Predicción de Series Temporales con Redes Neuronales

José Angel de Bustos Pérez

Investigación Operativa

Curso $1.999 - 2.000$

Índice General

Índice de Figuras

Capítulo 1

Introducción

En este primer capítulo vamos a dar una introducción sobre:

- Predicción de "*Series Temporales*".
- "*Redes Neuronales*".

1.1 Series Temporales

Las "*Series Temporales*" también reciben el nombre de "*Series Cronológicas*" o "*Series Históricas*". Se basan en el estudio del comportamiento de una determinada variable, observaciones, con el fin de poder predecir el comportamiento de dicha variable a lo largo del tiempo.

1.1.1 ¿Para que sirven?

Intentaremos crear un modelo matemático para predecir la evolución de una variable a lo largo del tiempo. Esta variable puede ser de cualquier tipo:

- **Económica**: *índice de precios al consumo*, *demanda de electricidad*, *producto nacional bruto*, *ventas de un determinado tipo*, ...
- **Física**: *velocidad del viento*, *temperatura en un proceso*, *concentración en la atmósfera de un contaminante*, ...
- **Social**: *número de nacimientos*, *número de defunciones*, *votos a un partido político*, ...

1.1.2 ¿Que se necesita?

Necesitaremos disponer de datos sobre la evolución de la variable que nos interesa a intervalos regulares de tiempo (horas, días, meses, años, ...). Utilizando estos datos intentaremos "adivinar" el comportamiento de dicha variable en el futuro, suponiendo que el comportamiento de la variable es uniforme a lo largo del tiempo, es decir que su comportamiento en el futuro[1] es similar al comportamiento que tuvo dicha variable en el pasado. Es decir que su comportamiento no experimenta cambios bruscos entre dos espacios de tiempo proximos.

Este tipo de analisis recibe el nombre de *análisis univariante*, el cual es util para predicciones a corto plazo. Si necesitaramos hacer predicciones a medio o largo plazo necesitariamos tener en cuenta una serie de variables relacionadas con la variable que queremos estudiar. Este tipo de análisis recibe el nombre de *análisis multivariante*.

1.1.3 Modelo matemático

El modelo matemático para una *serie temporal* es el concepto de "*proceso estocástico*". Supondremos que el valor observado de la serie en un instante t es una extracción al azar de una variable aleatoria definida en dicho instante t.

Una serie de n datos será una muestra de un vector de n variables aleatorias ordenadas en el tiempo $(z_1, \ldots, z_n)$. Se denomina *proceso estocástico* al conjunto de estas variables $\{z_t\}_{t=1}^{n}$, y la serie observada se considera una realización o trayectoria del proceso.

Resumiendo, una "*serie temporal*" es una sucesión de valores observados de una variable en diferentes instantes de tiempo, en la que las observaciones aparecen ordenadas cronológicamente. Los instantes de tiempo en los que se toman las mediciones de la variable suelen ser intervalos regulares de tiempo.

Hay "*series temporales*" que también reciben el nombre de "*series aleatorias*" ya que son práctiamente impredecibles, como por ejemplo pueden ser:

- Los números premiados en la Lotería Nacional.
- El número resultante del lanzamiento de un dado equilibrado.

[1] O al menos en un futuro no muy lejano.

Sin embargo también existen "*series temporales*" cuyo comportamiento es tan regular que se pueden hacer predicciones muy precisas:

- La posición de los astros.
- El horario de las mareas.

En un lugar intermedio, entre estos dos tipos de "*series temporales*", tenemos otro tipo de series que en su comportamiento tienen dos componentes, una componente "*regular*" y otra "*irregular*". Un ejemplo de estas series pueden ser las "*series temporales económicas*"[2].

1.1.4 Procesos estocásticos

Un "*proceso estocástico*", $\{X_t\}$, es una colección de variables aleatorias, X_t, ordenadas según un parámetro discreto, t, que para nosotros será el tiempo.

Como hemos observado antes los modelos estocásticos de series temporales conciben una serie temporal dada como una colección de observaciones muestrales, cada una de ellas correspondiente a una variable del proceso.

1.1.5 Definiciones sobre procesos estocásticos

Definición 1.1 (Función de medias) *Llamaremos "función de medias" del proceso a una función, dependiente del tiempo, que proporciona las medias de las distribuciones marginales z_t para cada instante t:*

$$\mu_t = E[z_t]$$

Si todas la variables tienen la misma media, entonces la "*función de medias*" es constante y diremos que el proceso es *estable* en la media.

Definición 1.2 (Función de varianzas) *Llamaremos "función de varianzas" del proceso a una función, dependiente del tiempo, que proporciona las varianzas en cada instante t:*

$$\sigma_t^2 = Var\ (z_t)$$

Si esta función es constante en el tiempo diremos que el proceso es *estable* en la varianza.

[2]Si no todas, al menos la mayoría.

Definición 1.3 (Función de autocovarianzas) *Llamaremos "función de autocorrelación" del proceso a la función que describe las covarianzas en dos instantes cualesquiera:*

$$Cov\ (t, t+j) = Cov\ (z_t, z_{t+j}) = E\ [(z_t - \mu_t)(z_{t+j} - \mu_{t+j})]$$

Definición 1.4 (Función de autocorrelación) *Llamaremos "función de autocorrelación" del proceso a la estandarización de la función de covarianzas:*

$$\varrho_{(t,t+j)} = \frac{Cov\ (t, t+j)}{\sigma_t, \sigma_{t+j}}$$

Estas dos últimas funciones dependen de dos parámetros, t y j, donde:

- t es el instante inicial.
- j es el intervalo entre observaciones.

En muchos fenómenos dinámicos se observa una condición de estabilidad en la cual la dependencia entre dos observaciones sólo depende del intervalo entre ellas y no del origen considerado, es decir:

$$Cov\ (t_1, t_{1+k}) = Cov\ (t_2, t_{2+k}) = \gamma_k \qquad \forall k$$

es decir la relación entre z_t y z_{t+j} es siempre igual a la relación entre z_t y z_{t-j}.

1.1.6 Procesos estacionarios

Diremos que un proceso estocástico (serie temporal) es estacionario en "sentido débil" si existen y son estables la media, la varianza y las covarianzas, es decir, si para todo t:

1. $\mu_t = \mu = \text{cte}$
2. $\sigma_t^2 = \sigma^2 = \text{cte}$
3. $Cov\ (t, t+k) = Cov\ (t, t-k) = \gamma_k\ \forall k$

Teniendo en cuenta que $\gamma_0 = \sigma^2$ la "función de autocorrelación" para un proceso estacionario es:

$$\varrho_{(t,t+k)} = \varrho_k = \frac{\gamma_k}{\gamma_0}$$

además se verifica que $\varrho_{-k} = \varrho_k$.

Llamaremos "*función de autocorrelación simple*"(fas) o "*correlograma*" a la representación de los coeficientes de autocorrelación en función del retardo.

Si la dependencia entre observaciones tiende a cero al aumentar el retardo diremos que el proceso es "*ergódico*". En lo sucesivo supondremos que los procesos estacionarios siempre son "*ergódicos*".

1.1.7 Procesos estacionales

Que un proceso sea estacional implica que la serie temporal que lo representa sigue una pauta regular de comportamiento periódico.

Si la serie es mensual, por ejemplo, diremos que existirá estacionalidad si los eneros tienden a ser similares en distintos años, y lo mismo con el resto de meses. La estacionalidad de una serie la hace no estacionaria, ya que el valor medio μ variará de unos meses a otros.

1.1.8 Procesos de ruido blanco

Un proceso estacionario muy importante es el definido por:

1. $E\left[z_t\right] = 0$
2. $Var\ (z_t) = \sigma^2$
3. $Cov\ (z_t, z_{t-k}) = 0\ \forall\ k$

este proceso recibe el nombre de "*proceso de ruido*". Si todas las variables de un proceso de este tipo tienen una distribución normal entonces diremos que es un proceso de "*ruido blanco*".

1.2 Redes Neuronales

1.2.1 ¿Que son?

Las "*Redes Neuronales*" surgieron como consecuencia del intento de modelizar el funcionamiento del cerebro humano, y en particular su capacidad de aprendizaje, para poder dotar de "inteligencia" a los ordenadores.

Estas redes son modelos simplificados de las redes de neuronas que forman el cerebro y también reciben el nombre de "**RNA**"[3] y al igual que el modelo biológico intentan "*aprender*" de los datos de los que le son suministrados.

1.2.2 ¿Para que sirven?

Las **RNA** se utilizan para resolver aquellos problemas en los que su complejidad hace casi imposible su resolución mediante las técnicas habituales de programación, ya que el algoritmo utilizado para su resolución es altamente complejo y dificilmente programable mediante los sistemas de computación secuenciales.

Un ejemplo de estos problemas son los "*Sistemas de Reconocimiento Optico de Caracteres*"[4]. Estos sistemas tratan, basicamente, de que un ordenador reconozca las letras y números del abecedario a partir de una imagen digital(reconocimiento de patrones).

1.2.3 Historia

Las primeras investigaciones en este campo empezaron en los años 40 con los trabajos de *McCulloh* y *Pitts*. Posteriormente *Donald O. Hebb* estableció una conexión entre la psicología y físiologia en la organización del comportamiento, proponiendo un método de aprendizaje que se sigue aplicando en muchas redes neuronales actualmente como una de las reglas más potentes.

[3] Redes Neuronales Artificiales.
[4] O.C.R.

