala predicción prefiere predecir un valor medio constante de los valores conocidos. Aunque esta predicción obtiene un error menor, no aporta ninguna información gráficamente.

4.5.3. Métodos de Inteligencia Computacional

En esta sección, tres métodos basados en IC han sido utilizados para comparar nuestras predicciones contra las llevadas a cabo por dichos métodos.

El primero de los métodos presentados ha sido desarrollado por Paulo Cortez [216] y se basa en *"Máquinas de Soporte Vectorial"* (MSV). Las MSV [217, 218] son un tipo de conjunto de algoritmos de aprendizaje supervisado que fueron originalmente diseñada para la solución de problemas no lineales de clasificación [219, 220], pero que recientemente se han aplicado a problemas de regresión [218, 221] y predicción de series temporales [216, 222, 223]. Ello se debe a su

capacidad de generalización [224], la cual es función directa de su estructura y de la metodología utilizada para la estimación de sus parámetros.

Por otro lado, un estudio llevado a cabo por Martin Stepnicka et al. [11], presenta una aproximación basada en lingüística difusa enfocada en modelar y predecir la tendencia de una serie temporal dada.

Esto significa que los componentes estacionales tiene que ser predichos por separado y compuesto junto con la predicción de la tendencia. Los componentes estacionales pueden ser predichos estadísticamente, como se describe en [225, 226], aunque cualquier otra técnica de predicción de series temporales se puede utilizar también. Dado el alcance de este trabajo, se propone el uso de técnicas de IC, es decir, el uso de RNA y MSV, lo que lleva a dos sistemas híbridos nuevos haciendo uso de lógica difusa, denominados *"Fuzzy RNA"* (FRNA) y *"Fuzzy MSV"* (FMSV).

Debido a que este estudio comparativo [11] se llevó a cabo en colaboración con otros investigadores, Paulo Cortez y Martin Stepnicka, se modificó la batería de series temporales a utilizar. En esta ocasión, se ha hecho uso de las ya conocidas series temporales Passengers, Abraham12 y Mackey-Glass para llevar a cabo la experimentación además de otras tres series temporales nuevas [203] (i.e. Pigs, Cars y Milk) ya explicadas en detalle en la sección 4.1.

Se ha hecho uso del error SMAPE y MSE representado en las ecuaciones 4.2 y 4.1 para medir los resultados.

Las tablas 4.47 y 4.48 muestra los errores SMAPE y MSE para cada una de las series temporales comentadas con anterioridad.

Tabla 4.47: SMAPE para Passengers, Pigs, Cars, Abraham12, Milk y Mackey-Glass con AEDRNAPC, MSV, FMSV y FRNA.

Series Temporales	AEDRNAPC	MSV	FMSV	FRNA
Passengers	**3,2**	7,1	3,9	**2,5**
Pigs	11,5	**7,2**	7,8	8,2
Cars	**9,8**	10,0	11,0	11,0
Abraham12	**4,4**	6,9	5,9	5,6
Milk	2,3	1,3	**1,0**	1,1
Mackey-Glass	4,0	**2,4**	18,0	9,6
Media	**5,8**	**5,8**	8,1	6,2
Mediana	**4,2**	7,0	6,85	6,9

Tabla 4.48: MSE para Passengers, Pigs, Cars, Abraham12, Milk y Mackey-Glass con AEDRNAPC, MSV, FMSV y FRNA.

Series Temporales	AEDRNAPC	MSV	FMSV	FRNA
Passengers	$0{,}68\times10^{-3}$	$0{,}33\times10^{-2}$	$0{,}85\times10^{-3}$	$\mathbf{0{,}30\times10^{-3}}$
Pigs	$0{,}17\times10^{-1}$	$\mathbf{0{,}64\times10^{-2}}$	$0{,}74\times10^{-2}$	$0{,}87\times10^{-2}$
Cars	$\mathbf{0{,}86\times10^{-2}}$	$0{,}89\times10^{-2}$	$0{,}99\times10^{-2}$	$0{,}10\times10^{-1}$
Abraham12	$\mathbf{0{,}13\times10^{-2}}$	$0{,}29\times10^{-2}$	$0{,}20\times10^{-2}$	$0{,}19\times10^{-2}$
Milk	$0{,}85\times10^{-3}$	$0{,}29\times10^{-3}$	$\mathbf{0{,}16\times10^{-3}}$	$0{,}19\times10^{-3}$
Mackey-Glass	$0{,}83\times10^{-3}$	$\mathbf{0{,}35\times10^{-3}}$	$0{,}25\times10^{-1}$	$0{,}71\times10^{-2}$
Media	$0{,}48\times10^{-2}$	$\mathbf{0{,}36\times10^{-2}}$	$0{,}75\times10^{-2}$	$0{,}47\times10^{-2}$
Mediana	$\mathbf{0{,}10\times10^{-2}}$	$0{,}31\times10^{-2}$	$0{,}47\times10^{-2}$	$0{,}45\times10^{-2}$

Se puede observar en la tabla 4.47 que AEDRNAPC consigue mejores resultados en tres de las seis series temporales, y en términos generales (media y mediana) es el mejor método de los presentados.

AEDRNAPC obtiene diferencias importantes en las series Passengers y Abraham12 cuando comparamos con los demás métodos.

Las dos opciones basadas en lógica difusa solo son capaces de mejorar a AEDRNAPC y MSV en dos series temporales. FRNA empata en resultados con AEDRNAPC mientras que la opción basada en lógica y MSV (i.e. FMSV) solo es mejor que todas las demás en la serie temporal Milk.

El método MSV planteado por Paulo Cortez obtiene resultados equiparables a AEDRNAPC obteniendo mejores resultados en las series temporales Pigs y Mackey-Glass.

Nos disponemos a continuación a medir la precisión de los diferentes métodos presentados, mediante la desviación estándar de cada uno de ellos. Como podemos observar en la tabla 4.49, el método más preciso de los cuatro presentados es MSV seguido por AEDRNAPC.

El test de Wilcoxon ha sido lanzado por parejas sobre los diferentes métodos. Ninguno de ellos ha evidenciado claramente que haya una diferencia estadística significativa entre los resultados obtenidos por estos métodos.

Tabla 4.49: Desviación estandar para AEDRNAPC, MSV, FMSV y FRNA.

	AEDRNAPC	MSV	FMSV	FRNA
Desv. Estand.	3,81	**3,29**	5,98	3.96
Media	**5,8**	**5,8**	8,1	6,2

Resultados Gráficos

Para poder observar de nuevo en más detalle las predicciones llevadas a cabo, a continuación se muestra una gráfica para cada serie temporal donde se muestran los valores conocidos de estas (*tr+val*) y los futuros que hay que predecir (*test*), además de los valores predichos por cada uno de los sistemas comentados con anterioridad, AEDRNAPC, MSV, FMSV y FRNA. La figura 4.33 muestra esta información para Passengers, en 4.34 podemos observar Pigs, por otro lado 4.35 hace lo propio para Cars, 4.36 para Abraham12, 4.37 enseña Milk y por último 4.38 muestra Mackey-Glass.

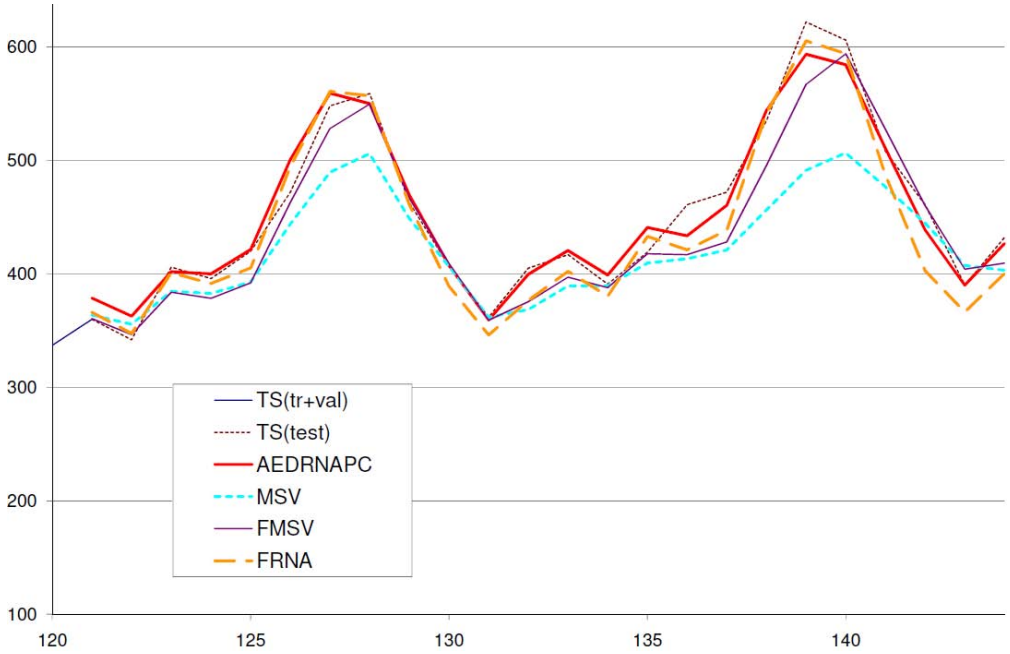


Figura 4.33: Zoom de la predicción de Passengers con AEDRNAPC, MSV, FMSV y FRNA.

En la figura 4.34 podemos observar como se repite una situación ya dada en experimentación anterior. El método MSV obtiene como predicción una línea

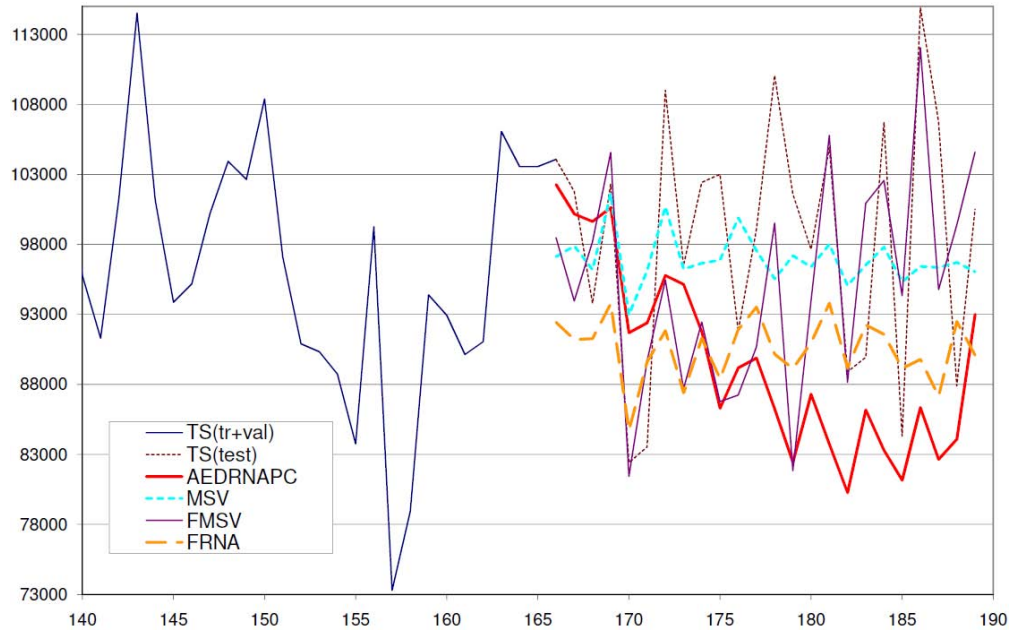


Figura 4.34: Zoom de la predicción de Pigs con AEDRNAPC, MSV, FMSV y FRNA.

