Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA ELÉCTRICA

DESARROLLO DE UNA HERRAMIENTA INFORMÁTICA PARA PREDECIR EL CONSUMO ENERGÉTICO DE UN EDIFICIO

Autor:

Sandonís Gatón, Héctor

Director:

Moríñigo Sotelo, Daniel

Dpto. Ingeniería Eléctrica

Co-director

Espí García, Fernando Javier

Valladolid, Junio 2016

Agradecimientos

A Daniel Moríñigo Sotelo, gracias por haberme dado la oportunidad de esta experiencia.

A mis padres, gracias por apoyarme y alentarme en todos estos años, por vuestro sacrificio y vuestro cariño. No podría haberlo hecho sin vosotros.

A Esther, gracias por tus consejos, por apoyarme y sobre todo por ser como eres.

A mis compañeros de clase, siempre era mas fácil venir a Universidad sabiendo que estaba alguno de vosotros.

Al resto de mi familia, especialmente a mis abuelos, gracias por haber estado siempre ahí.

A todos mis amigos, vosotros sabéis quiénes sois, gracias por animarme y por haberme hecho disfrutar de estos años de carrera. Con vosotros todo ha sido mucho más fácil.

A todos vosotros, mi más sincero agradecimiento.

Resumen

El siguiente proyecto está dedicado al estudio y realización de una Red Neuronal Artificial para predecir el consumo de energía eléctrica del Hospital Universitario Pío del Río Hortega.

Se enmarca dentro de un acuerdo entre el Departamento de Ingeniería Eléctrica de la Universidad de Valladolid y el Hospital Universitario Río Hortega.

Es un hospital público integrado dentro de la Red Asistencial de la Gerencia Regional de Salud y Castilla y León (Sacyl) y sus dimensiones son de 115.354 m^2. Este centro es el responsable de la atención sanitaria de la parte oeste de Valladolid.

Cabe destacar que trabajan unas 3200 personas y dispone de aproximadamente 600 camas.

Palabras clave

Red Neuronal Artificial, Predicción, Suavizado de Curvas, Spline Cúbico, Matlab.

Capítulo 1

Justificación y objetivos

1 Justificación y objetivos

Se va a estudiar, desarrollar e implementar una herramienta informática que permita estimar la demanda de energía eléctrica del Hospital Pío del Río Hortega. Dicha herramienta estará basada en Redes Neuronales Artificiales que ajusten el comportamiento del hospital a partir de ciertas variables explicativas. Deberá ser capaz de realizar una previsión de la demanda de energía a partir del pronóstico de las variables explicativas elegidas.

Debido a que la electricidad no se puede almacenar en grandes cantidades, es necesario un equilibro constante entre generación y consumo. En caso de no cumplirse este equilibrio, la empresa suministradora tendría pérdidas tanto si es por falta de suministro(en forma de denuncias de los clientes), como si es por un exceso de producción (se pierde la energía que no se utiliza).

Nosotros vamos a centrarnos en predecir el consumo de los consumidores, ya que por algunas razones, en algunos casos también necesitan conocer el gasto que tendrán en el futuro. Este conocimiento les puede permitir:

- Planificar el trabajo en la instalación siempre que sea posible, con el fin de alisar la curva de carga lo máximo posible y reducir de esta forma los picos que se producen en esta.
- Automatizar el control de los transformadores del Centro de Transformación, con el fin de que solo trabajen los que sean necesarios en función de la potencia demandada. De esta forma se reducirían las pérdidas por trabajo en vacío, y se podría ajustar más la curva de carga del transformador con la de máximo rendimiento de éste.
- Dimensionar las instalaciones cuando los dispositivos actuales llegan al fin de su vida útil, de tal forma que los nuevos se adecúen mas a las futuras condiciones de trabajo.

Para poder llevar a cabo esta predicción vamos a utilizar una Red Neuronal Artificial y, además nos apoyaremos en el programa informático Matlab.

Se empleará una serie de datos históricos con el fin de poder estimar la futura curva de carga. Lo que haremos será estimar la demanda horaria, pero si queremos periodos mayores de estimación, realimentaremos la red con los datos de carga estimados.

Este trabajo se basa en un Proyecto Fin de Carrera anterior (David Carnicero Vaquero, julio de 2001) sobre el mismo tema.

Como la curva de demanda de un hospital presenta muchas variaciones bruscas que dificultan el entrenamiento de la red, se tratarán previamente las curvas diarias para alisarlas o suavizarlas. Esto se realizará aplicando una técnica numérica basada en splines cúbicos.

Las curvas obtenidas serán las que se empleen para entrenar la red . Compararemos esos resultados con los previos sin el filtrado de las curvas.

En el ámbito de la Unión Europea, el Parlamento y el Consejo redactaron y publicaron en 2002 la Directiva 2002/91/CE relativa a la Eficiencia Energética de los Edificios, de aplicación obligatoria en los países miembros (entre los cuales se encuentra España), una vez trascurrido el periodo transitorio de adecuación correspondiente.

Esta directiva impulsa la consecución de la mayor eficiencia energética posible en todas y cada una de las instalaciones que concurren en un edificio. Por este motivo vamos a intentar predecir la curva de demanda eléctrica del hospital.

Debido al cumplimiento de esta normativa, se están haciendo multitud de esfuerzos enfocados a reducir el consumo energético en todas las instalaciones. Nuestro estudio estará enfocado en reducir el consumo de los transformadores del Centro de Transformación del Hospital.

En ningún momento debemos olvidarnos que se trata de un hospital, donde el objetivo fundamental y principal es garantizar el suministro de energía eléctrica. Cualquier otro objetivo, como el que planteamos en este trabajo, siempre estará supeditado al mencionado anteriormente.

La estrategia de control supondrá un ahorro energético, pero más importante será el aumento de la vida de los transformadores.

Para llevarla a cabo contaremos con una herramienta de software matemático, Matlab, Excel y los datos proporcionados por los analizadores de redes y sondas de medida que se encuentran en el hospital.

Este TFG se ha realizado en el Hospital Universitario Pío del Río Hortega dentro de un marco de colaboración entre este y la Universidad de Valladolid. Gracias a este acuerdo, he podido realizar allí las practicas obligatorias. Una de mis tareas era la recogida de datos con el fin de realizar el estudio posterior. Fruto de un problema en un analizador de redes, no se guardaron

bien los datos que se recogían periódicamente, lo cual ha lastrado todo el desarrollo del TFG, no pudiendo contar con los datos suficientes para hacer un estudio mucho más completo.

Capítulo 2

Técnicas de estimación de la demanda eléctrica

3 Introducción

El consumo de energía ha ido evolucionando a lo largo de la historia aumentando la demanda de esta con el paso del tiempo. Esto ha sido debido al avance del nivel de vida de la sociedad, siendo esta cada vez mas dependiente de la energía eléctrica. Las necesidades energéticas no son siempre iguales, lo cual ha repercutido en el campo de la previsión eléctrica. Las técnicas y medios de previsión usados se han basado en el potencial económico de cada momento.

2.1 Historia

Durante los años 50 y 60 el precio de los combustibles era estable, las economías de escala se aplicaban a las plantas de generación, la tendencia de la población era previsible, y la carencia de oferta y demanda en el sector eran sus principales características. Debido a todo esto, era muy fácil estimar los futuros picos de demanda mediante sencillas técnicas de predicción.

Pero a partir de los años 70 todo esto cambió de forma abrupta. Todo empezó con la crisis del petróleo en 1973, provocando una recesión económica mundial hasta el año 1975. Se pasó de la estabilidad en el precio de la energía a una gran volatilidad, que sumado al inesperado aumento de la demanda de los consumidores, el aumento del precio de la energía eléctrica era la principal preocupación de gobiernos y usuarios. La programación de la energía eléctrica sería desde este momento el protagonista dentro de las empresas del sector. Después se empezaron a desarrollar y aplicar técnicas mucho mas precisas.

Como siempre pasa en periodos de incertidumbre, se produjo un gran avance de la teoría estadística en lo que se refiere a técnicas de análisis de series temporales. Sobre todo a partir de la publicación del libro T*ime Series Analysis: Forecasting and Control* de los autores G.E.P. Box y G.M. Jenkins. Esto dio nombre a la metodología Box-Jenkins, la cual ha demostrado grandes resultados en multitud de análisis económicos, como es el caso del sector energético.

Lo que ocurrió en los países desarrollados a partir de 1990 es que en el sector de la energía empezó una desregularización, lo cual supuso un punto de inflexión, pasando a ser otro producto más, basado en la ley de la oferta y la demanda.

El problema de errar en la predicción de la demanda de energía es tener un déficit o superávit de ésta, lo cuál revierte en mas costes a sumar al precio. Como esto es algo inadmisible en un sector tan competitivo, las empresas invierten muchos recursos en técnicas de predicción de la demanda de electricidad con el fin de conseguir una ventaja competitiva.

Debido a toda esta evolución y la generalización del uso de estas técnicas, lo vamos a aplicar a la predicción del consumo de energía eléctrica del Hospital.