Posteriormente *Bernard Widrow* y *Marcian Hoff* desarrollaron un algoritmo de red adaptativa basado en un modelo simple de neurona al que llamaron **Adaline** y *Frank Rosenblantt* desarrollo el **Perceptron** que era capaz de clasificar patrones contaminados con ruido a la entrada[5] y fue inicialmente aplicado a "O.C.R.".

En 1.969 hay un declive en la investigación sobre **RNA** a partir de la publicación de un trabajo de *Minsky* y *Papert* que demostraba que el **Perceptron** no era capaz de resolver problemas no lineales simples, como la función lógica *XOR*[6].

En los años 80 el campo de la computación neuronal resurge como consecuencia de los trabajos de:

- **Kohonen**, que aportó trabajos sobre memorias adaptativas y aprendizaje competitivo, quedando como recompensa la llamada *red de Kohonen*, en honor al investigador.
- **Grossberg**, que trabajó sobre diseño y construcción de modelos neuronales, introduciendo la función de transferencia de tipo *sigmoide* con la que era posible establecer valores de activación reales y acotados. A partir de estos postulados se desarrolló su *Teoría de la Resonancia Adaptativa (ART)*.
- **Hopfield**, que desarrolló un modelo que consistía en un sistema de elementos totalmente interconectados y que buscaba la situación de mínima energía.
- **Rumelhart**, que desarrolló el algoritmo de aprendizaje "*Backpropagation*" o propagación del error hacia atrás.

[5] La información original se deteriora antes de ser procesada.

[6] Una red de perceptrones sí era capaz de resolver el problema *XOR* pero no existian algoritmos que pudieran entrenar todos los casos.

1.2.4 Características de las RNA

Las **RNA** presentan una serie de características que son:

Paralelismo: debido a su diseño son entidades altamente paralelas, ya que cada elemento de la red puede funcionar independientemente del resto (dentro de una misma capa).

Adaptabilidad: las **RNA** son capaces de autoorganizarse y aprender. Las redes no son programadas como los ordenadores tradicionales, sino que son entrenadas a través de la repetida presentación de ejemplos.

Aprendizaje adaptativo: consigue el conocimiento en base a una etapa característica que es el aprendizaje, que se basa en el entrenamiento.

Autoorganización: En base a ese entrenamiento la red va a crearse su propia representación de todo el dominio de la información que se le puede dar. Es decir, aunque no se la entrene para todas las posibles entradas que le pueden llegar es capaz de responder a todas ellas de una manera coherente.

Reconocimiento de patrones: las **RNA** tienen la "habilidad" de clasificar patrones. Dado un patrón de entrada encuentra el patrón de salida asociado.

Reconstrucción de patrones: la red toma un patrón de entrada incompleto[7] y es capaz de insertar la información perdida en el patrón recuperando el mejor patrón asociado con la entrada.

Tolerancia a fallos: la destrucción o eliminación de elementos de proceso de una red no causará que la red falle e forma global. Dado que la información en una red es distribuida, pequeñas porciones de información pueden perderse sin afectar seriamente al funcionamiento de la red.

Ejemplo 1.1 (RNA aplicada al O.C.R.)
*Podemos emplear una **RNA** para el reconocimiento de patrones. Supongamos que utilizamos una matriz rectangular para la representación de las diferentes letras del abecedario. La matriz tendrá dimensión* 7×5.

[7] Del que se ha perdido información.

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1

Figura 1.1: Ideal de la letra A

1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1

Figura 1.2: Letra A borrosa

*Podemos entrenar una **RNA** para que reconozca la entrada mostrada en la figura* 1.1 *como la letra* A*. Pero por algún motivo puede ocurrir que se suministre la figura* 1.2 *como letra* A*, puede ocurrir un error en la recepción de los datos, o simplemente una persona diferente a nosotros puede tener "otro ideal de letra* A*"*[8]*. Mediante el entrenamiento podemos lograr que una **RNA** reconozca el "patrón" de la letra* A *en la entrada mostrada en la figura* 1.2 *y lo asocie a la letra* A*.*

1.2.5 Elementos de una RNA

Una **RNA** consta de una serie de elementos interconectados entre sí a los que denominaremos "EP", los cuales son elementos simples de procesado, neuronas. Las conexiones entre "EP's" reciben el nombre de "*sinapsis*", las cuales irán ponderadas con unos determinados valores llamados "*pesos sinápticos*" que varíaran a medida que la red aprende, por lo cual necesitaremos de un algoritmo[9] que permita saber cómo y cuánto deben variar estos.

[8] Cada persona tiene un tipo de escritura propio.

[9] Algoritmo de aprendizaje de la red.

1.2.6 Elementos de proceso o EP's

Los "EP's" son los elementos individuales de una **RNA** y los podemos dividir en tres tipos:

- **Entrada:** reciben datos del "mundo exterior" para procesarlos.
- **Salida:** emiten los datos procesados.
- **Ocultos:** procesan los datos recibidos del "mundo exterior" o de otros "EP's" ocultos y emiten la información generada a otros "EP's" ocultos o a los de salida.

Todo "EP" tiene, como mínimo, una entrada y una salida la cual puede ser aplicada a las entradas de otros "EP's"(incluido el mismo EP).

Como hemos dicho antes todos los "EP's" que forman una **RNA** están interconectados entre sí. Esta interconexión permite transmitir los datos de salida de un "EP", en forma de un número real, a todos los "EP's" con los que se encuentre conectado.

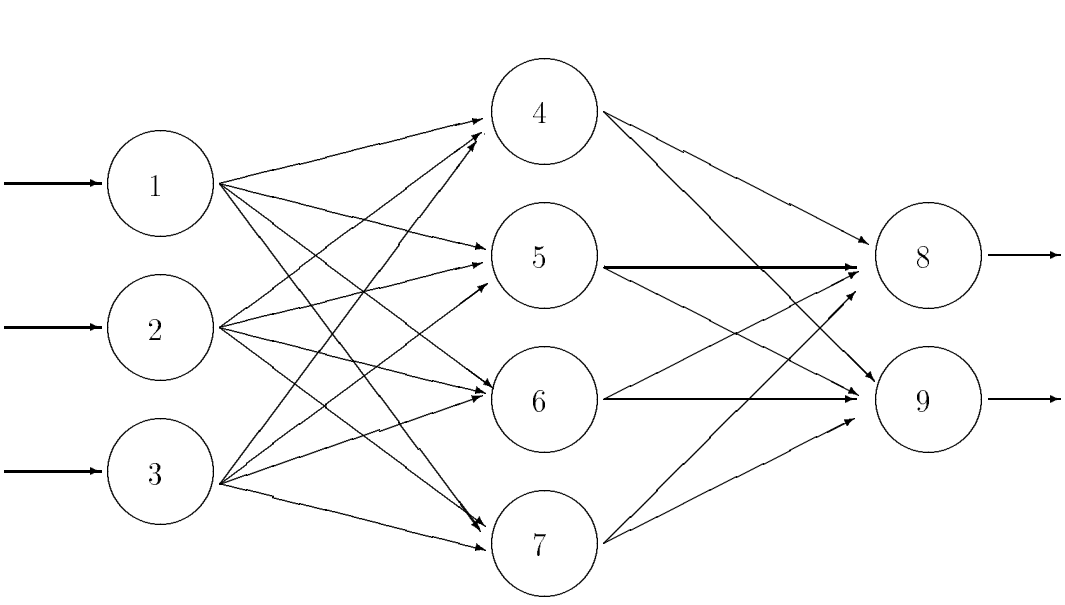

Figura 1.3: Ejemplo de RNA

Ejemplo 1.2 (RNA aplicada al O.C.R.)
Siguiendo con el ejemplo 1.1, *en la página* 12, *como cada letra esta representada por una matriz de dimensión* 7×5 *entonces tendremos* $7 \times 5 = 35$ *"EP's" de entrada y tendremos tantos "EP's" como letras queramos reconocer con nuestra* ***RNA***, *por ejemplo* 26 *para reconocer el siguiente conjunto de caracteres:*

$$A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ \tilde{N}\ O\ P\ Q\ R\ S\ T\ U\ V\ X\ Y\ Z$$

Como cada caracter está representado por una matriz de 7×5, *si ponemos todas las filas de la matriz seguidas tendremos* 35 *digitos, cada uno asociado con un "EP" de entrada. Así mismo como el conjunto de caracteres a reconocer es de* 26 *tendremos* 26 *"EP's" de salida, cada "EP" estará asociado a un caracter del conjunto.*

En algunas **RNA** aparece un "EP" especial, denominado *bias* por los informáticos, el cual no es más que un "EP" con salida constante que está conectado a todas la neuronas de una capa. En lo sucesivo supondremos que dicho "EP" no existe o que tiene salida nula.

1.2.7 Partes de un EP

Como hemos dicho antes una **RNA** esta formada por un conjunto de "EP's" interconectados entre sí. Al estar, todos ellos, interconectados entre sí tiene que haber una forma en la que la información procesada por un "EP" se transmita a los demás para seguir siendo procesada. De esto se encargan las diferentes partes de un "EP".

En un "EP" podemos encontrar:

- **Pesos Sinápticos**, valores asociados a la conexión de un "EP" con otros "EP's". Estos valores se van modificando a medida que la **RNA** va aprendiendo, mediante entrenamiento, para que de esta forma la **RNA** se comporte adecuadamente.

- **Entrada de Red o Entrada Total**, es una suma ponderada de los valores de entrada al "EP", procedentes de los "EP's" conectados con él, y de los pesos sinápticos asociados con dicho "EP".

- **Función de Activación**, esta función nos va a indicar si el "PE" está activo o no, y depende de la *entrada de red* y/o *valor de la función de activación en el instante anterior*.