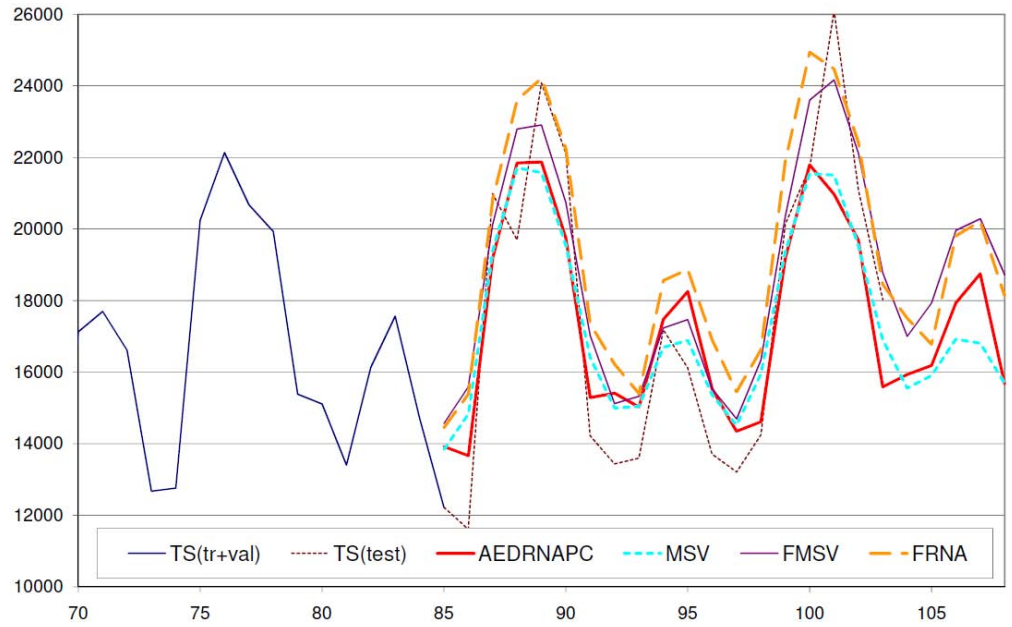


Figura 4.35: Zoom de la predicción de Cars con AEDRNAPC, MSV, FMSV y FRNA.

casi recta. Como ya ocurrió con anterioridad, aunque gráficamente esta predicción no es nada atractiva, resulta la mejor opción para obtener un error menor que el obtenido por el resto de métodos.

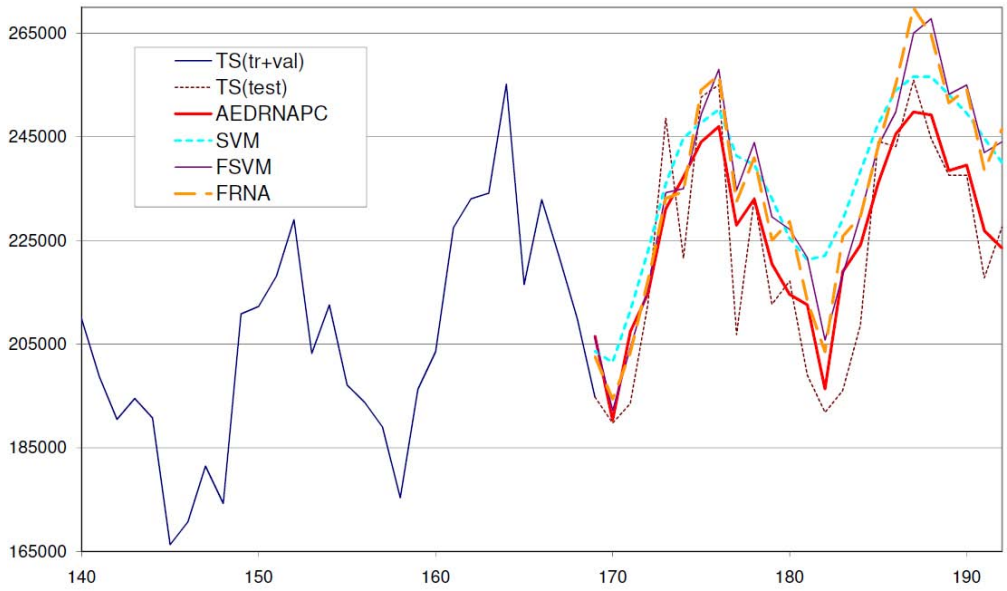


Figura 4.36: Zoom de la predicción de Abraham12 con AEDRNAPC, MSV, FMSV y FRNA.

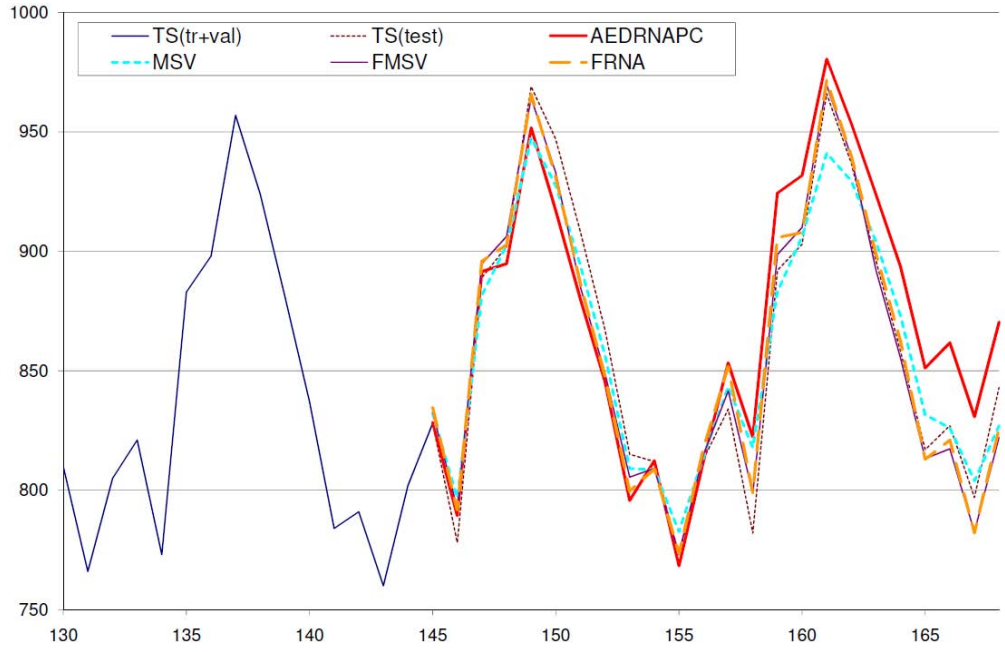


Figura 4.37: Zoom de la predicción de Milk con AEDRNAPC, MSV, FMSV y FRNA.

Se puede concluir por tanto que AEDRNAPC es uno de los mejores (si no el mejor) método computacional de los presentados en esta tesis doctoral.

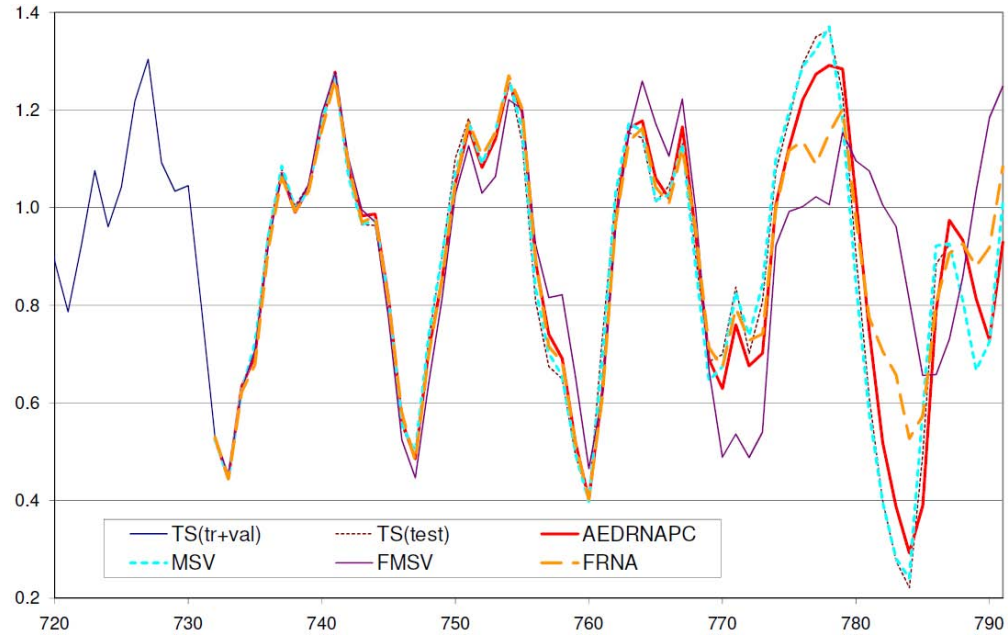


Figura 4.38: Zoom de la predicción de Mackey-Glass con AEDRNAPC, MSV, FMSV y FRNA.

4.6. Conclusiones

En este capítulo se ha recopilado toda la experimentación realizada sobre cual es la mejor función de evaluación, y resolver algunas dudas nuevas planteadas, como si el método *"Shuffle"* ayudaría a mejorar la predicción de series temporales.

Primero se ha detallado la batería de series temporales de pruebas y las medidas de error que iban a ser usadas a lo largo del capítulo.

Más tarde nos hemos centrado en el proceso de normalización de los datos, proceso previo e imprescindible para toda predicción de series temporales.

Se ha corroborado empíricamente lo avanzado por Schaffer [123] sobre el uso del conjunto de validación como función de evaluación, dando esta función mejores resultados que las otras dos propuestas.

Se han estudiado los datos de entrada dados por la serie temporal que intentamos predecir y como tratar dichos datos para que al presentárselos a las RNA podamos obtener unos mejores resultados.

Para ello, se optó por llevar a cabo un nuevo método llamado *"Shuffle"*. Con él se barajan los patrones de entrada de las RNA y podemos usar partes aleatorias de la serie temporal para aprender y validar en lugar de usar el comienzo de la

serie para aprender y el final para validar.