2.2 Perspectivas temporales de previsión

Lo primero que tenemos que tener en cuenta antes de construir el modelo, es la distancia temporal para la que queremos realizar nuestra previsión de demanda eléctrica. Para acotar las perspectivas de predicción, se ha realizado una posible clasificación en función del problema que se presenta en este TFG:

2.2.1 Corto plazo

Aquí nos moveremos entre los 30 minutos hasta la semana desde que se efectúa la predicción. Lo más habitual en este escenario es planificar las operaciones tácticas y tomar decisiones del día a día

- Factores que influyen en el corto plazo

  En la necesidad de energía eléctrica hay una naturaleza variable con el tiempo y afectada por diferentes factores (meteorológicos, económicos, sociales…). En principio, cuando la previsión es a corto plazo, las condiciones sociales y económicas no tienen influencia en las series. Sin embargo, el clima puede tener una gran influencia en la demanda. Normalmente el valor máximo de consumo suele coincidir cerca del extremo máximo y mínimo de temperatura. Los autores dicen que la curva de demanda horaria debe coincidir con el perfil de temperaturas diarias, pero hay que tener en cuenta que se trata de un hospital y tiene bastantes particularidades. En este grupo lo mas habitual es incluir las variables como hora del día, temperatura horaria, humedad, velocidad del viento , luminosidad, etc.

2.2.2 Medio plazo

Se entiende por medio plazo la previsión que abarca un periodo desde un mes hasta un año a partir del momento en que se estima la futura demanda.

Esta previsión se puede utilizar para obtener condiciones mas ventajosas en un contrato con una compañía eléctrica.

2.2.3 Largo plazo

El largo plazo suele abarcar la predicción desde uno a diez años en el futuro, aunque es variable dependiendo de los autores que se consulten. Es una previsión muy importante, ya que así se podría programar con suficiente antelación los posibles cambios que se necesiten realizar en una instalación.

- Factores influyentes a largo plazo

  A medida que crece el horizonte temporal, es necesario incluir nuevas variables explicativas. Hay que añadir a las citadas en el medio plazo, el precio de la electricidad, el Producto Interior Bruto (PIB) así como los indicadores económicos más relevantes.

Para estimar la demanda eléctrica existen multitud de métodos, y estos se van a elegir en función de las distintas variables, como pueden ser la longitud en el tiempo, las variables disponibles o la precisión necesaria.

A continuación se exponen las técnicas más utilizadas.

2.3 Series de Tiempo

Se utilizarán series temporales para crear un modelo que nos permita llegar al objetivo de predecir el consumo de electricidad del Hospital. A continuación se explican los métodos mas utilizados de predicción a través de series temporales.

- Modelos univariados: Predicen el futuro de una serie con base en su comportamiento histórico propio; son muy útiles si el patrón detectado en el pasado se mantiene hacia el futuro, de lo contrario no son aconsejables. Los modelos ARIMA (modelo autorregresivo integrado de promedio móvil) son representativos de este grupo.
- Modelos causales: Requieren la identificación de otras variables que se relacionan de la manera causa efecto con la variable que se desea predecir. Una vez identificadas estas variables relacionadas, se construye un modelo estadístico que pretende describir la relación entre estas variables y la variable que se desea pronosticar. Los modelos de regresión lineal simple y los

modelos de regresión lineal múltiple son los más conocidos de este grupo.

2.3.1 Composición de una Serie Temporal:

A continuación se expone una posible clasificación de las series temporales:

- A corto plazo
  - Factores periódicos: estaciones del año, periodos especiales (Navidad, etc.)
  - Puntuales: una catástrofe natural, la crisis financiera de un país, la quiebra de una gran empresa
- A medio plazo: crisis y recuperación económica
- A largo plazo: cambios estructurales en la población o en la economía

Una serie temporal cuenta con 4 componentes fundamentales:

- Estacionalidad: comportamiento periódico a corto plazo, repetido dentro de periodos más largos.
- Irregularidad: factor a corto plazo puntual e impredecible, no explicado por los otros componentes.
- Ciclo: comportamiento a medio plazo, asociado normalmente a los ciclos.
- Tendencia: comportamiento a largo plazo de la serie.

2.4 Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son conjuntos de elementos de cálculo simples, normalmente adaptativos, interconectados masivamente en paralelo y con una organización jerárquica que le permite interactuar con algún sistema del mismo modo que lo hace el sistema nervioso biológico.

Debido a su aprendizaje adaptativo, tolerancia a fallos, operación en tiempo real, su utilización se extiende a áreas tan dispares como la biología, finanzas, industria...

El capítulo 4 está dedicado a este método estimación de la demanda.

2.5 Modelos Econométricos

Estos modelos tratan de suplir las carencias de otras técnicas que no tienen en cuenta la no linealidad de los sistemas económicos, a continuación se expone el que mejor se adapta a la predicción de la demanda eléctrica:

2.5.1 Teoría de la Cointegración

Se trata de construir una serie de modelos que nos permitan conocer como inciden en la demanda eléctrica de una zona o región las distintas variables que la configuran. Para conseguir este objetivo se seguirán los desarrollos, tanto teóricos como empíricos, aportados por diversos autores en el ámbito de la modelización de la demanda turística , dándonos, en particular, una idea bastante aproximada de las variables que debemos utilizar para nuestro estudio analítico [Talaya E.].

Las ventajas de utilizar la econometría a la metodología analítica son las siguientes:

- Los modelos econométricos presentan la ventaja de que tienen en cuenta explícitamente el impacto que sobre la demanda eléctrica tienen los cambios en las variables explicativas.
- Los modelos econométricos proporcionan varias medidas estadísticas sobre el significado y la precisión de las ecuaciones de predicción.

Es cierto que este método no es el más utilizado en la predicción de demanda eléctrica, pero sí es aplicado en otros ámbitos con un buen resultado.

El análisis de cointegración permite detectar si existe la posibilidad de obtener estimaciones correctas, es decir, que los resultados sean fiables, de los parámetros que definen las relaciones entre dos o más variables.

Este método se puede utilizar tanto a corto plazo como a largo plazo. Una de las características importantes de los sistemas formados por variables cointegradas es que admiten la formulación en términos de un Modelo de Mecanismo de Corrección del Error, cuya expresión general es la siguiente:

$$\nabla Y_t = \propto^* (L)\Delta Y_{t-1} + \beta(L)\Delta X_t - \propto [Y_{t-1} - gX_{t-1}] + a_t \quad \text{(II.1)}$$

Como se observa, este modelo es una formulación estacionaria de variables no estacionarias que hace uso de la restricción a largo plazo que

liga a esas variables por el hecho de estar cointegradas, poniendo de manifiesto que cuando hay más de una variable y existe cointegración, la transformación estacionaria no se logra diferenciando, sino que requiere la inclusión del término de corrección del error junto con términos en diferencias.

Por tanto, los modelos de mecanismo de corrección del error permiten modelizar tanto las relaciones de largo plazo como la dinámica a corto plazo de las variables. La denominación de modelo de corrección del error se debe a la especificación del modelo, en donde las desviaciones de la relación a largo plazo entre los niveles de las variables funcionan como un mecanismo que impulsa a los cambios de las variables a acercarse a su nivel de equilibrio cuando se han alejado de éste, es decir, corrigen los errores de desequilibrio de periodos anteriores de forma gradual.

2.6 Modelo ARIMA

Se trata de un modelo estadístico para series temporales desarrollado por Box y Jenkins. La particularidad de este modelo es tener en cuenta la dependencia existente entre los datos, lo que significa, que cada observación en un momento dado es modelada en función de los valores anteriores. El nombre ARIMA por sus siglas en inglés, que deriva de sus tres componentes AR (Autorregresivo), I (Integrado) y MA (Medias Móviles).

La principal característica del modelo ARIMA es que permite describir un valor como una función lineal de datos anterior y errores debidos al azar, pudiendo además, incluir un modelo cíclico. Box y Jenkins recomiendan tener como mínimo 50 observaciones en la serie temporal para describir correctamente ese fenómeno.

A continuación se resume brevemente la metodología explicada por Box y Jenkins (Box, G., Jenkins, G.):

2.6.1 Identificación

Es la primera fase y más importante. Consiste en identificar el posible modelo ARIMA que sigue la serie.

Se trata de determinar los parámetros *p*, *d* y *q* que conforma el proceso ARIMA (p,d,q) generador de la serie. Hay que decir que aunque estos parámetros pueden adoptar cualquier valor, en la práctica casi la totalidad de los casos serán 0 ó 1, y raras veces 2, lo que hace que el proceso de identificación sea menos complejo de lo que aparentemente resulta.

Primero se determina el parámetro *d*, que es el grado de diferenciación para que la serie sea estacionaria.

Para identificar su valor, se observa gráficamente si la serie es estacionaria. Para esto, la serie debe de tener la misma media y variancia a lo largo de toda la serie. En caso de no ser estacionaria se procede a diferenciarla y comprobar de nuevo gráficamente si es estacionaria.

En caso de ser estacionaria, quiere decir que el valor d vale 1, en caso contrario se diferencia nuevamente, y así tantas veces hasta lograr la estacionalidad. El valor *d* corresponderá con el numero de veces que se haya diferenciado.