- **Función de Salida**, en esta función se procesa el valor obtenido en la *función de activación* y el valor obtenido es el valor de salida de la neurona, es decir, la información que suministra a otras neuronas. Normalmente se utiliza como *función de salida* la función identidad, con lo cual el valor devuelto por la *función de activación* es la información mandada a otras neuronas. En lo sucesivo supondremos que esta función es la función identidad, salvo que expresamente se diga lo contrario.

Un ejemplo de esto aplicado a la capa oculta de la **RNA** que aparece en la figura 1.3 de la página 14:

Pesos sinápticos para la RNA de la figura 1.3

EP	EP's que suministran información al EP	Pesos sinápticos
4	1, 2 y 3	$w_{1,4}$, $w_{2,4}$ y $w_{3,4}$
5	1, 2 y 3	$w_{1,5}$, $w_{2,5}$ y $w_{3,5}$
6	1, 2 y 3	$w_{1,6}$, $w_{2,6}$ y $w_{3,6}$
7	1, 2 y 3	$w_{1,7}$, $w_{2,7}$ y $w_{3,7}$

Los pesos asociados a cada "EP" se suelen determinar experimentalmente mediante el "entrenamiento" de la red. El "entrenamiento" se suele realizar introduciendo en la **RNA** las posibles entradas que se pueden presentar en la práctica, o al menos un subconjunto significativo de las mismas, y comparar la salida obtenida con la salida deseada y modificar los pesos mediante algún algoritmo de tal forma que el error cometido por la **RNA** sea nulo o practicamente nulo.

Entrada de red para la RNA de la figura 1.3

La *entrada de red*, en el instante t, de los diferentes "EP's" es:

$$s_j(t) = x_1(t) \cdot w_{1,j} + x_2(t) \cdot w_{2,j} + x_3(t) \cdot w_{3,j} = \sum_{i=1}^{3} x_i(t) \cdot w_{i,j}$$

para $j = 4, 5, 6, 7$. Donde $x_i(t)$, $i = 1, 2, 3$, es la entrada desde el i-esimo "EP", o lo que es lo mismo la salida del i-esimo "EP".

Función de activación para la RNA de la figura 1.3

La *función de activación* no tiene que ser la misma para todos los "EP's" de la **RNA** y dependerá del tipo de problema que estemos resolviendo. Denotemos como F_j a la función de activación para el j-esimo "EP". Normalmente la activación, A_j, para el j-esimo "EP" en el momento depende de la *entrada de red* en el momento t, es decir:

$$A_j(t+1) = F_j(s_j(t))$$

También se puede dar el caso que la activación en el momento $t+1$ dependa de la activación en el momento anterior t:

$$A_j(t+1) = F_j(A_j(t), s_j(t))$$

A las *funciones de activación* también se las conoce con el nombre de *funciones de transferencia.*

La *función logística* es comunmente utilizada como *función de activación.*

$$F_j(t) = \frac{1}{1 + e^{-t}}$$

Esta función toma todos los valores comprendidos entre 0 y 1.

En algunos casos se tiene definido un valor "*umbral*", θ, y dependiendo de si la "entrada de red" es mayor o no que este valor el "EP" correspondiente se activará o no. En caso de activarse mandaría la señal 1 y en caso contrario la señal 0(permanecería inactiva).

1.2.8 Notación para redes neuronales

En lo sucesivo utilizaremos la siguiente notación para una red neuronal:

- n_s número de neuronas en la capa s de la red neuronal.
- $f(x)$ función de activación para la red neuronal.
- $w_{j,i}[s]$ peso sináptico entre la neurona i-ésima de la capa $(s-1)$ con la neurona j-ésima en la capa s.
- $x_j[s](p)$ salida de la j-ésima neurona de la capa s y patrón p.

$$x_j[s](p) = f(I_j[s](p))$$

- $I_j[s](p)$ entrada total de red de la neurona j-ésima en la capa s y patrón p.

$$I_j[s](p) = \sum_{i=1}^{n_{(s-1)}} w_{j,i}[s] \cdot x_i[s-1](p)$$

1.2.9 Redes Feedforward

En este tipo de redes las salidas de los "EP's" de una capa unicamente son entrada de los "EP's" de la capa siguiente. Normalmente este tipo de redes son las más rápidas de todas y representan sistemas lineales.

Podemos ver un ejemplo de una red de este tipo en la figura 1.3 situada en la página 14.

1.2.10 Redes Feedback

En este tipo de redes las salidas de los "EP's" de una capa pueden ser la entrada de los "EP's" de las capas anteriores. Este tipo de redes representa sistemas no lineales.

1.2.11 Redes Feedlateral

En este tipo de redes las salidas de los "EP's" de una capa pueden ser la entrada de los "EP's" situados en la misma capa. Este tipo de redes representa sistemas no lineales.

1.2.12 Redes Recurrentes

En este tipo de redes pueden existir lazos cerrados, es decir, un "EP" le manda información a otro "EP" y este último después de procesar la información le manda información al primer "EP".

1.2.13 El aprendizaje en las RNA

Basicamente el aprendizaje en las Redes Neuronales consiste en encontrar los "pesos sinápticos" adecuados para que la red pueda realizar de una forma eficiente su trabajo.

Aprendizaje por corrección de error

Consiste en tomar un conjunto de pares de datos, entradas con la correspondiente salida en la red, y minimizar el error cometido por la red variando los "pesos sinápticos". El conjunto de pares de datos recibe el nombre de "*conjunto de entrenamiento*".

Aprendizaje por refuerzo

No se dispone de un conjunto completo del comportamiento deseado de la red, ya que no se conoce la salida deseada exacta para cada entrada, sino que se conoce el comportamiento de una manera global para diferentes entradas. La relación entrada-salida se realiza a través de un proceso de éxito o fracaso, produciendo este una señal de refuerzo que mide el buen funcionamiento del sistema. La función del supervisor es más la de un crítico que la de un maestro.

Aprendizaje estocástico

Consiste, básicamente, en realizar cambios aleatorios de los valores de los pesos y evaluar su efecto a partir del objetivo deseado.

1.2.14 El algoritmo "Backpropagation"

Supongamos que tenemos una **RNA** con s capas. El algoritmo de "backpropagation" consiste en ajustar los pesos sinápticos de la red mediante la comparación de resultados obtenidos mediante un conjunto de patrones con los resultados correctos que se deberían obtener con esos mismos patrones.

El nombre de "backpropagation"[10] es debido a que primero se ajustan los pesos de la capa de salida y luego se van ajustando los pesos de las diferentes capas hacia atras.

Con este algoritmo se entrena a la **RNA** mediante un conjunto de patrones, el cual se conoce como patrones de entrenamiento y lo denotaremos como $Entr$.

Supongamos que el conjunto de patrones de entrenamiento para nuestra red es:

$$Entr = \{ p_1, \ldots, p_n \}$$

Definamos los siguientes valores:

- $y_{p,k}$ salida correcta para el patrón de entrenamiento p en la neurona k-ésima de la capa de salida.
- $o_{p,k}$ salida real producida por el patrón de entrenamiento p en la neurona k-ésima de la capa de salida.

Para calcular el error cometido en el procesamiento de cada patrón utilizaremos el error cuadrático medio. Luego el error cometido al procesar el patrón p será:

$$E_p = \frac{1}{2} \sum_{i=1}^{n_s} \delta_{p,k}^2 = \frac{1}{2} \sum_{i=1}^{n_s} (y_{p,k} - o_{p,k})^2 \tag{1.1}$$

[10] Propagación hacia atras.

Según las definiciones la entrada total de red para la neurona k-ésima en la capa s será:

$$I_k[s](p) = \sum_{i=1}^{n_{s-1}} w_{k,i}[s] \cdot x_i[s-1](p) \tag{1.2}$$

donde $x_i[s-1]$ es la salida de la i-ésima neurona de la capa $s-1$. Luego tendremos que:

$$y_{p,k} = f(I_k[s](p)) \tag{1.3}$$

donde $f(x)$ es la función de activación de la red.

Para minimizar el error que comete la red tendremos que ajustar los pesos sinápticos $w_{i,j}$, y esto lo haremos mediante el "método del gradiente"[11]:

$$w_{j+1,k} - w_{j,k} = \Delta_p w_{j,k} = -\gamma \cdot \frac{\partial E_p}{\partial w_{j,k}} \tag{1.4}$$

donde γ es el "factor de aprendizaje" de la red, el cual es positivo y normalmente menor que uno.

Utilizando la regla de la cadena tendremos:

$$\frac{\partial E_p}{\partial w_{j,k}} = \frac{\partial E_p}{\partial I_k(p)} \cdot \frac{\partial I_k(p)}{\partial w_{j,k}} \tag{1.5}$$

Definiendo:

$$\delta_{p,k} = -\frac{\partial E_p}{\partial I_k(p)} \tag{1.6}$$

y derivando (1.2) respecto de $w_{j,k}$ tendremos:

$$\frac{\partial I_k(p)}{\partial w_{j,k}} = x_j(p) \tag{1.7}$$

Sustituyendo (1.6) y (1.7) en (1.5) entonces (1.4) nos queda:

$$\Delta_p w_{j,k} = \gamma \cdot \delta_{p,k} \cdot x_j(p) \tag{1.8}$$

La formula (1.8) nos dará el incremento necesario entre dos pesos para el patrón p.