Los resultados no fueron del todo positivos porque aunque mejoraban en tres de las series series temporales usadas como batería de pruebas, en otras tres los resultados obtenidos eran peores.

Para mejorar el aprendizaje en series temporales con pocos elementos (menos de 30) se propuso emplear validación cruzada y obtener así el máximo partido de dichos elementos.

Para ello se optó por dividir el conjunto de patrones en un número de subconjuntos (entre 2 y 8) y usar validación cruzada. Pero llegados a este punto apareció otro problema, si una misma topología de RNA se entrena con diferentes conjuntos de patrones, se obtienen como resultado diferentes arquitecturas.

Para solucionar este problema en el que una misma RNA con n subconjuntos de patrones genera n arquitecturas diferentes, se hizo uso de la combinación (o *"Ensemble"*) de todas y cada una de las salidas dadas por cada una de estas arquitecturas finales.

Los resultados obtenidos demuestran que la versión ponderada de validación cruzada obtiene mejores resultados y el mejor método de combinación o *"Ensembles"* es CLBR.

En cuanto a la computación evolutiva usada para llevar a cabo la búsqueda global en el sistema, los resultados de los experimentos revelan que usar ED y AED en lugar del AG obtiene buenos resultados y mejoran al AG.

Con tan solo cien generaciones, ED y AED mejoran los resultados comparado con AG siendo la media y la mediana de sus resultados mejores que la del AG.

En general, los resultados obtenidos con ED y AED hacen de estos dos métodos unas buenas opciones a la hora de ser usados para llevar a cabo una búsqueda global en un sistema híbrido como el nuestro para el diseño de RNA.

Si tenemos en cuenta que además AED no realiza mutación ni sobrecruzamiento y que ED consume más tiempo de ejecución que los otros dos métodos, la mejor opción es usar AED como método evolutivo.

Referente a la arquitectura de las RNA usadas, se han presentado dos nuevas alternativas.

AEDRNA es una modificación de ADANN donde pasamos a usar AED como método evolutivo y RPROP como algoritmo de aprendizaje. Se corroboró empíricamente lo ya avanzado por Riedmiller et al. [88], donde el algoritmo RPROP obtiene resultado similares al de retropropagación consumiendo menos tiempo computacional.

AEDRNAVD fue una modificación del sistema AEDRNA donde incluimos una ventana deslizante para seleccionar qué instantes anteriores específicos debemos usar para realizar una mejor predicción. Este nuevo sistema obtiene mejores resultados, modelos de redes más pequeños y consume menos tiempo que AEDRNA.

Además se obtuvo una nueva versión partiendo de AEDRNA, AEDRNAPC. En ella el perceptrón usado pasa a ser parcialmente conectado mediante una codificación binaria de las conexiones. Esta nueva versión (i.e. AEDRNAPC) mejora de nuevo a AEDRNA en resultados, modelos obtenidos y tiempo.

En este capítulo se ha llevado a cabo también una batería de pruebas considerables para poder comparar el sistema propuesto en esta tesis doctoral y más concretamente la aproximación AEDRNAPC presentada en la sección 4.4, frente a diferentes métodos de predicción de series temporales.

Estos métodos provienen de campos tan dispares como el estadístico, herramientas software usadas para predicción en el mundo empresarial y por último tres métodos basados en IC.

Se ha podido observar como AEDRNAPC obtiene buenos resultados comparados con todos ellos. En el caso de los métodos estadísticos, existe una serie temporal que tan solo AEDRNAPC ha podido predecir ya que estos métodos precisan de información previa sobre la serie temporal y conocimiento experto.

El sistema propuesto en esta tesis doctoral a diferencia de éstos, tan sólo necesita saber que serie temporal quiere ser predicha y cuantos valores futuros se requieren conocer. AEDRNAPC no precisa entonces de conocimiento experto por parte del usuario.

Al llevar a cabo la comparación de AEDRNAPC respecto a la herramienta software antes comentada (FP), se ha podido comprobar que nuestro sistema mejora considerablemente en tres de los cinco casos propuestos, además de obtener un error medio que representa la mitad del obtenido por FP.

En último lugar, al ser comparado AEDRNAPC con otros métodos computacionales (MSV, FMSV y FRNA), se ha podido comprobar que al menos uno de ellos (MSV) es mejor de media que AEDRNAPC, aunque nuestro sistema tiene una mejor mediana y además supera a MSV en la mitad de los casos propuestos.

Por lo tanto, AEDRNAPC es una herramienta de predicción de series temporales que no requiere de un usuario experto. El usuario tan solo debe indicarle sobre qué serie temporal quiere calcular la predicción y cuántos valores futuros quiere predecir.

Por todo lo anteriormente explicado, podemos concluir que nuestro sistema en su versión AEDRNAPC es una buena opción a usar a la hora de llevar a cabo

predicción de series temporales. AEDRNAPC reduce los errores obtenidos hasta ahora, además del tiempo computacional y es comparable a otros métodos de predicción provenientes de diferentes campos.

Capítulo 5

Conclusiones Generales y Trabajos Futuros

En este capítulo se presenta un resumen de lo realizado en esta tesis doctoral, además de las conclusiones finales y los trabajos que se podrían llevar a cabo en un futuro para complementar el trabajo desarrollado.

5.1. Sumario

En esta tesis doctoral se ha presentado y evaluado un sistema para el diseño de redes de neuronas artificiales aplicando éstas a un dominio específico, la predicción de series temporales. Además, se han ido proponiendo y probando mejoras al sistema, tanto en el mecanismo de búsqueda como en la manera de tratar los propios datos de entrada con el objetivo de mejorar los resultados obtenidos además de disminuir el tiempo computacional consumido.

Para poder entender mejor la tarea que se ha llevado a cabo, se parte desde la idea principal de que las redes de neuronas artificiales pueden aproximar cualquier función continua a cualquier precisión [86, 93, 94]. Éstas tienen más formas funcionales flexibles y generales que los tradicionales métodos estadísticos [95]. El problema que se aborda en esta tesis doctoral es, qué red de neuronas usar para predecir una serie temporal específica, es decir, como generar o diseñar la red que va a predecir dicha serie temporal.

Para ello, se estudiaron en detalle los diferentes métodos empleados para diseñar redes de neuronas encontrados en la bibliografía y se observó que de entre todos ellos, la computación evolutiva obtenía buenos resultados.

También se comprobó que aunque una red de neuronas entrenada con un algoritmo de aprendizaje obtiene mejores resultados que cuando se evolucionan los pesos de las conexiones por medio de algoritmos evolutivos, son los sistemas híbridos los que superan a todos. Es decir, aquellos compuestos primero por una búsqueda global de los parámetros de las redes por parte de un algoritmo evolutivo, afinando después los resultados por medio de un entrenamiento realizado por un algoritmo de aprendizaje.

Respecto a los parámetros a codificar dentro del cromosoma, se comprobó que los estudios han tratado por separado el diseño de RNA desde tres puntos de vista diferentes:

- Pesos de las conexiones: Valor para cada conexión presente en una RNA.
- Topología: Número de nodos en cada capa oculta, número de capas ocultas, número de nodos en la capa de entrada y salida.
- Reglas de aprendizaje: Factor de aprendizaje y valores del momento.

En nuestra aproximación se ha tratado de tener en cuenta todos estos puntos. Por lo tanto, hemos seleccionado como parámetros de las RNA sus topologías, los pesos de conexión e incluso las reglas de aprendizaje, en forma de factor de aprendizaje asociado al algoritmo BP. Es por ello que se llevó a cabo una codificación de todos estos parámetros más importantes de las redes de neuronas estudiados en el estado del arte y se propuso el sistema llamado ADANN.

Dicha aproximación obtuvo buenos resultados en la competición internacional NN5 quedando sexta en el ranking general con un error SMAPE del 21,9 %. La herramienta software comercial de predicción denominada Autobox [213], basada en el método estadístico Box-Jenkins, obtuvo un error de 23,9 % en dicha competición.

Una vez que se disponía de una base desde la que partir (i.e. ADANN) el objetivo era mejorarla *"atacando"* el problema desde tres puntos diferentes.

Primero nos centramos en el tratamiento que se le da a los datos de entrada del sistema. Para ello, se propusieron diferentes métodos.

Por un lado, se implementó un nuevo método denominado *"Shuffle"* (del inglés barajar) que se basa en desordenar los parámetros que forman los subconjuntos de patrones para entrenar y evaluar las RNA (i.e. entrenamiento y validación).

Se propusieron además, dos nuevas funciones de evaluación haciendo uso de un nuevo subconjunto de patrones (validación II) a parte de los dos subconjuntos ya conocidos (entrenamiento y validación).

Además, cuando por circunstancias ajenas al usuario, no se dispone de muchos datos o las series temporales a predecir son cortas, se propuso un sistema novedoso de validación cruzada para que las redes pudieran aprender mejor. Para ello hubo que apoyarse en la teoría de conjuntos o *"Ensembles"*.

Segundo, estudiamos la evolución en el sistema, lo que conocemos como aprendizaje global. Se presentaron dos algoritmos evolutivos alternativos al algoritmo genético usado hasta la fecha. Estos son el algoritmo de Evolución Diferencial (ED) y el Algoritmo de Estimación de Distribución (AED). Ambos mejoraron los resultados obtenidos por la primera aproximación y el último de ellos fue usado en la competición nacional SICO, celebrada en Valencia, donde nuestro sistema quedó tercero en el ranking general.

Por último, se quiso mejorar un punto flaco de nuestro sistema, el coste computacional. Para ello, se llevó a cabo una modificación que consistía en hacer uso de otras arquitecturas de red en lugar del perceptrón multicapa totalmente conectado utilizado en la primera aproximación.

Se demostró que haciendo uso de una ventana deslizante, con el objetivo de seleccionar en detalle los instantes previos de la serie temporal para predecir los valores futuros, mejoraban los resultados obtenidos hasta la fecha. Así, se redujo el número de nodos de entrada y por lo tanto el de conexiones, no solo se obtuvieron modelos más sencillos si no que además acortaba considerablemente el coste computacional.

Por otro lado, se optó también por usar el perceptrón multicapa parcialmente conectado con lo que se aumentó la capacidad de generalización de las redes y se redujo de nuevo el coste computacional y los modelos resultantes.

Al final de esta tesis doctoral se han llevado a cabo diferentes estudios comparativos de nuestro sistema frente a técnicas estadísticas, la herramienta software de predicción FP y otros métodos computacionales.

5.2. Conclusiones

La predicción de series temporales es un campo de estudio en continua y rápida expansión. Como se ha mostrado en el capítulo 2, son muchos los trabajos que aparecen cada año intentando mejorar las predicciones llevadas a cabo. Esto es debido a que este campo puede ser aplicado a múltiples problemas planteados en el mundo real. Desde casos como la predicción meteorológica hasta la predicción de los valores de bolsa o la frecuencia y magnitud de los terremotos.