Conocido el valor *d* se deben buscar los valores de *p* y *q*. Para ambos parámetros recurriremos a la función de autocorrelación ACF y a la función de autocorrelación parcial PACF.

Los modelos AR(p) presenta un decaimiento exponencial en los valores de ACF y picos en los primeros *p* valores del PACF. Por otro lado, los modelos MA(q) presentan *q* picos en los primeros *q* valores del ACF, y valores decrecientes exponencialmente en PACF.

2.6.2 Estimación

En la segunda fase se selecciona provisionalmente un modelo para la serie estacionaria, se pasa a la segunda etapa de estimación, donde los parámetros AR y MA del modelo se estiman por máxima verosimilitud y se obtienen sus errores estándar y los residuos del modelo.

2.6.3 Diagnóstico

La tercera fase se trata de hacer el diagnóstico, se comprueba que los residuos no tienen estructura de dependencia y siguen un proceso de ruido blanco. En caso de que los modelos muestren estructura se modifica el modelo para incorporarla y se repiten las fases anteriores hasta obtener un modelo adecuado.

2.6.4 Aplicación del modelo

En la cuarta fase se realizan las predicciones del modelo.

2.7 Selección de modelos

Si entendemos que una predicción es mejor que otra cuando comete menor error, los criterios de selección de modelos serían el error cuadrático medio (ECM), error absoluto medio (EAM) y error absoluto porcentual medio (EAPM). Estos indicadores se calcularían a período histórico, es decir, se calcularían los valores que el modelo ofrece para las H últimas observaciones y se compararían con el valor real, del siguiente modo:

$$ECM(H) = \frac{l}{H}\sum_{l-1}^{H} e_{T-H}^2(l) = \frac{l}{H}\sum_{l-1}^{H}[y_{T-H+l} - y_{T-H}(l)]^2 \text{ (II.2)}$$

$$EAM(H) = \frac{l}{H}\sum_{l-1}^{H} |e_{T-H}^2(l)| = \frac{l}{H}\sum_{l-1}^{H} |y_{T-H+l} - \hat{y}_{T-H}(l)| \text{ (II.3)}$$

$$EAPM(H) = \frac{1}{H}\sum_{l-1}^{H} \frac{|e_{T-H}(l)|}{y_{T-H+l}} * 100 = \frac{1}{H}\sum_{l-1}^{H} \frac{|y_{T-H+l} - \hat{y}_{T-H}(l)|}{y_{T-H+l}} * 100 \text{ (II.4)}$$

El problema es que estos indicadores no tienen en cuenta la estructura estocástica del modelo, no informan sobre alguna característica estocástica supuesta sobre el período extramuestral.

Capítulo 3

Técnicas de interpolación de curvas

3 Introducción

Los datos que se van a emplear para el entrenamiento de la red neuronal presentan mucha variación local, lo cual no es lo mas recomendable para esta tarea.

Será necesario utilizar alguna técnica de suavizado de curvas para pretratar los datos antes de ser utilizados para el entrenamiento de la red.

3.1 Descripción

En este apartado se exponen las principales herramientas para entender en que consisten las técnicas de introducción de curvas para después entender más fácilmente las explicaciones de las técnicas más utilizadas en la predicción de la demanda de energía eléctrica.

Para comprender los distintos métodos de interpolación de curvas, se van a utilizar las figuras 3.1, 3.2, 3.3 donde se muestran distintas curvas trazadas a partir del mismo conjunto de datos para observar las principales diferencias visualmente:

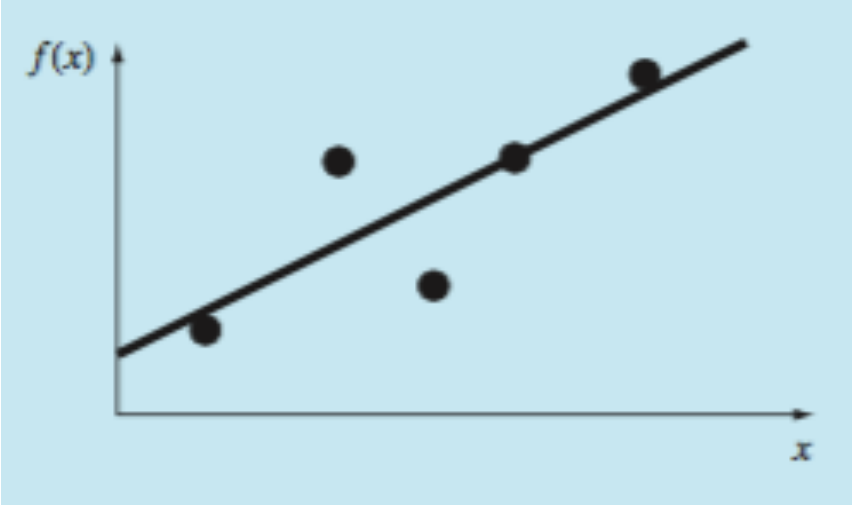

Figura 3.1- Línea recta

Consiste en la caracterización de la tendencia general ascendente con una línea recta.

Figura 3.2.-Unión de puntos mediante línea recta

Interpolación lineal para unir los distintos puntos a través de una línea recta.

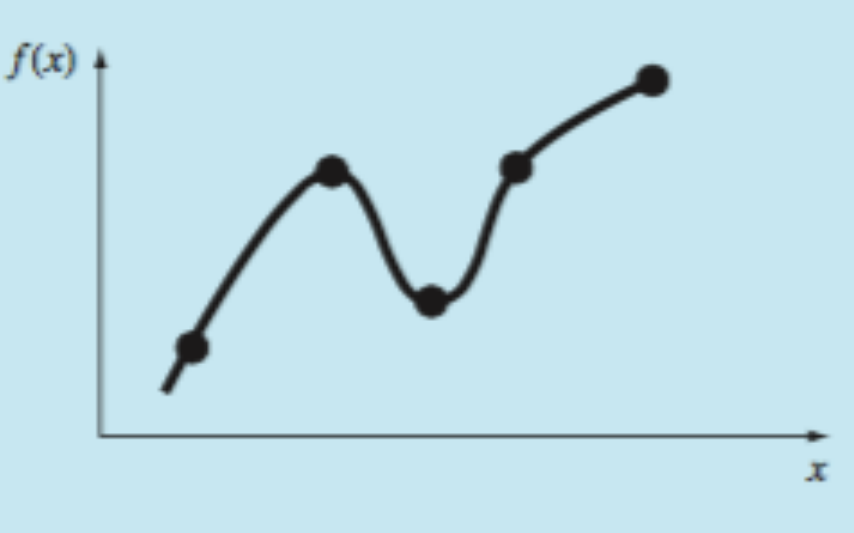

Figura 3.3.-Unión de puntos mediante línea curva

Curvas suaves para tratar de capturar el serpenteado sugerido por los datos

3.2 Conceptos básicos

A continuación se exponen los siguientes conceptos estadísticos para comprender mejor las técnicas que se explican mas adelante.

3.2.1 Media aritmética($\overline{y}$)

Se trata del estadístico de posición mas utilizado. Se define como la suma de los datos (y_i) devidida entre el número de datos (n)

$$\overline{\boldsymbol{y}} = \frac{\sum \boldsymbol{y_i}}{\boldsymbol{n}} \qquad \text{(III-1)}$$

3.2.2 Desviación estándar (s_y)

Es la medida de dispersión de una muestra mas común.

$$\boldsymbol{S_y} = \sqrt{\frac{\boldsymbol{S_t}}{\boldsymbol{n-1}}} \qquad \text{(III-2)}$$

donde S_t es la suma total de los cuadrados de las diferencias entre los datos y la media

$$\boldsymbol{S_t} = \sum (y_i - \bar{y})^2 \qquad \text{(III-3)}$$

Si las mediciones se encuentran muy dispersas alrededor de la media, *St* (y, en consecuencia, *Sy*) será grande. Si están agrupadas cerca de ella, la desviación estándar será pequeña.

3.2.3 Coeficiente de variación (c.v)

Es un estadístico final que se utiliza para cuantificar la dispersión de los datos. Es el cociente de la desviación estándar y de la media, proporcionando de esta manera una medición normalizada de la dispersión.

$$\mathbf{c.v} = \frac{\mathrm{s_y}}{\bar{\mathrm{y}}} 100 \qquad \text{(III-4)}$$

Como se puede observar se parece al error relativo porcentual. Por lo tanto, podemos decir que es la razón de una medición de error (s_y), respecto a un estimado del valor verdadero ($\overline{y}$).

3.2.4 Distribución normal

Es una herramienta para ver la distribución de datos. A través de un histograma podemos tener una representación visual simple de la distribución.