[11] No haremos referencia a las capas para no complicar más la notación

El calculo de los $\delta_{p,k}$ se puede hacer de forma recursiva desde la capa de salida hacía las capas ocultas:

$$\delta_{p,k} = -\frac{\partial E_p}{\partial I_k(p)} = -\frac{\partial E_p}{\partial x_k(p)} \cdot \frac{\partial x_k(p)}{\partial I_k(p)} \tag{1.9}$$

En la ecuación (1.9) el segundo factor es la derivada de la función de activación $f(x)$:

$$x_k[s](p) = f(I_k[s](p))$$

Para calcular el primer miembro de (1.9) consideraremos los siguientes casos:

- La neurona k-ésima está en la capa de salida.
- La neurona k-ésima está en una capa oculta.

En la capa de salida

En este caso tendremos que:

$$\frac{\partial E_p}{\partial x_i(p)} = -(y_{p,i} - o_{p,i})$$

ya que como estamos en la capa de salida tendremos que:

$$x_i(p) = x_i[s](p) = o_{p,i}$$

Resultando que para las neuronas de la capa de salida tendremos de la ecuación (1.9):

$$\delta_{p,k} = (y_{p,i} - o_{p,i}) \cdot f'(I_k[s](p)) \tag{1.10}$$

En las capas ocultas

En la capa z-ésima para el patrón de entrenamiento p utilizando las definiciones y la regla de la cadena tendremos:

$$\begin{aligned}\frac{\partial E_p}{\partial x_h[z](p)} &= \sum_{i=1}^{n_z} \frac{\partial E_p}{\partial I_i[z](p)} \cdot \frac{\partial I_i[z](p)}{\partial x_h[z](p)} = \\ &= \sum_{i=1}^{n_z} \frac{\partial E_p}{\partial I_i[z](p)} \cdot \frac{\partial}{\partial x_h[z](p)} \Big(\sum_{i=1}^{n_{z-1}} w_{j,i}[z] \cdot x_i[z-1](p) \Big) = \\ &= \sum_{i=1}^{n_z} \frac{\partial E_p}{\partial I_i[z](p)} \cdot w_{h,i}[z] = -\sum_{i=1}^{n_z} \delta_{p,i} \cdot w_{h,i}[z]\end{aligned}$$

en el último paso hemos utilizado la definición dada en (1.6).

Resultando que para las neuronas de las capas ocultas tendremos de la ecuación (1.9):

$$\delta_{p,k} = \Big(\sum_{i=1}^{n_z} \delta_{p,i} \cdot w_{k,i}[z] \Big) \cdot f'(I_k[z](p)) \tag{1.11}$$

Actualización de los pesos

Una vez que se conoce la expresión de los $\delta_{p,k}$, utilizando las ecuaciones (1.4), (1.5), (1.6) y (1.7) o lo que es lo mismo la ecuación (1.8) obtenemos la siguiente formula para la acutalización de los pesos de la capa z-ésima:

$$w_{j+1,k} = w_{j,k} + \gamma \cdot \delta_{p,k} \cdot x_j[z](p)$$

donde $\delta_{p,k}$ viene determinada por (1.10) o (1.11), dependiendo si la capa en la que estamos actualizando los pesos es la capa de salida o es una capa oculta.

Factor de aprendizaje

El valor γ recibe el nombre de "factor de aprendizaje" y ello es debido a que según el valor que tenga el aprendizaje de la red será más o menos rápido.

Este factor es siempre positivo y normalmente es un número pequeño, entre 0.05 y 0.25, para asegurar que la red converge a una solución.

Un valor pequeño de γ significa que la red tendrá que hacer un gran número de iteraciones. A veces es posible incrementar el valor de γ a medida que progresa el aprendizaje. Aumentando γ a medida que disminuye el error de la red suele acelerarse la convergencia incrementando el tamaño del paso conforme el error alcanza un valor mínimo, pero la red puede rebotar, alejándose demasiado del valor mínimo verdadero si γ llegara a ser demasiado grande.

Regla Delta generalizada

El algoritmo que hemos utilizado para ajustar los pesos sinápticos de la **RNA**, utilizando "backpropagation", recibe el nombre de *Regla Delta generalizada*.

Capítulo 2

Construcción del módelo clásico ARIMA

En este capítulo, y mientras no se diga lo contrario, supondremos que los modelos son "estacionarios", con lo cual la media es constante. Luego $\mu_t = \mu = E[z_t]$ para todo t.

De igual modo se tendrá que $\widetilde{z}_t = z_t - \mu$.

Definición 2.1 (Operador de retardo) *Definiremos el operador de retardo, B, de la siguiente manera:*

$$\begin{aligned} Bz_t &= z_{t-1} \\ B^k z_t = \underbrace{B \ldots B}_{k} z_t &= z_{t-k} \end{aligned}$$

Este operador deja invariantes las constantes.

2.1 El proceso autoregresivo general AR(p)

2.1.1 Descripción

Diremos que un proceso z_t es autoregresivo de orden p si:

$$\widetilde{z}_t = \phi_1 \cdot \widetilde{z}_{t-1} + \ldots + \phi_p \cdot \widetilde{z}_{t-p} + a_t \tag{2.1}$$

donde $\widetilde{z}_t = z_t - \mu$ y a_t es un proceso de ruido blanco independiente de z_{t-h} para todo $h \geq 1$ y las ϕ_i son constantes.

Podemos expresar la ecuación (2.1) utilizando este operador:

$$\widetilde{z}_t = (\phi_1 \cdot B + \ldots + \phi_p \cdot B^p)\widetilde{z}_t + a_t$$

o lo que es lo mismo:

$$(1 - \phi_1 \cdot B - \phi_2 \cdot B^2 - \ldots - \phi_p \cdot B^p)\widetilde{z}_t = a_t$$

2.1.2 Ecuación característica

Para simplificar llamemos $\phi_p(B) = 1 - \phi_1 \cdot B - \phi_2 \cdot B^2 - \ldots - \phi_p \cdot B^p$, entonces se tiene que $\phi_p(B)$ es un polinomio de grado p en la variable B. Llamaremos "*ecuación característica*" del proceso a la ecuación:

$$\phi_p(B) = 0 \tag{2.2}$$

Esta ecuación es una ecuación en B, la cual tiene p raices, que generalmente son distintas[1]. Sean estas raices G_1^{-1},...,G_p^{-1}, podemos expresar (2.2) de la siguiente forma:

$$\phi_p(B) = \prod_{i=1}^{p}(1 - G_i \cdot B) \tag{2.3}$$

Teniendo en cuenta la ecuación característica de un proceso **AR(p)** podemos expresar su ecuación como:

$$\phi_p(B)\widetilde{z}_t = a_t \tag{2.4}$$

Esta ecuación es la expresión general de un proceso autoregresivo.

El proceso es estacionario si $|G_i| < 1$ para todo i.

Para obtener información más detallada consultar [1].

[1] Ver Apéndice 15.A de [1].

2.2 El proceso de media móvil general MA(q)

2.2.1 Descripción

La ecuación de un **MA(q)** es:

$$\widetilde{z}_t = (1 - \theta_1 \cdot B - \theta_2 \cdot B^2 - \ldots - \theta_q \cdot B^q)a_t$$

que se suele denotar:

$$\widetilde{z}_t = \theta_q(B)a_t \tag{2.5}$$

El proceso es invertible cuando las raices de $\theta_q(B) = 0$ son, en módulo, mayores que la unidad. Sí q es finito entonces el proceso es estacionario.

2.3 El proceso ARMA(p,q)

Estos procesos son una generalización que permite combinar los procesos $AR(p)$ y $MA(q)$ en un único proceso, $ARMA(p,q)$, que será $AR(p)$ y $MA(q)$.

2.3.1 Descripción

La ecuación de un **ARMA(p,q)** es:

$$(1 - \phi_1 \cdot B - \phi_2 \cdot B^2 - \ldots - \phi_p \cdot B^p)\widetilde{z}_t = (1 - \theta_1 \cdot B - \theta_2 \cdot B^2 - \ldots - \theta_q \cdot B^q)a_t$$

que se suele denotar:

$$\phi_p(B)\widetilde{z}_t = \theta_q(B)a_t \tag{2.6}$$

El proceso es estacionario cuando las raices de $\phi_p(B) = 0$ están fuera del círculo unidad, e invertible si lo están las de $\theta_q(B) = 0$.

2.4 El proceso ARIMA(p,d,q)

El nombre $ARIMA$ viene de las siglas "autoregresive integrated moving average"[2].

2.4.1 Descripción

Dado un proceso $ARMA(p,q)$ cualquiera y permitiendo que el operador AR tenga raices unitarias obtenemos procesos no estacionarios del tipo:

$$(1 - \phi_1 \cdot B - \ldots - \phi_p \cdot B^p)(1 - B)^d \widetilde{z}_t = (1 - \theta_1 \cdot B - \ldots - \theta_q \cdot B^q) a_t$$

Este tipo de procesos recibe el nombre de **ARIMA**(p,d,q), donde:

- p es el orden de la parte autoregresiva estacionaria.
- d es el número de raices unitarias del operador AR. También recibe el nombre de "*orden de homogeneidad del proceso*".
- q es el orden de la parte media móvil.

Llamando $\nabla = 1 - B$ la ecuación que describe este proceso se puede escribir como:

$$\phi_p(B) \nabla^d \widetilde{z}_t = \theta_q(B) a_t$$

Todos los procesos $ARIMA(p,d,q)$ no estacionarios se caracterizan por decrecer lentamente los coeficientes de la función de autocorrelación simple.

[2] Procesos autoregresivos integrados de media móvil.

2.4.2 Procesos $ARIMA$ estacionales

Si la estacionalidad fuese siempre exactamente periódica podría eliminarse de la serie como un componente determinista. En general la estacionalidad es sólo aproximadamente constante y la eliminaremos de la serie tomando diferencias entre observaciones separadas por el período estacional. A estas diferencias las llamaremos "diferencias estacionales".