En nuestro caso, se desea definir un modelo en forma de red de neuronas arti-

ficial para predecir una serie temporal específica. La definición de dicho modelo puede ser una tarea compleja.

En la mayoría de las investigaciones actuales se proponen sistemas aplicados a un dominio concreto, o bien con la necesidad de mano experta para ser usados. Algunos incluso plantean la necesidad de disponer de información previa sobre la serie temporal para poder llevar a cabo una buena predicción.

Esto no ocurre en el sistema propuesto aquí, ya que en este trabajo se presenta un sistema semiautomático que no precisa del conocimiento de un experto ni de información previa de la serie temporal para poder predecir sus valores futuros. El usuario tan solo tendrá que indicarle a nuestro sistema qué serie temporal quiere predecir y cuántos valores futuros quiere conocer, dando el sistema como resultado dichos valores predichos.

El objetivo principal de esta tesis doctoral consistía en definir un marco que permitiera crear modelos de redes de neuronas artificiales para predecir series temporales. A diferencia de otros estudios, cabe destacar como novedad principal en este trabajo, el que se hayan seleccionado como parámetros de las RNA sus topologías, los pesos de conexión e incluso las reglas de aprendizaje conjuntamente. Debido a las características particulares del dominio de la predicción, para conseguir este objetivo se ha propuesto, desarrollado y evaluado una primera aproximación de dicho sistema denominada ADANN.

Para poder llevar a cabo esta primera aproximación conocida como ADANN, debemos tener en cuenta diferentes pasos importantes a estudiar:

- Normalización de los datos de entrada.
- Gestión de estos para poder producir los patrones con los que los modelos o redes de neuronas artificiales serán entrenadas y evaluadas.
- Una primera búsqueda a nivel global en el espacio de todas las posibles soluciones llevada a cabo por computación evolutiva.
- Dentro de cada generación del algoritmo evolutivo, se llevará a cabo una búsqueda local o entrenamiento mediante un algoritmo de aprendizaje para afinar los pesos ya seleccionados por la búsqueda evolutiva o global.
- Selección del individuo o individuos que serán usados para predecir los valores futuros.
- Predicción de los valores futuros y medida de los errores obtenidos por los resultados.

Una vez que se parte de una primera aproximación robusta (ADANN), se ha hecho frente a tres posibles mejoras por separado.

La primera de ellas, centrándonos en los datos de entrada que se le suministran a los modelos. En esta primera mejora se trataba de medir si la manera de presentar los datos con los que serán entrenadas y evaluadas las redes de neuronas es importante.

Para ello se tuvieron en cuenta dos casos diferentes. El primero de ellos, aquel en que la serie temporal dispone de multitud de elementos. Se planteó entonces generar el conjunto total de patrones donde a la hora de dividirlos entre entrenamiento y validación, fueron dados de manera ordenada (primera aproximación) y de manera aleatoria (desordenada). Se pudo observar que el nuevo método de presentar los patrones a las redes no obtenía mejoras significativas en los resultados comparado con la primera aproximación.

Se presentó entonces otro problema, cómo poder mejorar las predicciones de series temporales cortas, es decir, aquellas con pocos elementos.

Para ello, se hizo uso del conocido método de validación cruzada. Este método permitiría dividir el conjunto total de patrones en n subconjuntos y hacer uso de todos y cada uno de ellos para el proceso de entrenamiento y el de validación.

Este nuevo método no solo obtuvo mejoras significativas en los resultados sino que nos ayudó a aclarar el número de conjuntos óptimo en los que dividir la validación cruzada para toda serie temporal. Además se presentaron cuatro métodos de *"Ensembles"* diferentes, demostrando empíricamente que el mejor de ellos era CLBR [130].

En la segunda mejora propuesta en esta tesis doctoral, se planteó la pregunta de que algoritmo evolutivo usar para mejorar los resultados obtenidos hasta la fecha.

Por ello, se utilizó el algoritmo de evolución diferencial y el algoritmo de estimación de distribución para llevar a cabo la búsqueda global del sistema y fueron comparados con la primera aproximación de este sistema donde se hacía uso de un algoritmo genético para dicha tarea.

Quedó demostrado que tanto el algoritmo de estimación de distribución como el de evolución diferencial obtenían mejores resultados que la aproximación con algoritmo genético.

Cabe destacar también, que al no existir sobrecruzamiento en AED evitamos la posible interrupción causada por el sobrecruzamiento [124], además de reducir el número de parámetros necesarios para realizar la experimentación.

Además el estudio llevado a cabo con algoritmo de estimación de distribución se puede considerar novedoso, pues no se ha encontrado ningún estudio previo de diseño de redes de neuronas con dicho algoritmo para predecir series temporales.

Por otro lado, la aproximación en la que se usa el algoritmo de estimación de distribución se presentó en la competición nacional SICO de Valencia obteniendo el tercer puesto en predicción de series temporales.

En la última mejora, se pretendía desarrollar tres puntos. Uno seguía siendo los resultados obtenidos, el segundo era intentar disminuir el tiempo computacional invertido en obtener las predicciones y el tercero reducir los modelos obtenidos como soluciones.

Para ello se plantearon dos nuevas propuestas con un mismo objetivo, reducir el número de conexiones sin perjudicar los resultados.

La primera consistía en hacer uso de una ventana deslizante que de manera automática (incluida en la búsqueda global) seleccionara con más detalle que valores previos eran realmente necesarios para predecir el futuro y cuales podían ser prescindibles. Con esta nueva modificación no solo se mejoró los resultados obtenidos si no que además se obtuvieron importantes mejoras en el tiempo invertido en la experimentación. Además, los modelos resultantes eran más simples.

La otra propuesta fue la de intentar reducir el número de conexiones haciendo uso de un perceptrón multicapa parcialmente conectado. Así, sería posible mediante una codificación previa de las conexiones en el cromosoma, de eliminar conexiones no necesarias y reducir el tiempo de entrenamiento. Este objetivo también se alcanzó con esta nueva aproximación y se obtuvieron resultados mejores tanto de tiempo computacional como de errores de predicción. Además, se obtuvieron modelos más simples como resultado.

En general, podemos concluir que en esta tesis doctoral se ha llevado a cabo un estudio profundo del uso de redes de neuronas artificiales evolutivas para predecir series temporales.

Además, los temas tratados (datos de entrada, CE y arquitectura de las RNA) nos presentan mejoras interesantes a la hora de diseñar redes de neuronas y como mejorar los resultados que estas puedan obtener. Esta tesis doctoral también da pie a futuros trabajos o estudios que se quieran llevar a cabo para mejorar cualquiera de los tres temas tratados o de la aproximación base de la que se partió desde un principio (ADANN).

Por último, resaltar que el sistema propuesto es esta tesis doctoral es una buena opción a la hora de predecir series temporales comparado con métodos estadísticos, herramientas software comerciales y otros métodos computacionales.

5.3. Trabajos Futuros

La futura línea de investigación más importante que se podría plantear como trabajo futuro en esta tesis doctoral, sería ampliar la experimentación llevada a cabo en este estudio. Dicha experimentación podría otorgarnos más luz sobre las modificaciones aquí planteadas y poder así tener una mejor medida de éstas.

Además, los tres puntos de mejora propuestos en esta tesis doctoral (datos de entrada, CE y arquitectura de las RNA) pueden ser ampliados y estudiados en más detalle.

En cuanto al primer punto, sobre la manera de tratar los datos y como presentar estos a las redes de neuronas, sería interesante estudiar en más detalle la proporción que se debe dar al subconjunto de entrenamiento y al subconjunto de validación.

Sobre las futuras líneas de investigación del segundo punto, mejora de la computación evolutiva, los posibles trabajos futuros que se podrían llevar a cabo para intentar mejorar los ya buenos resultados obtenidos por el algoritmo de evolución diferencial y el algoritmo de estimación de distribución son:

- Al usar el algoritmo de evolución diferencial, en lugar de usar un $\vec{x_{r0}}$ aleatorio, hacer uso del mejor individuo de la población como se ha realizado en la tesis doctoral [227].

- De nuevo para el algoritmo de evolución diferencial, hacer uso de más vectores (individuos) para obtener más variación a la hora de calcular la diferencia. Por ejemplo ($\vec{x_{r1}}$-$\vec{x_{r2}}$+$\vec{x_{r3}}$-$\vec{x_{r4}}$) en lugar de la diferencia simple ($\vec{x_{r1}}$-$\vec{x_{r2}}$) que usamos en este trabajo.

- En cuanto al algoritmo de estimación de distribución, sería interesante calcular las posibles dependencias entre variables antes de calcular la probabilidad de distribución. Para ello se podría hacer uso de las estimaciones de distribución MIMIC con dependencias de orden uno entre variables o incluso hacer uso de *"TREE"*, sin restricciones en el número de dependencias entre variables.

Respecto al último punto, orientado a mejorar la arquitectura de los modelos propuestos, un trabajo futuro interesante sería combinar las técnicas ventana deslizante y perceptrón multicapa parcialmente conectados. Ésto nos ayudaría a comprobar si ambas técnicas unidas mejoran los resultados y reducen aún más el tiempo de computación y los modelos resultantes.

Por último, se quiere resaltar de nuevo la cantidad y variedad de dominios en los que se puede aplicar el diseño automático de redes de neuronas artificiales. Además del dominio de la predicción de series temporales, utilizado en esta tesis doctoral, se propone como línea de trabajo futura la aplicación de la mejoras aquí presentadas en otros dominios. Por ejemplo, el diseño de redes de neuronas para controlar BOTS [228] o personajes no jugadores en juegos de ordenador en primera persona como puede ser el Unreal Tournament, o cómo controlar un coche en un juego de carreras mediante estas técnicas [229]. Otro dominio al que podría ser aplicado lo estudiado en esta tesis doctoral es a resolver problemas de clasificación [166].

5.4. Publicaciones

Esta tesis doctoral ha dado origen a las publicaciones que se muestran a continuación.