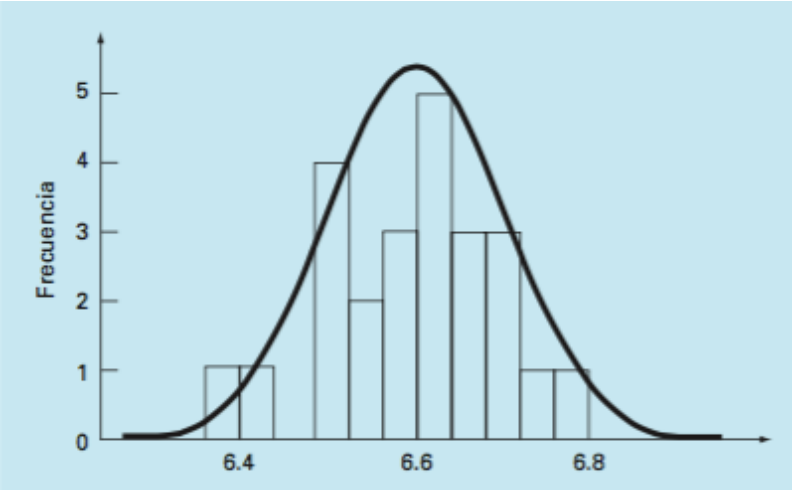

Figura 3.4.-Histograma

En la figura 3.4 observamos que la mayoría de medidas se concentra cerca del valor de la media de 6.6.

Si el conjunto de datos es muy grande, el histograma se puede aproximar mediante una curva suave. La curva simétrica, en forma de campana que se sobrepone en la figura 3.4, es una de estas formas características (la *distribución normal*). Dadas suficientes mediciones, en este caso particular el histograma se aproximará a la distribución normal.

3.3 Técnicas de suavizado de curvas

Cuando existe un gran número de datos

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots (x_k, y_k), (x_n, y_n)$$

tal que $x_\beta \neq x_j \ (i \neq j) \ i, j = 0,2, \dots, n.$

Se han de buscar otros valores, que por la medición no se pueden determinar o deducir datos hacia el fututo. A continuación se explican los métodos los que mejor se ajustan a nuestro problema en cuestión.

A continuación se explican las técnicas más importantes.

3.3.1 Regresión por mínimos cuadrados

Cuando los datos tienen errores sustanciales, la interpolación polinomial es inapropiada y puede dar resultados poco satisfactorios cuando se utiliza para predecir valores intermedios, como es el caso que nos ocupa. Los datos recogidos experimentalmente en algunas ocasiones presentan una variabilidad significativa. Por lo tanto tendremos que aplicar un suavizado de la curva para obtener un resultado fiable.

3.3.1.1 Regresión lineal

Se trata del ejemplo mas simple de aproximación por mínimos cuadrados. Consiste en trazar una línea recta a un conjunto de observaciones definidas por puntos: (x_1, y_1), (x_2, y_2),..., (x_n, y_n). La expresión matemática para la línea recta es

$$y = a_0 + a_1 x + e \qquad \text{(III-5)}$$

Donde a_0 y a_1 son coeficientes que representan la intersección con el eje "*y*" y la pendiente, respectivamente, *e* es el error, o diferencia, entre el modelo y las observaciones, el cual se representa al reordenar la ecuación:

$$e = y - a_0 - a_1 x \qquad \text{(III-6)}$$

Así, el *error* es la discrepancia entre el valor verdadero de *y* y el valor aproximado, $a_0 + a_1 x$, que predijo la ecuación lineal.

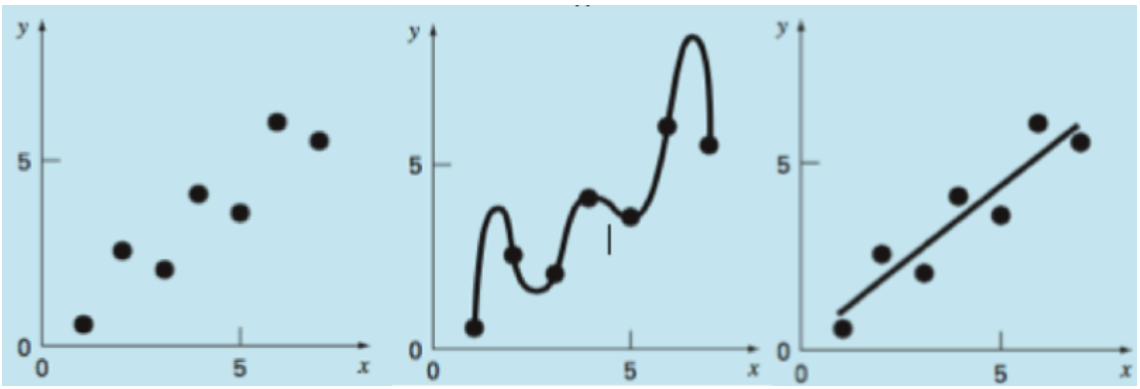

Figura 3.5.- Distintos ajustes de un conjunto de datos

La ilustración 3.5 muestra como de un conjunto de datos se realizan dos ajustes totalmente distintos, para tener una visión del error significativo:

- Primero se hace un ajuste polinomial oscilando mas allá del rango de los datos
- La tercera gráfica muestra un ajuste por mínimos cuadrados, cuyo resultado es el mas satisfactorio.

3.3.2 Regresión polinomial

Cuando los datos a representar no se ajustan al patrón de una línea recta, quizás una curva sea mas representativa.

Para ello se utiliza la regresión polinomial para ajustar polinomios. El procedimiento de mínimos cuadrados se puede extender fácilmente al ajuste de datos con un polinomio de grado superior.

Determinar un polinomio de grado dos por mínimos cuadrados es equivalente a resolver un sistema de tres ecuaciones lineales simultáneas.

El caso bidimensional se extiende con facilidad a un polinomio de *m*-ésimo grado como sigue:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m + e \quad \text{(IV-7)}$$

En la imagen 3.6 se muestra un ejemplo de aproximación por regresión polinomial de grado dos.

.

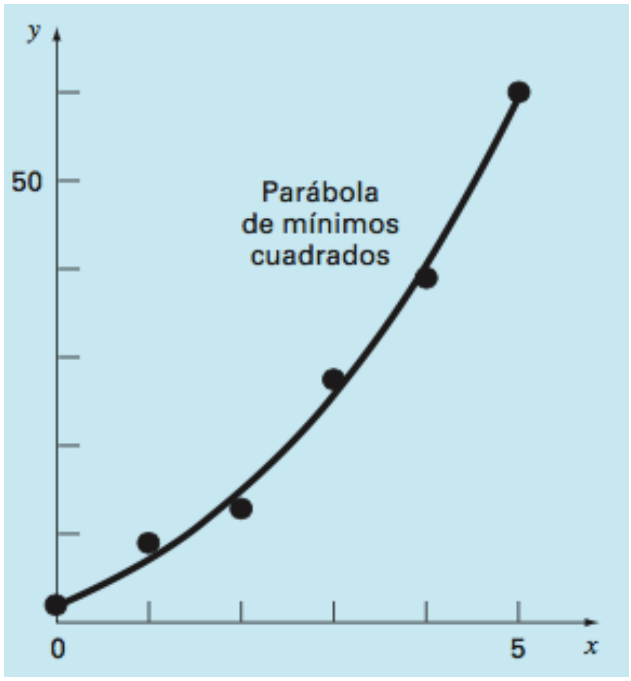

Figura 3.6.- Curva de polinomio de grado dos

Resolver un polinomio de *m*-ésimo grado es simple utilizando un soporte informático como Matlab, pero plantea un problema potencial en su implementación. Las ecuaciones normales algunas veces están mal condicionadas. Esto se presenta especialmente cuando se plantean polinomios de grado superior. En tales casos, los coeficientes calculados pueden ser altamente susceptibles al error de redondeo y, en consecuencia, los resultados serían inexactos. Entre otras cuestiones, este problema se relaciona con la estructura de las ecuaciones normales y con el hecho de que con polinomios de grado superior las ecuaciones normales pueden tener coeficientes muy grandes y muy pequeños. Lo anterior se debe a que los coeficientes son sumas de datos elevados a potencias. Lo bueno es que la mayoría de los problemas prácticos están limitados a polinomios de grado inferior, en los cuales el error de redondeo suele ser muy pequeño.

3.3.3 Suavizado por spline

Una *función spline* está formada por varios polinomios, cada uno definido sobre un subintervalo, que se unen entre sí obedeciendo a ciertas condiciones de continuidad.

Si que tenemos n+1 puntos, los cuales se denominan nudos, tales que $t_0<t_1\cdots<t_n$. Dando por hecho que se ha fijado un entero $k \geq 0$.

una función spline de grado *k* con nudos en $t_0,t_1,\text{-},t_n$ es una función S que satisface las condiciones:

i. en cada intervalo (t_{i-1},t_i), *S* es un polinomio de grado menor o igual a *k*.
ii. *S* tiene una derivada de orden (*k*-1) continua en $[t_0,t_n]$.

Los splines de grado 0 son funciones constantes por zonas. La figura 3.7 muestra el spline correspondiente a grado 0.