Sea z_t una serie estacional de período s:

- $s = 12$ para series mensuales.
- $s = 4$ para series cuatrimestrales.
- .....

y con una estructura como la siguiente:

$$z_t = S_t^{(s)} + n_t \tag{2.7}$$

donde:

- $S_t^{(s)}$ es el componente estacional determinista fijo:

$$S_t^{(s)} = S_{t-k\cdot s}^{(s)} \quad k \in \mathbb{Z}$$

- n_t es un proceso estacionario.

Llamando $\nabla_s = 1 - B^s$ y al tomar diferencias de observaciones separadas s períodos tendremos:

$$\nabla_s z_t = \nabla_s n_t$$

con lo que habremos convertido la serie en estacionaria.

Por el contrario, si la estacionalidad no es exactamente constante y se verifica que:

$$S_t^{(s)} = S_{t-k\cdot s} + v_t^{(s)} \quad k \in \mathbb{Z}$$

donde $v_t^{(s)}$ es un proceso estacionario.

Si tomamos diferencias estacionales en (2.7):

$$\nabla_s z_t = v_t^{(s)} + \nabla_s n_t$$

obtendremos, también, un proceso estacionario.

Luego el operador ∇_s convierte procesos estacionales en procesos estacionarios.

Formulación general

Sea z_t una serie estacional con período s conocido y suponiendo que tenemos $n = s \cdot h$ observaciones. Entonces podremos dividir la serie total en s series de h datos, tantas series como nos indique el periodo, que llamaremos $y_\tau^1,\ldots,y_\tau^s$ con $\tau = 1,\ldots,h$.

La relación entre estas series y la original z_t será:

$$y_\tau^{(j)} = z_{j+s\cdot(\tau-1)} \quad (\tau = 1,\ldots,h)\ y\ (j = 1,\ldots,s) \tag{2.8}$$

La relación que obtenemos es $t = j + s \cdot (\tau - 1)$.

Por ejemplo para series mensuales tendremos:

- $s = 12$ indicaría el mes.
- h nos indicaría de cuantos años tendríamos observaciones.

Construyamos ahora un modelo $ARIMA$ para cada una de estas series y supongamos que este modelo es "exactamente" el mismo para todas. El modelo será el siguiente:

$$(1-\Phi_1\cdot B-\ldots-\Phi_P\cdot B^P)(1-B)^D y_\tau^{(j)} = (1-\Theta_1\cdot B-\ldots-\Theta_Q\cdot B^Q)u_\tau^{(j)} \tag{2.9}$$

donde $j = 1,\ldots,s$ y $\tau = 1,\ldots,h$. Además como tenemos estacionalidad $D \geq 1$ ya que si $D = 0$ y las series fueran estacionarias, su modelo podría escribirse:

$$y_\tau^{(j)} = \mu_j + \Psi_j(B)u_\tau^{(j)} \quad (\tau = 1,\ldots,h)\ y\ (j = 1,\ldots,s)$$

donde $\mu_j = E[\{y_\tau^{(j)}\}_\tau]$ y como la serie es estacional tendremos que $\mu_i \neq \mu_j$ para $i \neq j$, con lo que las s series no podrían tener un modelo común.

Podemos escribir los modelos para las s series utilizando (2.8):

$$By_\tau^{(j)} = y_{\tau-1}^{(j)} = z_{j+s\cdot(\tau-2)} = B^s z_{j+s\cdot(\tau-1)}$$

Luego aplicar B a y_τ es lo mismo que aplicar B^s a z_t.

Definamos ahora una serie de ruido común, α_t, asignando a cada t el ruido del modelo (2.9) correspondiente a dicho t:

$$\alpha_t = \alpha_{j+s\cdot(\tau-1)} = u_\tau^{(j)}$$

Luego los s modelos (2.9) podemos escribirlos:

$$(1-\Phi_1\cdot B^s-\ldots-\Phi_P\cdot B^{P\cdot s})(1-B^s)^D\widetilde{z}_t=(1-\Theta_1\cdot B^s-\ldots-\Theta_Q\cdot B^{s\cdot Q})\alpha_t \quad (2.10)$$

donde $t=1,\ldots,n$.

Las series $u_\tau^{(j)}$ son, por hipótesis, ruido blanco, pero la serie α_t para $t=1,\ldots,n$ normalmente no lo será[3]. Suponiendo que α_t sigue un proceso $ARIMA$ regular tendremos:

$$\phi_p(B)\nabla^d\alpha_t=\theta_q(B)a_t \quad (2.11)$$

Sustituyendo (2.11) en (2.10) obtenemos el modelo completo para el proceso:

$$\Phi_P(B^s)\phi_p(B)\nabla^d\nabla_s^D z_t=\theta_q(B)\Theta_Q(B^s)a_t \quad (2.12)$$

La ecuación (2.12), que describe el proceso, recibe el nombre de modelo $ARIMA\ (P,D,Q)_s\times(p,d,q)$ y fue dado por *Box* y *Jenkins* en 1.976. Representa de una forma simple muchos fenómenos reales que encontramos en la práctica. A pesar de su frecuencia no se da siempre.

2.5 Modelos ARIMA en la práctica

Para identificar un modelo utilizaremos todos los datos que tengamos sobre la serie temporal.

2.5.1 Identificación de la estructura del modelo

En esta fase tendremos que decidir las transformaciones necesarias para transformar el proceso en estacionario.

- Tendremos que determinar el número de diferencias que se deben aplicar para que la media sea constante. Este número, d, es normalmente uno o dos.
- Tendremos que determinar si es necesario transformar la serie para que tenga varianza constante.

[3]Existirá en general dependencia entre observaciones contiguas.

Una vez que hemos transformado el proceso en estacionario tendremos que determinar un modelo para el proceso estacionario, es decir, los órdenes p y q de su representación $ARMA(p,q)$, y si el proceso es estacional tendremos que determinar los órdenes P y Q de la estructura $ARMA$ estacional.

Después tendremos que realizar un estudio acerca de la estacionalidad. En caso de presentar estacionalidad con un período s tendríamos que aplicar la diferencia $(1 - B^s)$ para convertirla en estacionaria.

Para ver si una serie temporal es estacional podemos utilizar:

- El gráfico de la serie, el cual presentará una pauta repetida de acuerdo con el período estacional s.
- La función de autocorrelación simple, la cual presentará coeficientes positivos que decrecerán lentamente en los retardos s, $2s$, $3s$, ...

2.5.2 Estimación de los parámetros del modelo

Una vez identificado el modelo estimaremos sus parámetros mediante la minimización de la suma de los cuadrados de los errores a_t.

Para un proceso $AR(p)$ perderemos las p primeras observaciones:

$$\sum_{t=p+1}^{t} a_t^2 = S(\phi_1, \ldots, \phi_p)$$

este tipo de estimaciones recibe el nombre de "estimaciones condicionadas", ya que depende de $z_1, \ldots, z_p$, que son tomados como dato.

Las estimaciones condicionadas obtenidas minimizando S se calculan con las siguientes hipótesis:

- Modelos AR valores iniciales: $z_1, \ldots, z_p$.
- Modelos MA valores iniciales: $a_0, \ldots, a_{-q}$.
- Modelos $ARMA$ valores iniciales: $z_1, \ldots, z_p; a_0, \ldots, a_{-q}$.

2.5.3 Diagnóstico del modelo

Una vez que tenemos construido el modelo hay que comprobar que dicho modelo representa el comportamiento de la serie temporal, esto lo haremos mediante un contraste de hipótesis. Un buen modelo debe cumplir:

- Los residuos del modelo estimado se aproximan al comportamiento de un ruido blanco.
- El modelo es estacionario e invertible.
- Los coeficientes son estadísticamente significativos.
- Los coeficientes del modelo están poco correlacionados entre sí.
- El grado de ajuste es elevado en comparación al de otros modelos alternativos.

El contraste que más se suele utilizar para comprobar que las autocorrelaciones en su conjunto no son significativas es el denominado *Box-Pierce*.

2.5.4 La esperanza condicionada como predictor óptimo

Si tenemos una observacion de longitud T $Z_T = (z_1, \ldots, z_T)$ de una serie temporal y queremos obtener una predicción del valor z_{T+k}, donde $k \in \mathbb{N}$, minimizando el error cuadrático medio. Podemos ver en [1] que:

$$\hat{z}_{T+k} = E[z_{T+k}|Z_T]$$

minimiza el error cuadrático medio. Utilizando este resultado podemos obtener el predictor cuando conozcamos las distribuciones condicionadas. Pero esto, en general, no ocurre y buscaremos predictores, funciones lineales de las observaciones, qeu minimizen el error de predicción.

2.5.5 La ecuación de predicción de un modelo ARIMA

Supongamos que tenemos T datos de un proceso $ARIMA(p,d,q)$ cuyos parámetros son conocidos:

$$z_t = \phi_1 \cdot z_{t-1} + \ldots + \phi_h \cdot z_{t-h} + a_t - \theta_1 \cdot a_{t-1} - \ldots - \theta_q \cdot a_{t-q}$$

donde el operador $\Phi_h(B) = \Theta(B)\nabla^d$ puede tener raices en el circulo unidad, luego $h = p + d$.