Título:	*Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble*
Autores:	J. Peralta, Paulo Cortez, G. Gutierrez, A. Sanchis
Revista:	Neurocomputing
DOI:	
Fecha:	Febrero, 2012

Título:	*Time series forecasting. A comparative study between an evolving artificial neural networks system and statistical methods*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Revista:	International Journal on Artificial Intelligence Tools
DOI:	10.1142/S0218213011000462
Fecha:	Agosto, 2011

Título:	*Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm*
Autores:	J. Peralta, Xiaodong Li, G. Gutierrez, A. Sanchis
Revista:	Neural Computing and Applications
DOI:	10.1007/s00521-011-0741-0
Fecha:	Septiembre, 2011

Título:	*Evolving Time-Lagged Feedforward Neural Networks for Time Series Forecasting*
Autores:	J. Peralta, P. Cortez, G. Gutierrez, A. Sanchis
Congreso:	GECCO 2011
Publicación:	Actas del GECCO 2011
Lugar:	Dublín, Irlanda
Fecha:	Julio, 2011

Título:	*Evolving Sparsely Connected Neural Networks for Multi-Step Ahead Forecasting*
Autores:	J. Peralta, P. Cortez, G. Gutierrez, A. Sanchis
Congreso:	GECCO 2011
Publicación:	Actas del GECCO 2011
Lugar:	Dublín, Irlanda
Fecha:	Julio, 2011

Título:	*Forecasting seasonal time series with computational intelligence: contribution of a combination of distinct methods*
Autores:	M. Stepnicka, J. Peralta, P. Cortez, Lenka Vavrickova, G. Gutierrez
Congreso:	EUSFLAT 2011
Publicación:	Actas del EUSFLAT 2011
Lugar:	Aix-Les-Bains, Francia
Fecha:	Julio, 2011

Título:	*Weighted cross-validation evolving artificial neural networks to forecast time series*
Autores:	J. Peralta, P. Cortez, G. Gutierrez, A. Sanchis
Congreso:	SOCO 2011
Publicación:	Actas del SOCO 2011
Lugar:	Salamanca, España
Fecha:	Abril, 2011

Título:	*Time series forecasting by evolving artificial neural networks using "Shuffle", cross-validation and ensembles*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	ICANN 2010
Publicación:	Actas del ICANN 2010
Lugar:	Tesalonika, Grecia
Fecha:	Septiembre, 2010

Título:	*Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution*
Autores:	J. Peralta, Xiaodong Li, G. Gutierrez, A. Sanchis
Congreso:	IJCNN 2010
Publicación:	Actas del WCCI-IJCNN 2010
Lugar:	Barcelona, España
Fecha:	Julio, 2010

Título:	*Time series forecasting by evolving artificial neural networks using genetic algorithms and estimation of distribution algorithms*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	IJCNN 2010
Publicación:	Actas del WCCI-IJCNN 2010
Lugar:	Barcelona, España
Fecha:	Julio, 2010

Título:	*Automatic Design of Artificial Neural Networks to forecast time series*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	CEDI 2010
Publicación:	Actas del CEDI 2010
Lugar:	Valencia, España
Fecha:	Septiembre, 2010

Título:	*Shuffle Design to Improve Time Series Forecasting Accuracy*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	CEC 2009
Publicación:	Actas del CEC 2009
Lugar:	Trondheim, Noruega
Fecha:	Mayo, 2009

Título:	*ADANN: Automatic Design of Artificial Neural Networks to forecast time series*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	ISF 2009
Publicación:	Actas de ISF 2009
Lugar:	Hong Kong, China
Fecha:	Julio, 2009

Título:	*ADANN: Automatic Design of Artificial Neural Networks*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	GECCO 2008
Publicación:	Actas de GECCO 2008
Lugar:	Atlanta, EEUU
Fecha:	Julio, 2008

Título:	*Design of Artificial Neural Networks Base on Genetic Algorithms to Forecast Time Series*
Autores:	J. Peralta, G. Gutierrez, A. Sanchis
Congreso:	CAEPIA 2007
Publicación:	Actas de CAEPIA 2007
Lugar:	Salamanca, España
Fecha:	Noviembre, 2007

Título:	*Algoritmos Paralelos para la resolución de Ecuaciones Diferenciales Ordinarias Mediante OpenMP*
Autores:	J. Peralta, E. Arias, V. Hernandez, J. J. Ibañez
Congreso:	XIV Jornadas de Paralelismo
Publicación:	Actas de las XIV Jornadas de Paralelismo
Lugar:	Madrid, España
Fecha:	Junio, 2003

Bibliografía

[1] Rob R. Weitz. Nostradamus a knowledge-based forecasting advisor. *International Journal of Forecasting*, 2(3):273–283, 1986.

[2] Amir D Aczel. *God's equation : Einstein, relativity, and the expanding universe*. MJF Books, New York, 1999.

[3] Paul Goodwin. The holt-winters approach to exponential smoothing: 50 years old and going strong. *Foresight: The International Journal of Applied Forecasting*, (19):30–33, 2010.

[4] Markos Papageorgiou, Apostolos Kotsialos, and Antonios Poulimenos. Long-term sales forecasting using holt-winters and neural network methods. *Journal of Forecasting*, 24(5):353–368, 2005.

[5] George E. P. Box and Gwilym M. Jenkins. *Time series analysis; forecasting and control [by] George E. P. Box and Gwilym M. Jenkins*. Holden-Day San Francisco,, 1970.

[6] S. Makridakis, S.C. Wheelwright, and R.J. Hyndman. *Forecasting methods and applications*. John Wiley & Sons, USA, 3rd edition, 2008.

[7] Ian Nunn and Tony White. The application of antigenic search techniques to time series forecasting. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 353–360, New York, NY, USA, 2005. ACM.

[8] Paulo Cortez. Data mining with neural networks and support vector machines using the r/rminer tool. In *Proceedings of the 10th industrial conference on Advances in data mining: applications and theoretical aspects*, ICDM'10, pages 572–583, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] Sven F. Crone and Nikolaos Kourentzes. Feature selection for time series prediction - a combined filter and wrapper approach for neural networks. *Neurocomput.*, 73:1923–1936, June 2010.

[10] Jose Luis Aznarte M., Jose Manuel Benitez, and Juan Luis Castro. Smooth transition autoregressive models and fuzzy rule-based systems: Functional equivalence and consequences. *Fuzzy Sets and Systems*, 158(24):2734 – 2745, 2007.

[11] M. Stepnicka, J. Peralta, P. Cortez, Lenka Vavrickova, and G. Gutierrez. Forecasting seasonal time series with computational intelligence: contribution of a combination of distinct methods. In *EUSFLAT 2011*, pages 464 – 471, 2011.

[12] N.K. Kasabov and Qun Song. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *Fuzzy Systems, IEEE Transactions on*, 10(2):144 –154, April 2002.

[13] Juan Peralta, German Gutierrez, and Araceli Sanchis. Adann: automatic design of artificial neural networks. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 1863–1870, New York, NY, USA, 2008. ACM.

[14] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[15] Md. Rafiul Hassan, Baikunth Nath, and Michael Kirley. A fusion model of hmm, ann and ga for stock market forecasting. *Expert Syst. Appl.*, 33:171–180, July 2007.

[16] Imran Maqsood, Riaz Khan, and Ajith Abraham. An ensemble of neural networks for weather forecasting. *Neural Comput. Appl.*, 13:112–122, June 2004.

[17] Samy Bengio, Francoise Fessant, and Daniel Collobert. A connectionist system for medium-term horizon time series prediction. In *IN PROC. INTL. WORKSHOP APPLICATION NEURAL NETWORKS TO TELECOMS*, pages 308–315, 1995.

[18] Xin Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:539–567, 1993.

[19] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[20] George Box, Gwilym M. Jenkins, and Gregory Reinsel. *Time Series Analysis: Forecasting & Control (3rd Edition)*. Prentice Hall, 3rd edition, February 1994.

[21] Daniel Peña. *Análisis de series temporales*. Alianza Editorial, Madrid, 2010.

[22] Agustín Maravall and Samuel Bentolila. Una medida de volatilidad en series temporales con una aplicación al control monetario en españa. *Investigaciones Economicas*, 10(1):185–199, January 1986.

[23] T. Czernichow A. Muñoz. Predicción de series temporales no lineales con el modelo narmax. aplicación a la predicción de la demanda de energía eléctrica. *Anales de Mecánica y Electricidad*, 75:46–57, September 1998.

[24] Spencer S. Jones, R. Scott Evans, Todd L. Allen, Alun Thomas, Peter J. Haug, Shari J. Welch, and Gregory L. Snow. A multivariate time series approach to modeling and forecasting demand in the emergency department. *Journal of Biomedical Informatics*, 42(1):123 – 139, 2009.

[25] Armando Aguirre Jaime. *Introducción al tratamiento de series temporales. Aplicación a las Ciencias de la Salud*. S.A. EDICIONES DIAZ DE SANTOS, Madrid, Spain, 1st edition, 1994.

[26] Chi-Jie Lu, Tian-Shyug Lee, and Chih-Chou Chiu. Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, 47(2):115 – 125, 2009.

[27] Athanasios Orphanides and John C. Williams. Learning, expectations formation, and the pitfalls of optimal control monetary policy. *Journal of Monetary Economics*, 55(Supplement 1):S80 – S96, 2008. Contributions to Macroeconomics in Honor of John Taylor - Supplement issue sponsored by the Federal Reserve Bank of Dallas, containing selected papers from an October 12-13, 2007 conference honoring John Taylor.

[28] Juan Camilo Santana. Prediccóin de series temporales con redes neuronales: una aplicación a la inflación colombiana. *REVISTA COLOMBIANA DE ESTADÍSTICA*, 2006.

[29] Niu Lin, Zhao Jian-guo, Du Zhi-gang, and Jin Xiao-ling. Application of time series forecasting algorithm via support vector machines to power system wide-area stability prediction. In *Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES*, pages 1 –6, 2005.

[30] Jesús Jara, María Pilar y Rosel. *Análisis de series temporales: un ejemplo de aplicación en ámbitos psicológicos*. UNIVERSITAT JAUME I, Castellón, Valencia, 1st edition, 2002.

[31] K.W. Hipel and A.I. McLeod. *Time series modelling of water resources and environmental systems*. Developments in water science. Elsevier, 1994.

[32] Kyoung-jae and Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307 – 319, 2003. <ce:title>Support Vector Machines</ce:title>.

[33] Gutiérrez E. Galván I. Strozzi F. Zaldivar, J. and A Tomasin. Forecasting high waters at venice lagoon using chaotic time series analysis and nonlinear neural networks. *Journal of Hydroinformatics*, 2:61–84, 2000.

[34] Ashu Jain and Avadhnam Madhav Kumar. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, 7(2):585 – 592, 2007.

[35] Valls J. M. Luque, C. and P Isasi. Predicción local de series temporales mediante la evolución de regiones de voronoi. In *II Simposio de Inteligenica Computacional (SICO2007)*, pages 397–402, 2007.

[36] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, July 1977.

[37] E. Gutiérrez, F. Taucer, T. De Groeve, D. H. A. Al-Khudhairy, and J. M. Zaldivar. Analysis of worldwide earthquake mortality using multivariate demographic and seismic data. *American Journal of Epidemiology*, 161(12):1151–1158, 15 June 2005.

[38] Aihua Shen, Rencheng Tong, and Yaochen Deng. Application of classification models on credit card fraud detection. In *Service Systems and Service Management, 2007 International Conference on*, pages 1 –4, june 2007.