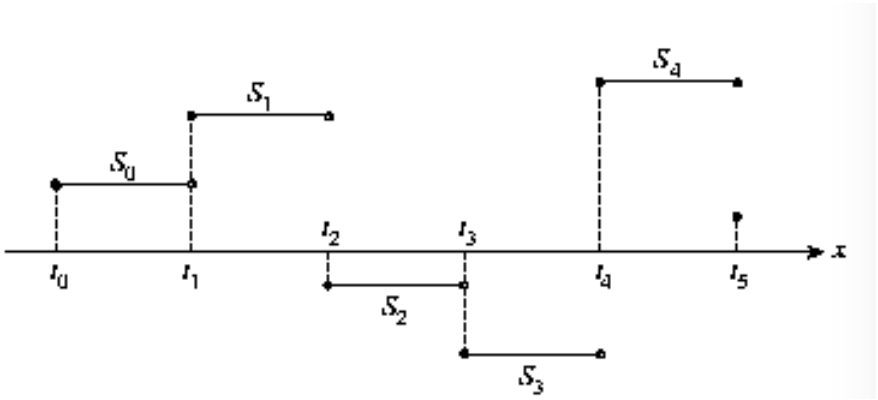

Figura 3.7.- Spline de grado 1

3.3.4 Suavizado por spline cúbico

Los autores Richard L. Burden y J.Douglas Faires en el libro Numerical Analysis(2010), nos dice que un spline cúbico consiste en dividir el intervalo en una serie de subintervalos, y en cada subintervalo construir un polinomio diferente de aproximación. A esta forma de aproximar por medio de funciones se le conoce como aproximación polinómica fragmentaria.

La aproximación polinómica fragmentaria es la interpolación lineal fragmentaria que consiste en unir una serie de puntos de datos

$$\{(x_0f(x_0)), (x_1f(x_1)),..., (x_nf(x_n))\}$$

mediante una serie de fragmentos de rectas como los que parecen en la figura 3.8.

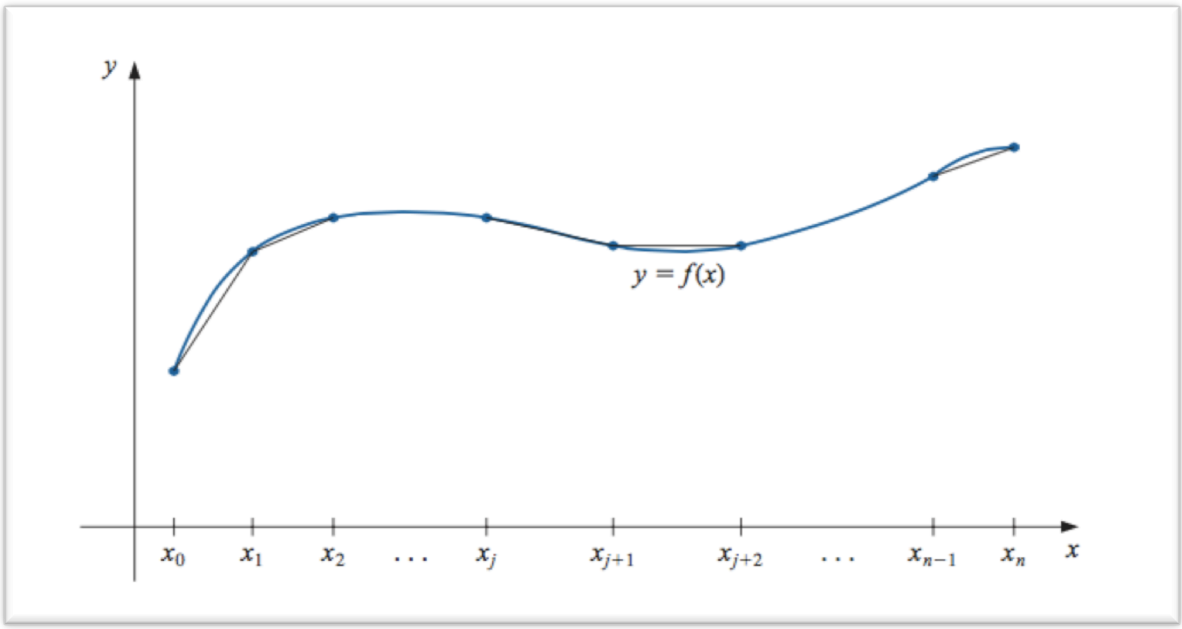

Figura 3.8.-Aproximación Spline cúbico

La desventaja de la aproximación por funciones lineales es que no se tiene la seguridad de que haya diferenciabilidad en los extremos de los subintervalos, lo cual dentro de un contexto geométrico significa que la función interpolante no es “suave” en dichos puntos.

La aproximación polinómica fragmentaria más común utiliza polinomios entre cada par consecutivo de nodos y recibe el nombre de spline cúbico. Un polinomio cúbico general contiene cuatro constantes; así pues, el procedimiento del trazador cúbico ofrece suficiente flexibilidad para garantizar que el interpolante no sólo sea continuamente diferenciable en el intervalo, sino que además tenga una segunda derivada continua en el intervalo. Sin embargo, como se muestra en la figura 3.9, en la construcción del spline cúbico no se supone que las derivadas del interpolante concuerdan con las de la función, ni siquiera en los nodos.

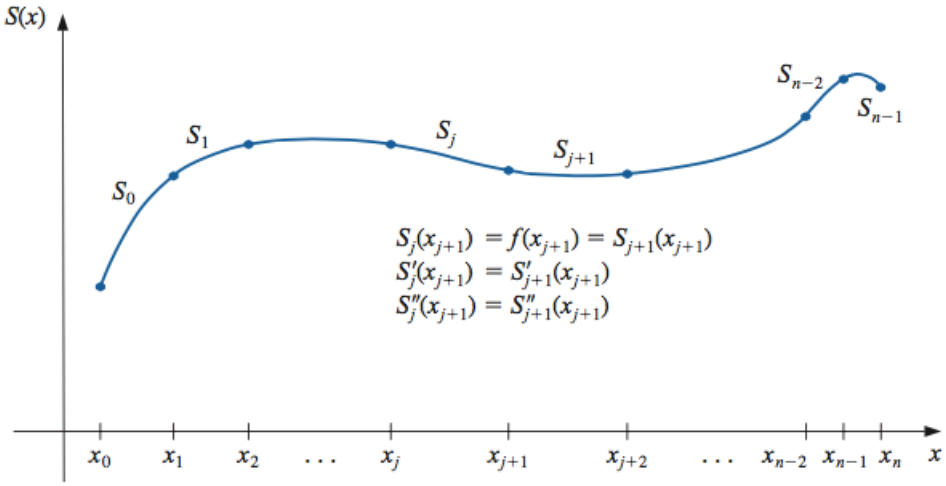

Figura 3.9.-Construcción de un spline cúbico

Definición: Dada una función f definida en [a, b] y un conjunto de nodos $a=x_0<x_1<\text{-}<x_n=b$, un spline cúbico interpolante S para f es una función que cumple con las condiciones siguientes:

1. S(x) es un polinomio cúbico, denotado por $S_j(x)$, en el subintervalo $[x_j,x_{j+1}]$ para cada j = 0, 1, ... , n-1;
2. $S(x_j)=f(x_j)$para cada j=0,1,...,n;
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ para cada j = 0, 1, ... , n-2;
4. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ para cada j = 0, 1, ... , n-2;
5. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ para cada j = 0, 1, ... , n-2;
6. Una de las siguientes condiciones de frontera se satisface:

   i. $S''(x_0) = S''(x_n) = 0$ (frontera libre o natural);

   ii. $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$ (frontera sujeta).

Aunque los splines cúbicos se definen con otras condiciones de frontera, las condiciones dadas en (f) son suficientes en este caso. Cuando se presentan las condiciones de frontera libre, el spline recibe el nombre de spline natural, y su gráfica se aproxima a la forma que adoptaría una varilla larga y flexible si la hiciéramos pasar por los puntos $\{(x_0f(x_0)), (x_1f(x_1)),..., (x_nf(x_n))\}$.

En términos generales, en las condiciones de frontera sujeta se logran aproximaciones más exactas, ya que abarcan más información acerca de la función. Pero para que se cumpla este tipo de condición de frontera, se

requiere tener los valores de la derivada en los extremos o bien una aproximación precisa de ellos.

Si queremos construir el spline cúbico interpolante de determinada función f, aplicamos las condiciones de la definición a los polinomios cúbicos:

$$S_j(x)=a_j+b_j+(x-x_j)+c_j+(x-x_j)^2+d_j+(x-x_j)^3$$

para cada j = 0, 1, … , n-1.

Está claro que

$$S_j(x_j)=a_j=f(x_j)$$

y si se aplica la condición (c)

$$a_{j+1}=S_{j+1}(x_{j+1})=S_j(x_{j+1})=a_j+b_j(x_{j+1}-x_j)+c_j(x_{j+1}-x_j)^2+d_j(x_{j+1}-x_j)^3$$

para cada j = 0, 1, … , n-2.

Puesto que los términos $x_{j+1}-x_j$ se utilizarán varias veces en este desarrollo, conviene introducir la notación más simple

$$H_j=x_{j+1}-x_j$$

para cada j = 0, 1, … , n-1.

Si también definimos $a_n=f(x_n)$,entonces la ecuación

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \qquad \text{(III-8)}$$

será válida para cada j = 0, 1, … , n-1.