La predicción óptima de z_{T+k} será la esperanza condicionada de esta variable dada la realización $Z_T = (z_1, \ldots, z_T)$. Llamando:

$$\begin{aligned} \hat{z}_t(j) &= E[z_{T+j}|Z_T] \\ \hat{a}_t(j) &= E[a_{T+j}|Z_T] \end{aligned}$$

donde T representa el origen de la predicción y j el horizonte. Sustituyendo z_t en la ecuación anterior por z_{T+j} y tomando esperanzas condicionadas a Z_T tendremos:

$$\hat{z}_T(j) = \phi_1 \cdot \hat{z}_T(j-1) + \ldots + \phi_h \cdot \hat{z}_t(j-h) - \theta_1 \cdot \hat{a}_T(j-1) - \ldots - \hat{a}_T(j-q) \quad (2.13)$$

donde:

- $\hat{z}_T(-i) = z_{T-i}$ para $i > 0$.
- $\hat{a}_T(i) = 0$ para $i > 0$.
- $a_{T+i} = 0$ para $i < 0$.

Para $j = 1$ tendremos:

$$a_{T+1} = z_{T+1} - \hat{z}_T(1)$$

que indica que las perturbaciones del modelo son los errores de predicción de un período por delante.

La ecuación (2.13) indica que después de unos valores iniciales la predicción queda determinada por la parte autoregresiva del modelo. En efecto, para $j > q$ las predicciones de (2.13) satisfacen la ecuación:

$$(1 - \phi_1 \cdot B - \ldots - \phi_h \cdot B^h)\hat{z}_T(j) = 0$$

siendo $B \cdot \hat{z}_T(j) = \hat{z}_t(j-1)$, es decir, el operador B actúa ahora sobre el horizonte de la previsión j ya que T, origen, el fijo.

Capítulo 3

Aplicación a series temporales

La predicción de series temporales es importante para gobiernos y empresas, ya que para este tipo de organismos es importante conocer, a priori, estimaciones sobre demandas, costos o ingresos que se producirán en el futuro y, de esta manera poder obrar en consecuencia.

El método más comunmente utilizado para la predicción de series temporales es el método de *Box* y *Jenkins*, el cual hemos visto en el capítulo anterior. Las razones por las que este método es tan usado son dos:

- Es un modelo sencillo.
- Sus predicciones son "precisas".

Pero este método también tiene desvenjas y son que, debido a que utiliza un conjunto de valores previos para predecir el futuro comportamiento de la serie temporal, le es imposible encontrar patrones sutiles en el comportamiento de la serie temporal.

Con el paso del tiempo aumenta la velocidad y potencia de los ordenadores, lo que favorece la busqueda de otros métodos para poder predecir el comportamiento de las series temporales en el futuro, uno de estos métodos es el uso de **RNA**. Las **RNA** son utiles para el reconocimiento de patrones, como vimos en el primer capítulo.

3.1 Redes neuronales RBF

De todos los tipos de **RNA** uno de los más empleados es el RBF[1]. Este tipo de red ha sido empleado en la predicción de series temporales con éxito ya que puede ser entrenado para encontrar complejas relaciones entre datos.

Pero no todo son ventajas, este tipo de red también tiene inconvenientes y es que si utilizamos datos "lejanos" a los patrones de entrenamiento de la red los resultados obtenidos no son muy fiables.

Las redes de tipo RBF y el módelo de Box y $Jenkins$ se complementan muy bien ya que las redes RBF pueden encontrar patrones en los datos, pero no pueden dar resultados significativos si empleamos, como datos de entrada, datos "lejanos" de los patrones de entrenamiento. Y por el contrario el modelo de Box y $Jenkins$ da buenos resultados, en general, pero al no poder determinar patrones de comportamiento en los datos de la serie puede dar predicciones imprecisas.

Ambos métodos se pueden combinar para obtener un modelo más fiable.

3.1.1 Descripción de la red RBF

Este tipo de red es una red de tres capas:

- La primera capa es la capa de entrada de datos, la cual suministra los datos de entrada a los EP's de la segunda capa.
- La segunda capa, capa oculta, se comporta de una forma muy diferente a otro tipo de redes. Cada EP de esta capa representa a un grupo de datos centrados en un punto fijo y con radio conocido.

  Por comodidad supondremos que en esta capa tenemos tres EP's.
- La tercera capa, capa de salida, es la capa que procesa los datos obtenidos en la segunda capa y nos devuelve la predicción para la serie temporal.

Una forma lógica de escoger los C_i sería agrupando todos los patrones de entrenamiento en tantos discos como EP's tengamos en la capa oculta y tomar como C_i el centro de estos discos y como radio el radio de los mismos.

[1] Radial Basis Function.

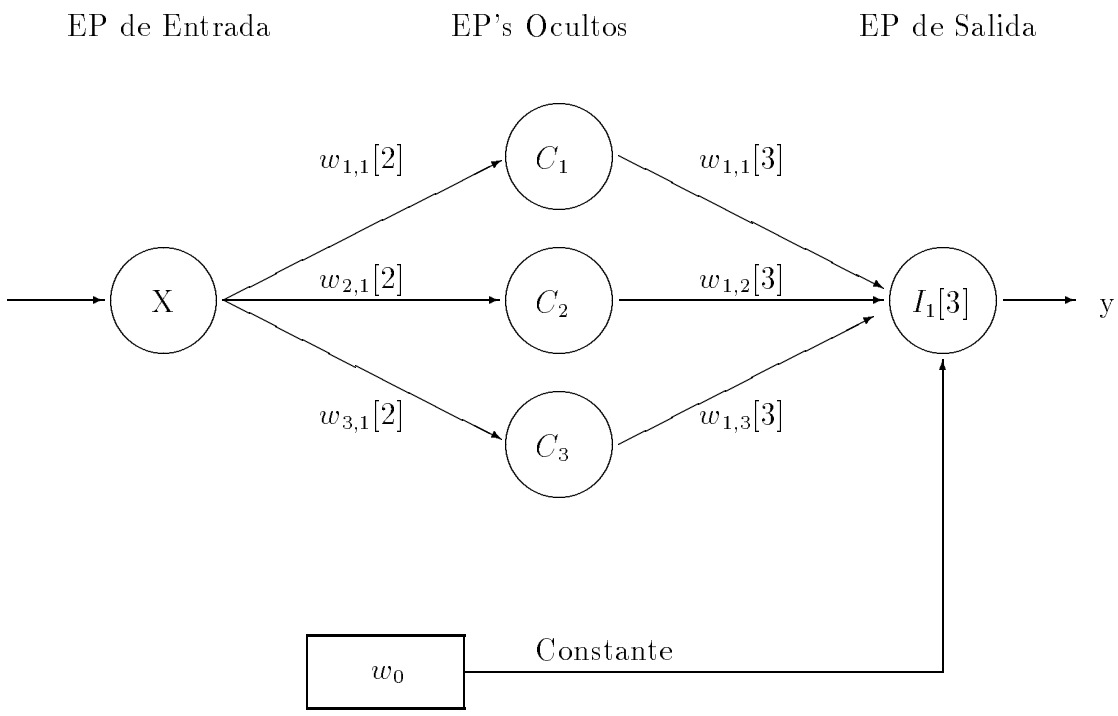

Figura 3.1: Red neuronal del tipo RBF.

3.1.2 Funcionamiento de una red RBF

Supongamos que tenemos un vector $X = (x_1, \ldots, x_n)$, donde $\{x_i\}_{i=1}^{n}$ son valores de una serie temporal cualquiera.

El EP de la primera capa, capa de entrada de datos, suministra el vector X a los EP's de la segunda capa, por lo tanto los pesos sinápticos de estas conexiones serán 1:

- $w_{1,1}[2] = 1$.
- $w_{2,1}[2] = 1$.
- $w_{3,1}[2] = 1$.

Los EP's de la segunda capa funcionan de la siguiente manera:

- Calculan la distancia del vector X al centro del grupo de datos que representan $C_i = (c_{1,i}, \ldots, c_{n,i})$ para $i = 1, 2, 3$.

  Para el calculo de la distancia normalmente se emplea la distancia euclidea:

  $$d(X, C_i) = \sqrt{\sum_{j=1}^{n} (x_j - c_{j,i})^2} \quad para\ i = 1, 2, 3$$

  aunque también puede usarse cualquier otra distancia.

- La distancia $d(X, C_i)$ es transformada por alguna función y el resultado obtenido es la salida del nodo.

  Una función tipicamente usada para transformar las distancias $d(X, C_i)$ es la siguiente:

  $$\phi_i(x) = e^{-(\frac{x}{r_i})^2}$$

  donde r_i es el radio del i-ésimo EP de la segunda capa, el cual es conocido. Estas funciones reciben el nombre de "*funciones gausianas*".

Los pesos sinápticos asociados a las conexiones de esta capa con la capa de salida se determinan de forma experimental mediante algoritmos como el de "backpropagation" que ya vimos.

El EP de la tercera capa funciona de la siguiente manera:

- Calcula la entrada total de red:

  $$I_1[3] = I_1[3](X) = \sum_{i=1}^{3} w_{1,i}[3] \cdot \phi_i(d(X, C_i))$$

- Se le suma un valor constante a la entrada total de red y el valor obtenido es la predicción sobre el comportamiento de la serie temporal:

  $$y = w_0 + I_1[3][X]$$

3.1.3 Ventajas de las redes RBF

Los complejos sistemas no lineales, como son las series temporales, normalmente son dificiles de modelizar usando técnicas de regresión lineal estándar. A diferencia de la regresión las redes neuronales son no lineales. Sus parámetros se determinan mediante entrenamiento con algoritmos como el de "backpropagation".

El mayor problema que tienen las redes neuronales estándar es el de determinar su parámetros. Pero en las redes RBF los únicos parámetros a determinar son los pesos sinápticos de interconexión entre las capas segunda y tercera. Estos tres parámetros[2] son la solución de un sistema lineal y se pueden obtener mediante interpolación[3]. Por lo tanto estos parámetros se calculan de forma más rápida que para otro tipo de redes.