[39] Md.M. Rahman, P. Bhattacharya, and B.C. Desai. A framework for medical image retrieval using machine learning and statistical similarity matching techniques with relevance feedback. *Information Technology in Biomedicine, IEEE Transactions on*, 11(1):58 –69, jan. 2007.

[40] Beatriz Garcia, Agapito Ledezma, and Araceli Sanchis. Mmrf for proteome annotation applied to human protein disease prediction. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP 2010).*, Lecture Notes in Artificial Intelligence, page In Press, June 2010.

[41] S. Srinivasan and DeLiang Wang. A supervised learning approach to uncertainty decoding for robust speech recognition. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, page I, may 2006.

[42] E. de Groote. Machine learning in go: Supervised learning in move prediction. Master's thesis, University of Twente, March 2005.

[43] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34 –40, aug. 2007.

[44] Kei Keung Hung and Lei Xu. Further study of adaptive supervised learning decision (asld) network in stock market. In *in IEEE-EURASIP Workshop Nonlinear Signal Image Processing, 1999, Paper*, page 154, 1999.

[45] Russell D. Reed and Robert J. Marks. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, Cambridge, MA, USA, 1998.

[46] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2:139–172, September 1987.

[47] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[48] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[49] http://darwin online.org.uk/biography.html. The complete works of darwin online - biography. Último acceso en Julio del 2011.

[50] John H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992.

[51] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press, 1 edition, December 1992.

[52] John H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992.

[53] Jose A. Lozano, Pedro Larrañaga, and Inaki Inza. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer, February 2006.

[54] Shummet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Pittsburgh, PA, USA, 1994.

[55] Heinz Mühlenbein. The equation for response to selection and its use for prediction. *Evol. Comput.*, 5:303–346, September 1997.

[56] Jeremy S. De Bonet, Charles L. Isbell, Jr., and Paul Viola. Mimic: Finding optima by estimating probability densities. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, page 424. The MIT Press, 1996.

[57] Shumeet Baluja and Scott Davies. Combining multiple optimization runs with optimal dependency trees. Technical report, Computer Science Department, Carnegie Mellon University, 1997.

[58] Pedro Larrañaga, Ramon Etxeberria, José A. Lozano, and José M. Peña. Combinatorial optimization by learning and simulation of bayesian networks. In *in Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 343–352. Morgan Kaufmann, 2000.

[59] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[60] R. Sagarna P. Larrañaga C. Cotta, E. Alba. *Adjusting Weights in Artificial Neural Networks using Evolutionary Algorithms. In Estimation of Distribution Algorithms. A new tool for Evolutionary Computation*. 2002.

[61] Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.

[62] P. L. Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mémoires des Savants étrangers présentés á lAcadémie de Saint-Pétersbourg*, 7:539 – 586, 1854.

[63] K.V. Price. Genetic annealing. *Dr. Dobb's Journal*, 10:128 – 132, 1994.

[64] R. de A. Araujo, G.C. Vasconcelos, and T.A.E. Ferreira. Hybrid differential evolutionary system for financial time series forecasting. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4329 –4336, 2007.

[65] H. M. Abdul-Kader. Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting. *International Journal of Computer Science and Network Security*, 9, March 2009.

[66] Upali Wickramasinghe and Xiaodong Li. Choosing leaders for multiobjective pso algorithms using differential evolution. In Xiaodong Li, Michael Kirley, Mengjie Zhang, David Green, Vic Ciesielski, Hussein Abbass, Zbigniew Michalewicz, Tim Hendtlass, Kalyanmoy Deb, Kay Tan, Jurgen Branke, and Yuhui Shi, editors, *Simulated Evolution and Learning*, volume 5361 of *Lecture Notes in Computer Science*, pages 249–258. Springer Berlin / Heidelberg, 2008.

[67] Jarmo Ilonen, Joni-Kristian Kamarainen, and Jouni Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Process. Lett.*, 17:93–105, March 2003.

[68] Jui-Fang Chang, Chi-Ming Kuan, and Yu-Wen Lin. Forecasting exchange rates by genetic algorithms based back propagation network model. *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on*, 0:791–794, 2009.

[69] Yi-Shian Lee and Lee-Ing Tong. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, 24(1):66 – 72, 2011.

[70] George G. Szpiro. Forecasting chaotic time series with genetic algorithms. *Phys. Rev. E*, 55(3):2557–2568, Mar 1997.

[71] P. Grassberger and I. Procaccia. Phys. rev. lett. 50, 448. 1983.

[72] M. B. Porecha, P. K. Panigrahi, J. C. Parikh, C. M. Kishtawal, and Sujit Basu. Forecasting non-stationary financial time series through genetic algorithm. Quantitative finance papers, arXiv.org, 2005.

[73] Chin Teck Chai, Chua Hong Chuek, D.P. Mital, and Tan Thiam Huat. Time series modelling and forecasting using genetic algorithms. In *Knowledge-Based Intelligent Electronic Systems, 1997. KES '97. Proceedings., 1997 First International Conference on*, volume 1, pages 260 –268 vol.1, May 1997.

[74] Paulo Cortez, Miguel Rocha, and José Neves. A meta-genetic algorithm for time series forecasting, 2001.

[75] N. H. Packard. A Genetic Learning Algorithm for the Analysis of Complex Data. *Complex Systems*, 4(5):543–572, 1990.

[76] T. P. Meyer and N. H Packard. Local forecasting of high dimensional chaotic dynamics. 1992.

[77] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press, February 1998.

[78] Cristobal Luque, Jose Maria Valls Ferran, and Pedro Isasi Vinuela. Time series forecasting by means of evolutionary algorithms. *Parallel and Distributed Processing Symposium, International*, 0:246, 2007.

[79] David Quintana, Cristóbal Luque, and Pedro Isasi. Evolutionary rule-based system for ipo underpricing prediction. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 983–989, New York, NY, USA, 2005. ACM.

[80] Tang C. Li C. an Yuan C. Zuo, J. and A long Chen. Time series prediction based on gene expression programming. In *Proceedings of theAdvances in Web-Age Information Management: 5th International Conference, WAIM 2004*, volume 3129, pages 56–64, 2004.

[81] José Maria Valls Ferrán Cristobal Luque del Arco-Calderón, Pedro Isasi Viñuela. *Predicción local mediante algoritmos evolutivos*. PhD thesis, Departamento de Informática. Universidad Carlos III de Madrid, 2009.

[82] Aranildo Rodrigues Lima Junior and Tiago Alessandro Espínola Ferreira. A hybrid method for tuning neural network for time series forecasting. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 531–532, New York, NY, USA, 2008. ACM.

[83] Nicos G. Pavlidis, Dimitris K. Tasoulis, and Michael N. Vrahatis. Time series forecasting methodology for multiple-step-ahead prediction. In *Computational Intelligence'05*, pages 456–461, 2005.

[84] Luc BAUWENS, Arnaud DUFAYS, and Bruno DE BACKER. Estimating and forecasting structural breaks in financial time series. CORE Discussion Papers 2011055, UniversitÃ© catholique de Louvain, Center for Operations Research and Econometrics (CORE), November 2011.

[85] J.E. RUBIN, MICHAEL y SAFDIEH. *Netter neuroanatomia esencial*. MASSON, 2008.

[86] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2:303–314, 1989.

[87] S E Fahlman. *Faster-learning variations on Back-propagation: An empirical study*, volume pp, pages 38–51. Morgan Kaufmann, 1988.

[88] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586 –591 vol.1, 1993.

[89] R. D. Reed. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks (Bradford Book)*. MIT Press, March 1999.

[90] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265 – 278, 1994.

[91] Yu-Chi Bryson, Arthur E. y Ho. *Applied optimal control; optimization, estimation, and control*. Blaisdell Pub. Co, 1969.

[92] Halbert White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[93] B. Irie and S. Miyake. Capabilities of three-layered perceptrons. In *Neural Networks, 1988., IEEE International Conference on*, pages 641 –648 vol.1, July 1988.

[94] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2:359–366, July 1989.

[95] Patricia M. West, Patrick L. Brockett, and Linda L. Golden. A comparative analysis of neural networks and statistical methods for predicting consumer choice. *MARKETING SCIENCE*, 16(4):370–391, 1997.

[96] Gulin Dede and Murat Husnu SazlI. Speech recognition with artificial neural networks. *Digital Signal Processing*, 20(3):763 – 768, 2010.

[97] M. P. Sesmero, J. M. Alonso-Weber, G. Gutiérrez, A. Ledezma, and A. Sanchis. Specialized ensemble of classifiers for traffic sign recognition. In *Proceedings of the 9th international work conference on Artificial neural networks*, IWANN'07, pages 733–740, Berlin, Heidelberg, 2007. Springer-Verlag.

[98] KIvan Kili, Ismail Hakki BoyacI, Hamit Koksel, and Ismail Kusmenoglu. A classification system for beans using computer vision system and artificial neural networks. *Journal of Food Engineering*, 78(3):897 – 904, 2007.

[99] Farhad Gharagheizi, Ali Eslamimanesh, Amir H. Mohammadi, and Dominique Richon. Artificial neural network modeling of solubilities of 21 commonly used industrial solid compounds in supercritical carbon dioxide. *Industrial and Engineering Chemistry Research*, 50(1):221–226, 2011.

[100] Bernard Widrow, David E. Rumelhart, and Michael A. Lehr. Neural networks: applications in industry, business and science. *Commun. ACM*, 37(3):93–105, 1994.

[101] G. Peter Zhang. A neural network ensemble method with jittered training data for time series forecasting. *Information Sciences*, 177(23):5329 – 5346, 2007.

[102] Ramesh Sharda. Neural networks for the ms/or analyst: An application bibliography. *INTERFACES*, 24(2):116–130, 1994.

[103] Lars-Erik Oller. Modelling nonlinear economic relationships : C.w.j. granger and t. terasvirta, 1993, (oxford university press, new york,) 187 pp., hardback $30, isbn0-19-877319-6; paperback$14.95, isbn 0-19-877320-. *International Journal of Forecasting*, 10(1):169–171, June 1994.

[104] Anderson A.P. Granger, C.W.J. *An Introduction to Bilinear Time Series Models*. Vandenhoeck und Ruprecht, 1978.

[105] H. Tong and K. S. Lim. Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(3):pp. 245–292, 1980.

[106] Robert F Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007, July 1982.

[107] M.J.C Hu. *Application of the adaline system to weather forecasting, Master Thesis, Technical Report 6775-1*. PhD thesis, Stanford Electronic Laboratories, Stanford, CA, June. 1964., 1964.

[108] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning representations by back-propagating errors*, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[109] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.

[110] Farber R. Lapedes, A. *Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LA-UR-87-2662*. PhD thesis, Los Alamos National Laboratory, Los Alamos, NM., 1987.

[111] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart. Predicting Sunspots and Exchange Rates with Connectionist Networks. In M. Casdagli and S. Eubank, editors, *Nonlinear modeling and forecasting*, pages 395–432. Addison-Wesley, 1992.