De manera análoga, defina $b_n = S'(x_n)$ y observe que

$$S'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

significa que $S'_j(x)=b_j$ para cada j = 0, 1, … , n-1. Al aplicar la condición (d) obtenemos

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2 + d_j h_j^3 \qquad \text{(III–9)}$$

para cada j = 0, 1, … , n-1.

Al definir $c_n = S''(x_n)/2$ y aplicar la condición (e), se obtiene otra relación entre los coeficientes de Sj En este caso, para cada j = 0, 1, … , n-1,

$$c_{j+1} = c_j + 3d_j h_j \qquad \text{(III-10)}$$

Al despejar d_j en la ecuación (III-10) y sustituir este valor en las ecuaciones (III-8) y (III-9) para cada j = 0, 1, … , n-1 se obtienen las ecuaciones

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \qquad \text{(III-11)}$$

y

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \qquad \text{(III-12)}$$

La relación final que incluye los coeficientes se obtiene resolviendo la ecuación correspondiente en la forma de la ecuación (III-11), primero para b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_{j-1}}{3}(2c_j + c_{j+1}) \qquad \text{(III-13)}$$

y entonces, con una reducción del índice, para b_{j-1}:

$$b_{j-1} = \frac{1}{h_j - 1}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) \qquad \text{(III-14)}$$

Cuando sustituimos estos valores en la ecuación obtenida de la ecuación (III-13), con el índice reducido en 1, obtenemos el sistema de ecuaciones lineales

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \text{ (III-15)}$$

para cada j = 0, 1, … , n-1. Este sistema contiene sólo $\{c_j\}_{j=0}^{n}$ como incógnitas, ya que los valores de $\{h_j\}_{j=0}^{n-1}$ y de $\{a_j\}_{j=0}^{n}$ están dados por el espaciado de los nodos $\{x_j\}_{j=0}^{n}$y los valores de f en éstos.

Nótese que una vez que se conocen los valores de $\{c_j\}_{j=0}^{n}$, encontrar el resto de las constantes $\{b_j\}_{j=0}^{n-1}$ partiendo de la ecuación (III-13) y $\{d_j\}_{j=0}^{n-1}$ de la ecuación (III-10) para construir los polinomios cúbicos $\{S_j(x)\}_{j=0}^{n-1}$ es fácil.

Capítulo 4

Teoría Sobre Redes Neuronales Artificiales

4 Introducción

Las Redes Neuronales Artificiales tienen una gran importancia en la predicción de las curvas de demanda eléctrica, puesto a lo largo del tiempo han demostrado ser un modelo que se aproxima mucho a la realidad. Se empezaron a utilizar debido a que los modelos estadísticos requerían un gran conocimiento de las relaciones entre variables y altos costos computacionales.

Su uso consiste en aproximar una función no lineal a unos datos históricos, partiendo desde un concepto totalmente equivalente al ajuste de funciones mediante métodos regresión como pueden ser los mínimos cuadrados. La gran diferencia con los modelos de regresión es que estos, determinan los parámetros de una función no lineal determinada (se conoce el tipo de linealidad, pero se desconocen los parámetros que la concretan), mientras que las redes neuronales artificiales determinan completamente la forma de la no linealidad.

Las entradas a esta función no lineal serán los valores pasados de la demanda, además de otras variables externas. Entrenando la RNA correctamente se consigue un ajuste de los parámetros capaz de hacer que dicha función consiga las características de la demanda eléctrica, simplemente a través de datos históricos, sin necesidad de que una persona experta cuantifique el efecto que tiene cada variable sobre la demanda eléctrica.

El proceso de selección de estas entradas es una parte muy importante, aquí englobaríamos variables como la velocidad del viento, la nubosidad, la pluviosidad, la temperatura, etc. Sin embargo nos encontramos con que no todos los factores climáticos afectan a la demanda eléctrica. Algunos de ellos son típicamente aleatorios y otros aparecen interrelacionados. Por ejemplo, la temperatura viene explicada parcialmente por la nubosidad, la pluviometría, la humedad relativa, etc. Entre todos estos factores la temperatura es el más relevante, dado influye de forma directa sobre múltiples fuentes de consumo eléctrico como sistemas de calefacción, aire acondicionado, refrigeradores, etc. Como primera parte del estudio, la temperatura será la variable que introduciremos en nuestra red, dado que la inmensa mayoría de los trabajos consultados, señalan esta variable como principal determinante meteorológica de la demanda eléctrica. En nuestra base de datos también hemos recogido la humedad relativa y la luminosidad.

En cuanto a estas dos últimas, debemos señalar que si bien existen discrepancias, nosotros hemos optado por incluir la humedad relativa diaria

en la segunda parte del estudio, con el fin de demostrar si es válida, para contar con un modelo lo más completo posible. Posteriormente, atendiendo a los resultados obtenidos en el entrenamiento y en la predicción nos podremos plantear su eliminación.

4.1 Definición y características

Una RNA puede definirse como un sistema de procesamiento de información compuesto por un gran número de elementos de procesamiento (neuronas), profusamente conectados entre sí a través de canales de comunicación. Estas conexiones establecen una estructura jerárquica y permiten la interacción con los objetos del mundo real tratando de emular al sistema nervioso biológico. A diferencia de la programación tradicional, basada en algoritmos predecibles, la programación neuronal permite desarrollar sistemas que resuelvan problemas complejos cuya formalización matemática es sumamente difícil. Esto se logra gracias a los principios de funcionamiento de las RNA, de los cuales citamos a continuación los cinco más importantes (Hilera JR., 1995):

- Aprendizaje adaptativo: esta es quizás la característica más importante de las RNA, ya que pueden comportarse en función de un entrenamiento con una serie de ejemplos ilustrativos. De esta forma, no es necesario elaborar un modelo a priori, ni establecer funciones probabilísticas. Una RNA es adaptativa porque puede modificarse constantemente con el fin de adaptarse a nuevas condiciones de trabajo.
- Generalización: mientras que el aprendizaje es un proceso donde se modifica la información interna de la RNA, la generalización consiste en la modificación de toda la red completa con el fin de llevar a cabo un objetivo específico. Una red puede responder a datos o situaciones que no ha experimentado antes, pero que puede inferir en base a su entrenamiento. Esta característica es muy útil sobre todo cuando la información de entrada es poco clara o se encuentra incompleta.
- Tolerancia a fallos: en la informática tradicional la perdida de un fragmento pequeño de información puede acarrear comúnmente la inutilización del sistema. Las RNA poseen una alta capacidad de tolerancia a fallos. La tolerancia a fallos se entiende aquí en dos sentidos: primero, las redes pueden reconocer patrones de información con ruido, distorsión o incompletos (tolerancia de fallos respecto de los datos); y segundo, pueden seguir trabajando (con cierta degradación) aunque se destruya parte de la red

(tolerancia a fallos respecto de la estructura). La explicación de este fenómeno se encuentra en que, mientras la informática tradicional almacena la información en espacios únicos, localizados y direccionales, las redes neuronales lo hacen de forma distribuida y con un alto grado de redundancia.

- Operación en tiempo real: de todos los métodos existentes, la RNA son las más indicadas para el reconocimiento de patrones en tiempo real, debido a que trabajan en paralelo actualizando todas sus instancias simultáneamente. Es importante destacar que esta característica solo se aprecia cuando se implementan redes con hardware especialmente diseñado para el procesamiento en paralelo.
- Fácil inserción en la tecnología existente: es relativamente sencillo obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilita la integración modular en los sistemas existentes. Esto ya no es tan importante como hace algunos años, puesto que la potencia de cálculo de los procesadores actuales suele ser suficiente. Por supuesto que en muchos casos si que será necesario un hardware específico, pero no para el proyecto que nos ocupa.

4.2 Partes de una Red Neuronal Artificial

Una RNA está constituida por nodos, que están unidas mediante conexiones. Algunas de las unidades están conectadas al exterior y se designan como neuronas de entrada o de salida según reciban estímulos externos o den la respuesta del sistema, respectivamente. Hay otras neuronas que simplemente están conectadas a otras neuronas del sistema y que al no tener relación directa con la información de entrada ni con la de salida del sistema, se denominan como neuronas ocultas. Se conoce como capa o nivel a un conjunto de neuronas cuyas entradas provienen de la misma fuente (que puede ser otra capa de neuronas) y cuyas salidas se dirigen al mismo destino (que puede ser otra capa de neuronas).

A continuación se explican mas detenidamente todos los elementos.

4.2.1 Neurona artificial.

Son elementos de cálculo simples que proporcionan una respuesta única (salida) a partir de un vector de entrada procedente del exterior. Las podemos dividir en tres grupos diferentes:

- Las que reciben la información del exterior, también denominadas como neuronas de entradas.
- Las que no tienen ningún contacto con el exterior y solamente intercambian información con otras neuronas de la red, a estas se las llama neuronas ocultas.
- Las que transmiten información al exterior, denominadas neuronas de salida.

En cualquier tipo de RNA las neuronas se encuentran fuertemente interconectadas entre sí, organizándose por capas y formando diferentes topologías. Existen diferentes topologías para resolver diferentes tipos de problemas. En la imagen 4.1 se muestra un esquema típico de RNA.