3.1.4 Inconvenientes de las redes RBF

Las redes RBF dan predicciones poco fiables cuando el vector de entrada está lejos del conjunto de entrenamiento. Está lejanía nos la da la distancia que estemos utilizando, normalmente será la distancia euclidea.

En el caso de las funciones gausianas:

$$\phi_i(x) = e^{-(\frac{x}{r_i})^2} = \frac{1}{e^{(\frac{x}{r_i})^2}}$$

cuando el vector de entrada está lejos del centro de los grupos de datos C_i se tiene que:

$$\frac{1}{e^{(\frac{x}{r_i})^2}} \to 0 \quad donde\ x = d(X, C_i)$$

Luego las redes RBF utilizando funciones gausianas son buenas solamente cuando el vector de entrada está proximo a los centros C_i. En caso contrario las predicciones obtenidas son poco fiables.

[2] $w_{1,1}[3]$, $w_{1,2}[3]$ y $w_{1,3}[3]$.

[3] C. Bishop, Improving the generalization propierties of radial basis function neural networks, Neural Computat. 3 (1.991) 579 – 588.

3.2 Redes RBF con factor de seguridad

En los apartados anteriores hemos visto como en algunos casos las predicciones de una red RBF no son fiables. Luego no todas las predicciones de una red de este tipo son buenas predicciones, por lo tanto es muy importante identificar que predicciones son buenas y cuales no lo son.

La forma más directa de solucionar este problema es tener una red RBF que distinga las buenas predicciones de las malas.

Las funciones gausianas ϕ_i toman los valores entre 1.0 y 0.0. Para distinguir las buenas de las malas predicciones utilizaremos un dato al que denominaremos "*factor de seguridad*", el cual vendrá dado por los valores que toman las funciones ϕ_i.

La salida de las neuronas de la segunda capa será de la forma:

$$\phi_i(d(X, C_i)) = e^{-(\frac{d(X,C_i)}{r_i})^2}$$

sabemos que $\phi_i(x) \in [0, 1]$ y el exponente de la función exponencial de ϕ_i nos indica lo siguiente:

- Si $\phi_i(d(X, C_i)) \approx 1$ entonces X está próximo al centro C_i y su aportación a la predicción final es buena.
- Si $\phi_i(d(X, C_i)) \approx 0$ entonces X no está próximo al centro C_i y su aportación a la predicción final puede estropear esta.

En el caso de que todos los EP's tengan valores de ϕ_i próximos a cero la predicción obtenida no será buena.

El problema es como obtener una medida de la precisión de la predicción utilizando estos valores. A primera vista se le puede ocurrir a uno utilizar el máximo de los ϕ_i, pero esto representa un problema y es que puede haber varios EP's con $\phi_i \approx 1$ que hagan que la predicción sea buena y, por el hecho de existir un ϕ_i grande desechar esta predicción.

Para dar una medida de la precisión de la predicción se utiliza la siguiente fórmula recursiva:

$$CF_i = CF_{i-1} + (1 - CF_{i-1}) \cdot max_i(\{\phi_j(d(X, C_j))\}_{j=1}^{N}) \qquad (3.1)$$

donde:

- N es el número de EP's en la capa oculta.
- max_i será el i-ésimo mayor valor de los ϕ_j.
- $CF_0 = 0$ por definición.
- CF_i es el i-ésimo factor de seguridad.
- CF es el factor de seguridad.

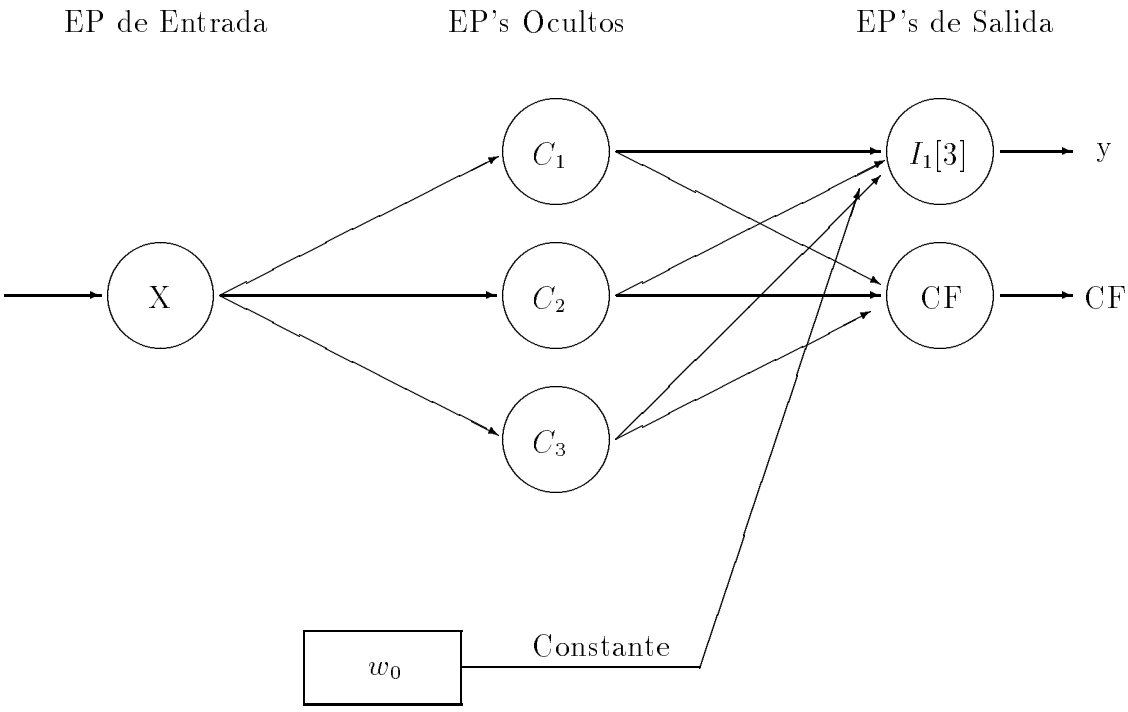

Figura 3.2: Red neuronal del tipo RBF con factor de seguridad.

Normalmente se hace que i varie entre 1 y N, pero también se puede hacer que varie entre 1 y M donde $M < N$. El factor de seguridad que mejor nos indica la precisión de las predicciones será $CF = CF_N$.

En este tipo de red el calculo de los pesos se realiza de igual forma que en la red RBF sin factor de seguridad:

- Los pesos de interconexión entre la primera y segunda capa son todos uno:
$$w_{1,1}[2] = w_{2,1}[2] = w_{3,1}[2] = 1$$
- Los pesos de interconexión entre la segunda y tercera capa se calculan de la siguiente forma:
  - Aquellos que van al EP de predicción se calculan como en el caso anterior. Estos pesos son:
$$w_{1,1}[3] \quad w_{1,2}[3] \quad w_{1,3}[3]$$
  - Aquellos que van al EP que nos da el factor de seguridad son todos iguales a uno:
$$w_{2,1}[3] = w_{2,2}[3] = w_{2,3}[3] = 1$$

3.3 El modelo hibrido

Hemos ido mejorando los modelos de redes.

En primer lugar teniamos una red RBF, la cual podía producir predicciones poco fiables y no teniamos ninguna forma de saber que predicciones eran fiables y cuales no. Para mejorar esto modificamos la red e introdujimos un factor de seguridad, el cual nos indicaba cuando una predicción era fiable o no.

Aunque la red RBF con factor de seguridad ofrece más garantias que la red RBF sin factor de seguridad no tenemos resueltos todos los problemas. En el caso de obtener una predicción no fiable la red RBF no nos da una predicción fiable para ese caso, únicamente nos indica que la predicción no es fiable.

Para solucionar esto utilizamos un modelo hibrido, es decir un modelo que utiliza una red neuronal y un método de predicción, en nuestro caso el de Box y $Jenkins$, y con ambos resultados nos ofrece una predicción.