[112] M. Cottrell, B. Girard, Y. Girard, M. Mangeas, and C. Muller. Neural modeling for time series: A statistical stepwise method for weight elimination. *Neural Networks, IEEE Transactions on*, 6(6):1355 –1364, November 1995.

[113] Zaiyong Tang, Chrys de Almeida, and Paul A. Fishwick. Time series forecasting using neural networks vs. Box- Jenkins methodology. *SIMULATION*, 57(5):303–310, 1991.

[114] Ramesh Sharda and Rajendra B. Patil. Connectionist approach to time series prediction: an empirical test. *Journal of Intelligent Manufacturing*, 3:317–323, 1992.

[115] Zaiyong Tang and Paul A. Fishwick. Feed-forward neural nets as models for time series forecasting. *ORSA Journal of Computing*, 5:374–385, 1993.

[116] http://www.neural-forecasting competition.com. Forecasting competition for neural networks & computational intelligence. Último acceso en Marzo del 2011.

[117] Feng Xue and Weizhong Yan. Parametric model-based anomaly detection for locomotive subsystems. In *IJCNN*, pages 3074–3079, 2007.

[118] Donald F. Specht. The general regression neural network-rediscovered. *Neural Netw.*, 6:1033–1034, July 1993.

[119] Darrell Whitley and Thomas Hanson. Optimizing neural networks using faster, more accurate genetic search. In *Proceedings of the third international conference on Genetic algorithms*, pages 391–396, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[120] David J. Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, pages 762–767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[121] Thomas P. Caudell and Charles P. Dolan. Parametric connectivity: Training of constrained networks using genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 370–374, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[122] D. B. Fogel, L. J. Fogel, and V. W. Porto. Evolving neural networks. *Biol. Cybern.*, 63:487–493, September 1990.

[123] J. David Schaffer, Richard A. Caruana, and Larry J. Eshelman. *Using genetic search to exploit the emergent behavior of neural networks*, pages 244–248. MIT Press, Cambridge, MA, USA, 1991.

[124] Richard K. Belew, John Mcinerney, and Nicol N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In *In*, pages 511–547. Addison-Wesley, 1990.

[125] J. W. L. and R. F. Port. A stochastic learning algorithm for neural networks. Technical report, Dept. of Linguistics and Computer Science, Indiana Univ.,. Bloomington, 1988.

[126] Richard S. Rosenberg. Simulation of genetic populations with biochemical properties: Ii. selection of crossover probabilities. *Mathematical Biosciences*, 8(1-2):1 – 37, 1970.

[127] J. David Schaffer and Amy Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 36–40, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[128] P. Barlett and T. Downs. Training a neural network with genetic algorithm. Technical report, Technical Report. Univ. of Queensland, 1990.

[129] Hiroaki Kitano. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 2*, AAAI'90, pages 789–795. AAAI Press, 1990.

[130] Md. Islam and Xin Yao. Evolving artificial neural network ensembles. In John Fulcher and L. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 851–880. Springer Berlin / Heidelberg, 2008.

[131] Abhinav Saxena and Ashraf Saad. Evolving an artificial neural network classifier for condition monitoring of rotating mechanical systems. *Applied Soft Computing*, 7(1):441 – 454, 2007.

[132] Paulo Chaves and Fi-John Chang. Intelligent reservoir operation system based on evolving artificial neural networks. *Advances in Water Resources*, 31(6):926 – 936, 2008.

[133] Scott Fahlman Christian and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufmann, 1990.

[134] Marcus Frean. The upstart algorithm: a method for constructing and training feedforward neural networks. *Neural Comput.*, 2:198–209, April 1990.

[135] Jocelyn Sietsma and Robert J. F. Dow. Creating artificial neural networks that generalize. *Neural Netw.*, 4:67–79, January 1991.

[136] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing neural networks using genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 379–384, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[137] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[138] Steven Alex Harp, Tariq Samad, and Aloke Guha. Towards the genetic synthesis of neural network. In *Proceedings of the third international conference on Genetic algorithms*, pages 360–369, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[139] A. Siddiqi and S. M. Lucas. A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *1998 IEEE International Conference on Evolutionary Computation*, pages 392–397. IEEE Press, 1998.

[140] José Mira and José R. Álvarez, editors. *Studying the Capacity of Grammatical Encoding to Generate FNN Architectures*, volume 2686 of *Lecture Notes in Computer Science*. Springer, 2003.

[141] T. Ash. Dynamic node creation in backpropagation networks. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, page 623 vol.2, June 1989.

[142] F. Gruau. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pages 55 –74, June 1992.

[143] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *Neural Networks, IEEE Transactions on*, 8(3):694 –713, May 1997.

[144] V Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1994.

[145] D Whitley, T Starkweather, and C Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347 – 361, 1990.

[146] Stewart W. Wilson. Perception redux: Emergence of structure. *Physica D: Nonlinear Phenomena*, 42(1-3):249 – 256, 1990.

[147] Daniel Rivero, Julian Dorado, Juan Rabuñal, and Alejandro Pazos. Generation and simplification of artificial neural networks by means of genetic programming. *Neurocomput.*, 73:3200–3223, October 2010.

[148] Marco Castellani and Hefin Rowlands. Evolutionary artificial neural network design and training for wood veneer classification. *Eng. Appl. Artif. Intell.*, 22:732–741, June 2009.

[149] Steven A. Harp, Tariq Samad, and Aloke Guha. Advances in neural information processing systems 2. chapter Designing application-specific neural networks using the genetic algorithm, pages 447–454. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[150] N. Dodd, D. Macfarlane, and C. Marland. Optimisation of artificial neural network structure using genetic techniques on multiple transputers. In *Proceedings of the world transputer user group (WOTUG) conference on Transputing '91*, pages 687–700, Amsterdam, The Netherlands, The Netherlands, 1991. IOS Press.

[151] P. Hancock and Leslie Smith. Gannet: Genetic design of a neural net for face recognition. In Hans-Paul Schwefel and Reinhard MAnner, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 292–296. Springer Berlin / Heidelberg, 1991.

[152] E. Mjolsness, D. H. Sharp, and B. K. Alpert. Scaling, machine learning, and genetic neural nets. *Adv. Appl. Math.*, 10:137–163, June 1989.

[153] S. Bornholdt and D. Graudenz. General asymmetric neural networks and structure design by genetic algorithms: a learning rule for temporal patterns. In *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, pages 595 –600 vol.2, October 1993.

[154] P. Isasi J.M. Molina Gutiérrez, A. Sanchis and I. M. Galván. Non-direct encoding method based on cellular automata to design neural network architectures. *Computer and Informatics*, 24(3), 2005.

[155] John W. L. Merrill and Robert F. Port. Fractally configured neural networks. *Neural Netw.*, 4:53–60, January 1991.

[156] B. West. The fractal structure of the human lung. 1988.

[157] G. Mani. Learning by gradient descent in function space. In *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pages 242 –247, November 1990.

[158] David Lovell and Ah Chung Tsoi. The performance of the neocognitron with various s-cell and c-cell transfer functions. Technical report, University of Queensland, Queensland 4072, Australia, 1992.

[159] Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. Technical report, Amherst, MA, USA, 1987.

[160] David J. Chalmers. The evolution of learning: an experiment in genetic connectionism. In *Proceedings of the 1990 Connectionist Models Summer School*, pages 81–90. Morgan Kaufmann, 1990.

[161] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. In *in Conference on Optimality in Biological and Artificial Networks*, 1991.

[162] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule, 1997.

[163] Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, new edition edition, June 1949.

[164] Peter J. B. Hancock, Leslie S. Smith, and William A. Phillips. A biologically supported error-correcting learning rule. *Neural Computation*, 3(2):201–212, 1991.

[165] S; Singer W Artola, A; Brocher. Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, 347:69–72, 1990.

[166] Miguel Rocha, Paulo Cortez, and José Neves. Evolution of neural networks for classification and regression. *Neurocomput.*, 70:2809–2816, October 2007.

[167] Somesh Selvaratnam and Michael Kirley. Predicting Stock Market Time Series Using Evolutionary Artificial Neural Networks with Hurst Exponent Input Windows. pages 617–626. 2006.

[168] Paulo Cortez, Miguel Rocha, and José Neves. Evolving time series forecasting neural network models, 2001.

[169] V. M. Rivas, J. J. Merelo, P. A. Castillo, M. G. Arenas, and J. G. Castellano. Evolving rbf neural networks for time-series forecasting with evrbf. *Inf. Sci. Inf. Comput. Sci.*, 165:207–220, October 2004.

[170] Tiago A. E. Ferreira, Germano C. Vasconcelos, and Paulo J. L. Adeodato. A new evolutionary method for time series forecasting. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 2221–2222, New York, NY, USA, 2005. ACM.

[171] James Surowiecki. *The Wisdom of Crowds*. Anchor, August 2005.

[172] Xin Yao, Senior Member, Yong Liu, and Student Member. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 28:417–425, 1998.

[173] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[174] P. Cortez, M. Rio, M. Rocha, and P. Sousa. Internet Traffic Forecasting using Neural Networks. In *Proceedings of The 2006 International Joint Conference on Neural Networks (IJCNN 2006)*, pages 4942–4949, Vancouver, Canada, IEEE Computer Society, July 2002.

[175] Anders Krogh and Peter Sollich. Statistical mechanics of ensemble learning. *Phys. Rev. E*, 55(1):811–825, Jan 1997.

[176] Thomas G. Dietterich. Ensemble methods in machine learning. In *INTERNATIONAL WORKSHOP ON MULTIPLE CLASSIFIER SYSTEMS*, pages 1–15. Springer-Verlag, 2000.

[177] M. Sesmero, J. Alonso-Weber, G. GutiÃ©rrez, A. Ledezma, and A. Sanchis. A new artificial neural network ensemble based on feature selection and class recoding. *Neural Computing and Applications*, pages 1–13. 10.1007/s00521-010-0458-5.

[178] David W. Opitz and Jude W. Shavlik. Actively searching for an effective neural-network ensemble. *CONNECTION SCIENCE*, 8(3):337–353, 1996.

[179] Amanda J. C. Sharkey. On combining artificial neural nets. *Connection Science*, 8:299–313, 1996.

[180] Amanda J.C. Sharkey, A J. C. Sharkey, and Noel E. Sharkey. Combining diverse neural nets. *The Knowledge Engineering Review*, 12:231–247, 1997.

[181] Yong Liu, Xin Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *Evolutionary Computation, IEEE Transactions on*, 4(4):380 – 387, nov 2000.

[182] Hany Sallam, Carlo S. Regazzoni, Ihab Talkhan, and Amir Atiya. Evolving neural networks ensembles nnes, 2008.

[183] Seymour Geisser. *Predictive inference. An introduction.* London: Chapman and Hall, 1993.