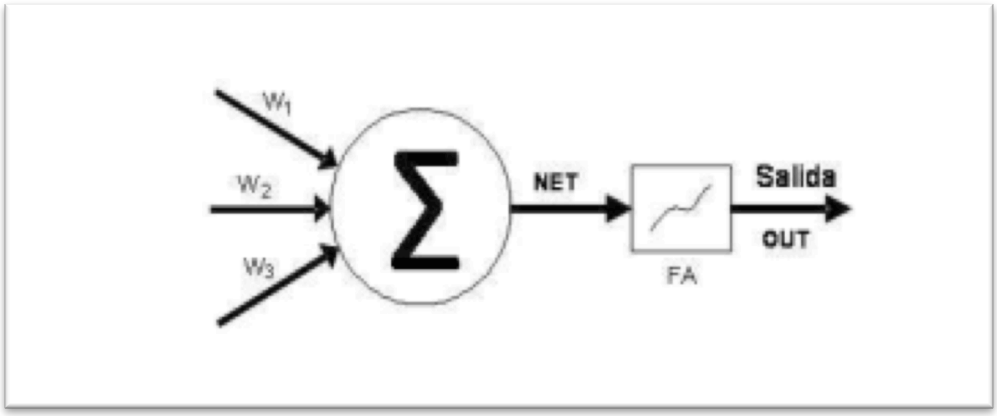

Figura 4.1-Esquema básico de una RNA

4.2.2 Unidad de proceso

Como se muestra en la imagen 4.2 la distribución de neuronas dentro de la red se realiza formando distintas capas, con un numero determinado en cada una de ellas. En función de su ubicación en la red distinguimos tres tipos de capas:

- Capa de entrada: Recibe directamente la información que viene del exterior de la red (entradas). Aquí están las denominadas neuronas de entrada.
- Capas ocultas: Se ubican entre la capa de entrada y la de salida. Su numero es indeterminado. Las neuronas de las capas ocultas pueden estar conectadas de múltiples maneras, y eso es lo que determina, junto a su numero, los distintos tipo de RN. Aquí se encuentran las neuronas ocultas.

- Capa de salida: Capa donde se transfiere la información de la red al exterior. Donde estarían las neuronas de salida.

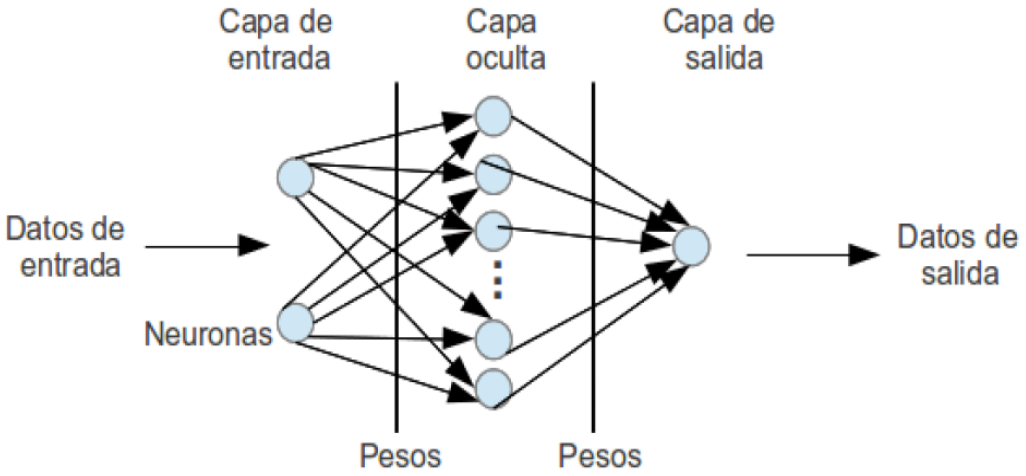

Figura 4.2.- Esquema de una RNA con una capa oculta

4.3 Estructura de una Red Neuronal Artificial

Una RNA se compone de la unión de varias neuronas de un determinado modo. Existen multitud de tipos de RNA, que dependiendo del autor tienen distintas formas de clasificarlas. Yo me ceñiré a los siguientes tres criterios, y a continuación se realiza el desarrollo de cada punto:

- Numero de capas
- Grado de conexión
- Tipo de conexión

4.3.1 Numero de capas

En función de la cantidad de capas de la RNA esta se clasifica de la siguiente manera:

- Redes Neuronales monocapas: Es la RN mas sencilla ya que tiene una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los cálculos.
- Redes Neuronales multicapa: Se podría decir que es una generalización de lo anterior. Es el caso más común, y como se observa en la imagen 4.3 existe una o varias capas intermedias entre la de entrada y salida llamada capa oculta. El numero de neuronas de cada nivel puede variar desde 1 hasta n, y dependerá de la tipología de cada modelo en particular.

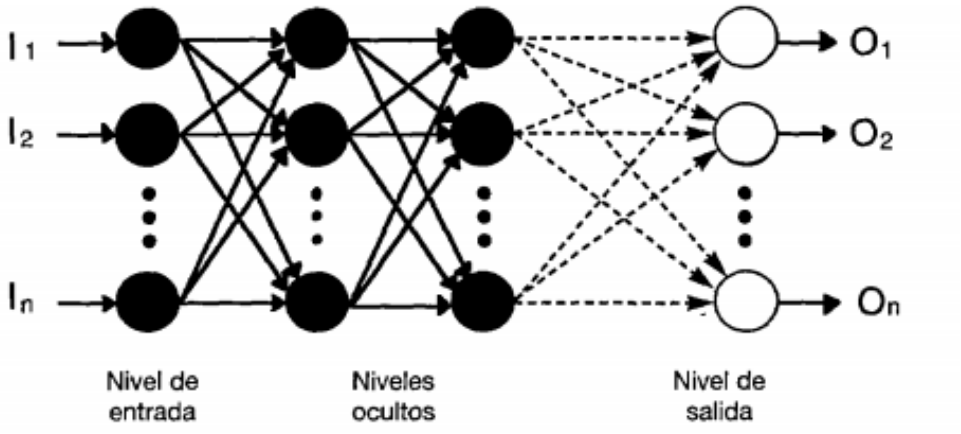

Figura 4.3.-Niveles de una RNA (Hilera JR.,1995)

4.3.2 Grado de conexión

Las conexiones de las neuronas de una RNA no siempre están todas conectadas entre sí, y se dividen en dos grupos, RNA completamente conectadas y RNA parcialmente conectadas, y se detallan a continuación.

- Redes neuronales completamente conectadas: Todas las neuronas de una capa se encuentran conectadas con las de la capa siguiente (redes no recurrentes) o con las de la anterior(redes recurrentes).
- Redes parcialmente conectadas: No se da una conexión total entre las neuronas de distintas capas.

4.3.3 Tipo de conexión

A continuación se detallan los dos tipos de RNA que es posible encontrar.

- Redes neuronales no recurrentes: En esta red la propagación de las señales se produce en un sentido solamente, no existiendo la posibilidad de realimentaciones. Lógicamente estas estructuras no tienen memoria.
- Redes neuronales recurrentes: Esta red viene caracterizada por la existencia de lazos de realimentación. Estos lazos pueden ser entre neuronas de diferentes capas, neuronas de la misma capa o, más sencillamente, entre una misma neurona. Esta estructura estudia principalmente la dinámica de sistemas no lineales.

4.4 Características de una RNA

Las RNA se caracterizan por tres partes fundamentales:

- Tipología de la red
- Regla de aprendizaje
- Tipo de entrenamiento

Inconvenientes de una RNA:

- Si esta es demasiado grande no es posible seguir paso a paso el razonamiento que les ha llevado a esas conclusiones.

- El problema es la limitación del hardware. La capacidad de las RNA radica en procesar una gran cantidad de datos simultáneamente, por ello necesitamos equipos muy potentes. Este problema deja a veces las RNA fuera de las soluciones viables de un problema.

4.5 Tipos de Redes Neuronales Artificiales

En la actualidad existen multitud de tipos de RRNNAA, por lo que en este TFG me centraré en los modelos más importantes y a su vez los que más se ajustan al requerimiento de nuestro problema. A continuación se exponen detalladamente.

4.5.1 RNA tipo perceptrón multicapa (MLP)

El perceptrón multicapa cuenta con una capa de entrada, al menos una capa oculta y una capa de salida.

Se trata de una estructura que propaga la señal hacia la salida. Las conexiones entre las neuronas se denominan pesos sinápticos, y son optimizados por el algoritmo de aprendizaje.

Esta arquitectura tiene como características fundamentales las siguientes:

- Es una estructura altamente no lineal.
- Alta tolerancia a los fallos
- Este sistema es capaz de establecer una relación entre dos conjuntos de datos.

- Se puede hacer una implementación hardware.

La propagación se realiza de manera que cada neurona hace una combinación lineal de las señales procedentes de las neuronas de la capa anterior siendo los coeficientes de esta combinación los pesos sinápticos. Posteriormente, se aplica la función de activación no lineal.