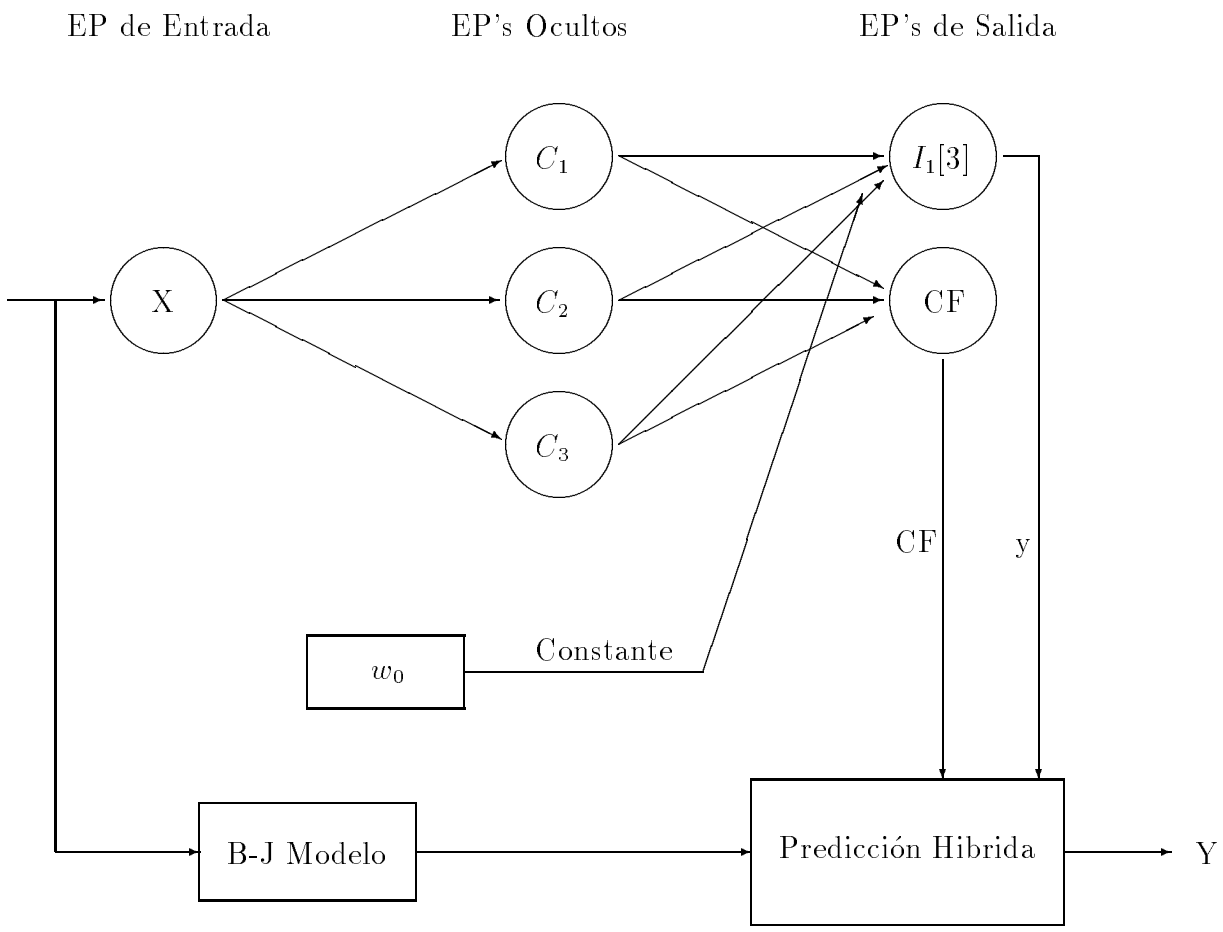

Figura 3.3: Modelo de red hibrida utilizando RBF y el modelo de Box y $Jenkins$.

En este modelo tenemos tres datos para realizar la predicción Y:

- La predicción dada por la red RBF, y_{RBF}.
- El factor de seguridad, CF.
- La predicción dada por el modelo de Box y $Jenkins$, y_{B-J}.

Podemos combinar estos datos de varias formas para obtener la predicción deseada sobre el comportamiento de la serie temporal.

3.3.1 Método de tolerancia

En este método el usuario dará una tolerancia, Tol, para el factor de seguridad. Esta tolerancia se determinará experimentalmente para obtener un grado alto de precisión sobre las predicciones.

El algoritmo para este método es el siguiente:

if ($CF \geq Tol$) then

$Y = y_{RBF}$

else

$Y = y_{B-J}$

3.3.2 Método de las aproximaciones de igual importancia

Este método es muy similar al anterior y en él también se ha de especificar una tolerancia, Tol, la cual se determina de igual forma que en el método anterior.

En el método anterior despreciamos la predicción dada por el modelo de Box y $Jenkins$ cuando $CF \geq Tol$, suponiendo esta predicción como peor que la obtenida por la red RBF, pero esto puede no ser cierto. Es por esta razón que en este método se utiliza la media de ambas predicciones cuando $CF \geq Tol$. El algoritmo para este método es el siguiente:

if ($CF \geq Tol$) then

$Y = \frac{y_{RBF}+y_{B-J}}{2}$

else

$Y = y_{B-J}$

3.3.3 Método de la aproximación según el factor de seguridad

Este método combina las predicciones dadas tanto por la red RBF como por el modelo de Box y $Jenkins$, pero da tanta importancia a la predicción de la red RBF como seguridad nos indique el factor de seguridad. Es decir si el factor de seguridad es 0.80 en la predicción final el 80 % será aportado por y_{RBF} y el 20 % restante será aportado por y_{B-J}.

La predicción por este método vendrá dada por la siguiente fórmula:

$$Y = (CF \cdot y_{RBF}) \cdot (1 - CF) \cdot y_{B-J}$$

3.4 Test de precisión para las predicciones

El criterio por el cual elegiremos un método de predicción será el grado de precisión que tenga el método. Entenderemos que un método es preciso cuando sus predicciones se ajusten a la realidad observada.

El método más utilizado para medir la precisión en modelos para series temporales es el denominado "*Desviación Absoluta Principal*" al que denominaremos MAD:

$$MAD = \frac{1}{T} \sum_{i=1}^{T} (Y_i - \widehat{Y}_i)$$

donde:

- T es un número de predicciones hechas por el modelo.
- $Y_i = Z_i - \mu$ y $\mu = E[Z_i]$.
- $\widehat{Y}_i$ es una predicción para Y_{i-1}.

Capítulo 4

Conclusiones

Este trabajo esta orientado a la descripción de **RNA** y a su utilización para la predicción de series temporales. Por lo que la descripción de series temporales y modelos de regresión no se ha tratado en profundidad.

4.1 Series temporales

Las series temporales son procesos que siguen un comportamiento dificil de modelizar, de forma exacta, mediante una ecuación matemática. Por lo tanto para conocer el comportamiento de una serie temporal en el futuro se ha de modelizar con funciones matemáticas "simples", las cuales nos indican, de forma aproximada, el comportamiento de dicha serie en el futuro.

Para empresas, gobiernos, ... es muy importante conocer estos comportamientos con la mayor precisión posible, por lo tanto se intenta resolver el problema desde distintos puntos de vista y obtener métodos cada vez más precisos para la predicción de series temporales.

4.2 Redes neuronales

Las redes neuronales tienen un alto grado de paralelismo y son muy utilizadas para el reconocimiento de patrones. Gracias a esto también se pueden utilizar para el reconocimiento de pautas de comportamiento.

Es por esto que se utilizan para resolver aquellos problemas en los que no podemos dar un modelo matemático exacto o aquellos problemas que son de dificil resolución mediante programación secuencial.

4.3 Modelos $ARIMA$

Para la predicción de series temporales hemos utilizado los modelos $ARIMA$ ya que son los modelos que más se ajustan a las series temporales, aunque no a todas, y son un modelo facil de utilizar para predecir el comportamiento de una serie temporal.

Estos modelos tienen inconvenientes derivados de ser modelos univariantes, es decir, únicamente utiliza los estados anteriores de la serie para determinar el comportamiento futuro de la serie.

En la mayoría de las series temporales intervienen procesos ajenos al comportamiento pasado de la serie, y es por esto que estos modelos tienen sus inconvenientes. No pueden reconocer patrones sutiles de comportamiento en la serie temporal.

La utilización práctica de modelos $ARIMA$ se ha descrito de una forma breve, para una información más detallada ver [1], [2] y [4].

4.4 Predicción con redes neuronales

Debido al grado de paralelismo y las capacidades de las redes neuronales para el reconocimiento de patrones solucionan el problema del reconicimiento de patrones sutiles de comportamiento en las series temporales.

A pesar de esto hemos visto que algunas veces la predicción de una red neuronal puede no ser fiable. Para solucionar estos problemas se introdujeron los factores de seguridad y los modelos hibridos, mediante los cuales se obtienen valores más fiables que en los modelos de redes neuronales puros.

Para obtener buenos resultados con estos métodos tenemos que elegir el método más preciso. El modelo que mejores resultados da es el hibrido, pero para dar predicciones con este modelo tenemos tres posibles. La forma para elegir uno será comprobar los tres métodos y elegir el más preciso.

Antes de hacer esto hemos de entrenar la red con un conjunto de patrones que hagan que la red se comporte de la "mejor" forma posible para nuestros propositos.

El modelo hibrido tiene la ventaja de poder ser utilizado con otro método de predicción que no sea el de *Box* y *Jenkins* con muy pocos cambios. De esta manera si no se obtienen resultados precisos con este modelo se puede cambiar el modelo de *Box* y *Jenkins* por otro modelo, ya sea univariante o multivariante, con el cual se logren predicciones que se ajusten a nuestras necesidades.

En [7] se pueden ver ejemplos del uso de estos métodos a casos reales.

Bibliografía

[1] Estadística modelos y métodos 2. Modelos lineales y series temporales. 2^a edición revisada. Daniel Peña Sánchez de Rivera. Alianza Universidad Textos.

[2] Modelos econométricos y predicción de series temporales. José M. Otero. Editorial AC.

[3] Estadística para ingenieros. Ramón Ardanuy Albajar. Quintín Martín Martín. Editorial Hespérides.

[4] Modelos econométricos. Antonio Pulido. Editorial Pirámide.

[5] Algoritmos genéticos sobre redes neuronales. Proyecto fin de carrera de Pablo Cabezas Mateos.

[6] Aplicaciones de las redes neuronales a la I.O. Estructura y funcionamiento de las Redes Neuronales Artificiales. F. S. Wong. Articulo proporcionado por D. Quintín Martín Martín.

[7] Predicción de series temporales combinando redes RBF, factores de seguridad, y el modelo de *Box-Jenkins*. Donald K. Wedding II, Krzysztof J. Cios. Articulo proporcionado por D. Quintín Martín Martín.

[8] Simulador de Redes Neuronales Stuttgart. Articulo de Ed Petron, publicado en el número 3 de la revista "*LiNUX Journal*" (edición en Castellano).

[9] Redes neuronales. Algoritmos, aplicaciones y técnicas de programación. James A. Freeman y David M. Skapura. Addison-Wesley.