[184] Frederick Mosteller. A sample slippage test for an extreme population. In Stephen Fienberg and David Hoaglin, editors, *Selected Papers of Frederick Mosteller*, Springer Series in Statistics, pages 101–109. Springer New York, 2006.

[185] Simo Sarkka, Aki Vehtari, and Jouko Lampinen. Cats benchmark time series prediction by kalman smoother with cross-validated noise density. *Neurocomputing*, 70(13-15):2331 – 2341, 2007. Selected papers from the 3rd International Conference on Development and Learning (ICDL 2004), 3rd International Conference on Development and Learning; Time series prediction competition: the CATS benchmark.

[186] Benjamin W. Wah and Minglun Qian. Time-series predictions using constrained formulations for neural-network training and cross validation. In *In Proc. Int'l Conf. on Intelligent Information Processing, 16th IFIP World Computer Congress*, pages 220–226. Kluwer Academic Press, 2000.

[187] Lawrence Davis and Melanie Mitchell. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[188] Rainer Storn and Kenneth Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11:341–359, December 1997.

[189] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[190] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE, Piscataway, NJ, USA, 1995.

[191] Andreas Zell, Niels Mache, Tilman Sommer, and Thomas Korb. Recent developments of the SNNS neural network simulator. volume 1469, pages 708–718. SPIE, 1991.

[192] Genetic Algorithms Marek Obitko. http://www.obitko.com/tutorials/genetic-algorithms/. Último acceso en Mayo del 2011.

[193] A. Tsukahara and A. Kanasugi. Genetic algorithm that can dynamically change number of individuals and accuracy. In *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*, pages 785 –789, oct. 2007.

[194] Kenneth V. Price. *An introduction to differential evolution*, pages 79–108. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[195] Ricardo Landa Becerra and Carlos A. Coello Coello. *Use of domain information to improve the performance of an evolutionary algorithm*. PhD thesis, Computer Science Department, National Polytechnic Institute of Mexico, 2007.

[196] Scott E. Fahlman. The recurrent cascade-correlation architecture. In *Proceedings of the 1990 conference on Advances in neural information processing systems 3*, NIPS-3, pages 190–196, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[197] Barry L. Kalman and Stan C. Kwasny. Trainrec: A system for training feedforward & simple recurrent networks efficiently and correctly, 1993.

[198] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[199] N. Moseley. Modeling Economic Time Series Using a Focused Time Lagged FeedForward Neural Network. *Proceedings of Student Research Day, CSIS, Pace University*, May 2003.

[200] P. Cortez, M. Rocha, and J. Neves. *Time Series Forecasting by Evolutionary Neural Networks*, chapter III, pages 47–70. Idea Group Publishing, USA, 2006.

[201] A. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1989.

[202] http://www.forecastpro.com/. Forecast pro is a registered trademark of business forecast systems, inc. copyright 2011, business forecast systems, inc. Último acceso en Marzo del 2011.

[203] http://robjhyndman.com/TSDL/. Time series data library. Last access 2011.

[204] Harold F. Dorn. Pitfalls in population forecasts and projections. *Journal of the American Statistical Association*, 45(251):pp. 311–334, 1950.

[205] Stanley K. Smith and Terry Sincich. The relationship between the length of the base period and population forecast errors. *Journal of the American Statistical Association*, 85(410):pp. 367–375, 1990.

[206] Steven P. Schnaars. Situational factors affecting forecast accuracy. *Journal of Marketing Research*, 21(3), 1984.

[207] J.S. Armstrong. *Long-range forecasting*. Wiley New York ETC., 1985.

[208] R.J. Hyndman and A.B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[209] R.R. Andrawis and A.F. Atiya. A new Bayesian formulation for Holt's exponential smoothing. *Journal of Forecasting*, 28(3):218–234, 2009.

[210] Joan Fisher Box. Guinness, gosset, fisher, and small samples. *Statistical Science*, 2(1):pp. 45–52, 1987.

[211] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):pp. 591–611, 1965.

[212] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[213] http://www.autobox.com/. Herramienta profesional de predicción de series temporales. Último acceso en Marzo del 2011.

[214] http://cedi2005.ugr.es. Congreso espaÑol de informática (cedi). Último acceso en Marzo del 2011.

[215] C. James Taylor, Diego J. Pedregal, Peter C. Young, and Wlodek Tych. Environmental time series analysis and forecasting with the captain toolbox. *Environ. Model. Softw.*, 22:797–814, June 2007.

[216] P. Cortez. Sensitivity Analysis for Time Lag Selection to Forecast Seasonal Time Series using Neural Networks and Support Vector Machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, pages 3694–3701, Barcelona, Spain, July 2010. IEEE.

[217] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[218] Vladimir Vapnik, Steven E. Golowich, and Alex Smola. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press, 1996.

[219] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[220] A. I. Belousov, S. A. Verzakov, and J. von Frese. A flexible classification approach with optimal generalisation performance: support vector machines. *Chemometrics and Intelligent Laboratory Systems*, 64(1):15 – 25, 2002.

[221] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, August 2004.

[222] Sayan Mukherjee, Edgar Osuna, and Federico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *IEEE Workshop on Neural Networks for Signal Processing VII*, pages 511–519. IEEE Press, 1997.

[223] Klaus-Robert Müller, Alexander J. Smola, Gunnar Rätsch, Bernhard Schökopf, Jens Kohlmorgen, and Vladimir Vapnik. *Using support vector machines for time series prediction*, pages 243–253. MIT Press, Cambridge, MA, USA, 1999.

[224] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, USA, 2002.

[225] Martin Stepnicka, Antonín Dvorák, Viktor Pavliska, and Lenka Vavrícková. A linguistic approach to time series modeling with help of the f-transform. *Fuzzy Sets and Systems*, In Press, Corrected Proof, 2011.

[226] Vilem Novak. Analysis of seasonal time series using fuzzy approach. *International Journal of General Systems*, 39:305–328(24), April 2010.

[227] Ricardo Landa Becerra and Carlos A. Coello Coello. Use of domain information to improve the performance of an evolutionary algorithm. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, GECCO '05, pages 362–365, New York, NY, USA, 2005. ACM.

[228] Philip Hingston. The 2k botprize. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 1–1, Piscataway, NJ, USA, 2009. IEEE Press.

[229] Jorge Muñoz, German Gutierrez, and Araceli Sanchis. A human-like TORCS controller for the Simulated Car Racing Championship. In *Proceedings 2010 IEEE Conference on Computational Intelligence and Games*, pages 473–480, August 2010.

Apéndice A

Configuración de la Experimentación

En este apéndice se detallan los diferentes parámetros de los que disponen las llamadas al sistema ADANN. Estos parámetros deben ser seleccionados por el usuario antes de poder llevar a cabo la experimentación correspondiente. Para realizar el lanzamiento de un experimento en ADANN, la llamada al sistema es:

ADANN -P fichero_parametros.txt -t serie_temporal.txt -n 0

Donde ADANN corresponde al fichero de la aplicación compilado y:

- -P serie_temporal.txt es el archivo que contiene la serie temporal con cada uno de sus elementos conocidos.
- -t fichero_parametros.txt corresponde al archivo donde están incluidos los parámetros que hay que modificar para cada experimentación como veremos más adelante.
- -n 0 es un parámetro que indica al sistema si la primera generación es obtenida desde cero de forma aleatoria por el sistema o por el contrario es leída desde un archivo (-n 1).

A.1. Parámetros del Sistema

A continuación pasamos a describir uno a uno los distintos parámetros del sistema y como sus posibles valores afectan a las experimentaciones que se llevan a cabo:

- **PRCT_INC**: Como ya se indicó en la sección 4.2.1, este parámetro es usado con un valor de 10 % cuando la serie temporal a predecir es estacionaria (i.e. no tiene tendencia) o bien 50 % cuando no es estacionaria (i.e. sí tiene tendencia), habiendo considerado estos valores mediante experimentación previa.
- **CROSSVALIDATION**: Indica el número de subconjuntos en que se dividen los datos para llevar a cabo validación cruzada. Este parámetro puede tomar valores comprendidos entre 1 y 8. Si este parámetro toma el valor 1, indica que no se lleva a cabo validación cruzada.
- **PRCT_TR**: Su valor indica el tanto por ciento que se selecciona del conjunto total de patrones para generar el conjunto de entrenamiento.
- **PRCT_VAL**: Su valor indica el tanto por ciento que se selecciona del conjunto total de patrones para generar el conjunto de validación.
- **PRCT_VAL_II**: Su valor indica el tanto por ciento que se selecciona del conjunto total de patrones para generar el conjunto de validaciónII.
- **PRCT_FITNESS_VALID**: Como ya indicamos en la sección 3.3, este parámetro sirve para establecer el valor α, o lo que es igual, el peso asociado a la función de fitness correspondiente al conjunto de validación.
- **PRCT_FITNESS_VALIDII**: Semejante a su predecesor, este parámetro sirve para establecer el valor β, siendo éste el peso asociado a la función de fitness correspondiente al conjunto de validaciónII.
- **EVOLUTIONARY_ALGORITHM**: Indica con un 1 que el algoritmo evolutivo utilizado para el experimento es un algoritmo genético, con un 2 que se hace uso del algoritmo de estimación de distribución y con un 3 que el algoritmo seleccionado para realizar la búsqueda global es el de evolución diferencial.
- **CR**: Como se indicó en la sección 3.6.2, este parámetro representa la probabilidad de sobrecruzamiento si el algoritmo evolutivo seleccionado es el de evolución diferencial.
- **F**: También explicado en la sección 3.6.2, este parámetro es utilizado sólo en el algoritmo de evolución diferencial como factor de mutación.
- **POPULATION_SIZE**: Tamaño de la población del proceso evolutivo.
- **MAX_NUM_GENERATIONS**: Número de generaciones durante las que se llevará a cabo el proceso evolutivo.

- **PERCENTAGE_ELITE_POPULATION**: Parámetro de elitismo o tanto por ciento de individuos de la generación *i* que pasan directamente a la generación *i+1*.
- **TRAINING_CYCLES**: Número de ciclos de entrenamiento de las redes de neuronas artificiales.
- **NUMBER_VALID**: Indica cuantas comprobaciones de validación se llevarán a cabo durante los ciclos de entrnamiento previamente seleccionados. Si se hubieran elegido 5000 ciclos de entrenamiento y 500 de validación, esto significará que cada 10 ciclos de entrenamiento se llevará a cabo una comprobación de validación de la red para comprobar el error de ésta.
- **MAX_ENTRADAS**: Este parámetro sirve para delimitar el número máximo de nodos de entrada de la RNA como se indicó en la sección 3.6.1. Supongamos que el número de elementos de la serie temporal es 125 y este parámetro tiene un valor de 0,3, entonces el número máximo de nodos de entrada que podrán tener las redes de neuronas artificiales es de 125*0,3 o lo que es igual, 38.
- **SHUFFLE**: Indica con un 1 si se lleva a cabo *"Shuffle"* o con un 0 si no es así.