4.5.2 Backpropagation

El algoritmo backpropagation, o propagación hacia atrás, es el elegido para nuestra red neuronal. A continuación se exponen las principales razones por las que nos hemos decantado por este algoritmo:

- Eficacia
- Capacidad para adaptarse a diferentes problemas.
- Sencillez en los razonamientos empleados
- Independencia respecto a las condiciones iniciales
- Bajo coste de computación
- Alta capacidad de generalización

Es un algoritmo de descenso por gradiente que retropropaga las señales desde la capa de salida hasta la capa de entrada, optimizando los valores de los pesos sinápticos, mediante un proceso iterativo que se basa en la minimización del error. Debido a esto, el algoritmo se puede dividir en dos fases:

- Propagación hacia delante: Las señales se propagan desde la capa de entrada hasta la de salida, determinándose la salida de la red y el error cometido al comparar esta con el valor de la salida deseada que se facilita a la red durante la etapa de aprendizaje.
- Propagación hacia atrás: En función de los errores cometidos en la capa de salida, el algoritmo se encarga de optimizar los valores de los pesos sinápticos que determinan las conexiones entre las neuronas mediante la retropropagación del error desde la capa de salida a la de entrada a través de las sucesivas capas ocultas.

Para la descripción de este algoritmo se tiene en cuenta el esquema de una neurona y el error como la diferencia existente entre la salida deseada y la salida observada. En caso de tener mas capas, se realizará la retropropagación del error durante un mayor número de capas.

En este proceso, los primeros pesos en actualizarse son los de la capa de salida.

El sistema de funcionamiento de este tipo de red, consiste en ir comparando el error obtenido de comparar el resultado deseado, del obtenido. Esto se puede hacer minimizando una función monótona creciente del error. En nuestro caso, se ha utilizado el error relativo.

Existen dos posibilidades para realizar el entrenamiento:

- On-line: Este aprendizaje se realiza patrón a patrón. Durante todo el entrenamiento se pasa a la red cada entrada junto con la salida deseada. A partir de aquí, se mide el error y en función de este se adaptan los pesos sinápticos mediante el algoritmo de aprendizaje escogido.
- Batch: Es un aprendizaje por lotes. Un lote supone el paso de todos los patrones de entrenamiento por la red. Se le pasa a la red todos los patrones de entrenamiento, se mira el error total cometido y se adaptan los pesos en función de ese valor del error promediado según el número de patrones[Haykin S.]. Este es el que hemos utilizado para nuestro estudio. La elección se basó en que la curva de carga es periódica, por lo tanto en cada lote tendremos la tendencia de la red.

La actualización de los pesos depende de la señal de error $e_p(t)$.Esta señal tiene un tratamiento distinto en función de la capa a la que pertenezcan las neuronas.

4.5.3 RNA tipo cascada correlación (CASCOR)

La red neuronal artificial tipo cascada correlación (CASCOR) presenta ventajas conceptuales interesantes en relación con el problema de estimación del MLP, aunque puede sufrir de sobreajuste.

CASCOR está diseñada bajo el esquema de crecimiento en el tamaño de la red o aprendizaje constructivo, donde no es necesario de conocer a priori el número de neuronas ocultas requeridas, de modo que el aprendizaje puede ser más rápido y puede tener una mejor capacidad de generalización que un MLP.

Las redes neuronales CASCOR se caracterizan principalmente por su aprendizaje constructivo, en el cual se comienza con una red mínima, sin neuronas en la capa oculta, luego se adopta un proceso constructivo donde se van añadiendo neuronas, evaluando y optimizando en cada paso el error global de la red. En el proceso de adición de neuronas ocultas a la red, cada neurona recibe una nueva conexión sináptica de cada entrada y de cada neurona oculta que la precede. Después de la adición de la nueva neurona oculta, los pesos sinápticos de las entradas se congelan, mientras que sus pesos de salida son optimizados. Este proceso es continuo hasta que se alcanza un rendimiento satisfactorio. La imagen 4.4 muestra el esquema de una red CASCOR; las cajas en las intersecciones de las líneas indican los pesos que se congelan (parámetros) una vez que se ha añadido una unidad en la capa oculta. Los cruces indican los pesos que son modificados después de la inserción de la neurona.

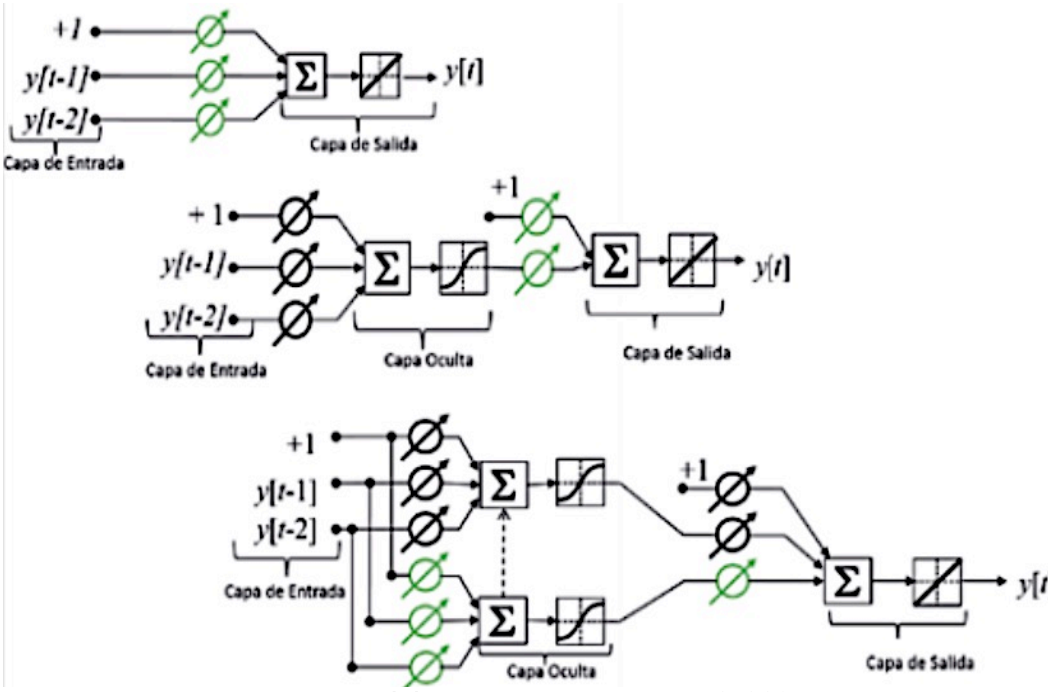

Figura 4.4.-Imagen de una RNA tipo CASCOR

4.6 Ajuste de una RNA

Una vez elegido el tipo de red que mejor se ajusta al problema a resolver, se pasa a la segunda fase, que consiste en darla forma ajustando su estructura. Para ello se debe hacer un estudio, ya que una mala elección de alguna de estas posibilidades podría hacer que la RNA no fuera lo suficientemente buena.

4.6.1 Numero de neuronas

Tenemos muchas posibilidades. El número de neuronas que forman las capas de entrada y salida viene determinado por el problema, puesto que va a depender del número de entradas y las salidas que queramos obtener de nuestra RN. Sin embargo, el numero de capas ocultas y de neuronas en cada una de ellas no está fijado ni por el problema, ni por ninguna regla teórica, por lo que el diseñador de la red será quien decida esta arquitectura en función de la aplicación que vaya a hacer de esta tras analizar los valores de estos dos factores. Lo único que se ha conseguido demostrar hasta ahora, es que dado un conjunto de datos conexo, con una sola capa oculta es posible establecer una relación en el conjunto de datos, sin especificar el numero de neuronas necesarias. Según Kolmogorov, si el conjunto no es conexo, harán falta al menos dos capas ocultas.

Todo esto, nos puede llevar a pensar que con un mayor número de capas y neuronas intermedias, obtendremos un mejor resultado. Pero esto tiene una serie de problemas:

- Aumentará la carga computacional, que automáticamente implica mayor implementación en tiempo real y un crecimiento en el tiempo de aprendizaje por pate de la red.
- Se perderá capacidad de generalización. Al aumentar el número de neuronas en la capa oculta, aumenta el número de pesos sinápticos, por lo que la red estará caracterizada por mas parámetros. Esto tiene la ventaja de proporcionar una mejor modelización de los patrones utilizados, pero se pierde capacidad de generalización, puesto que ante nuevos datos, la red no se adaptará a estos y obtendremos un peor resultado.

4.7 Función de activación

Es la función no lineal que se aplica a la salida de la neurona. Debe de cumplir solamente dos condiciones: Ser continua y diferenciable. Estas son las tres funciones mas utilizadas:

4.7.1 Signo

Fue la primera que se utilizó. Esta definida por:

$$sign(x) = \begin{cases} 1\ si\ x > 0 \\ -1\ si\ x \leq\ 0 \end{cases}$$

Debido a sus malas propiedades matemáticas(no es diferenciable en 0) se dejó de utilizar. Las dos siguientes derivan de esta función.

4.7.2 Sigmoide

Esta será la que se utilizará en la RNA propuesta para el estudio debido a su buen comportamiento en valores que tendremos en nuestras variables de entrada. Como se ve tanto en la ecuación IV-1 como en la imagen 4.5, toma valores entre 0 y 1 para una variación de la variable independiente x entre $-\infty\, y\, +\infty$.

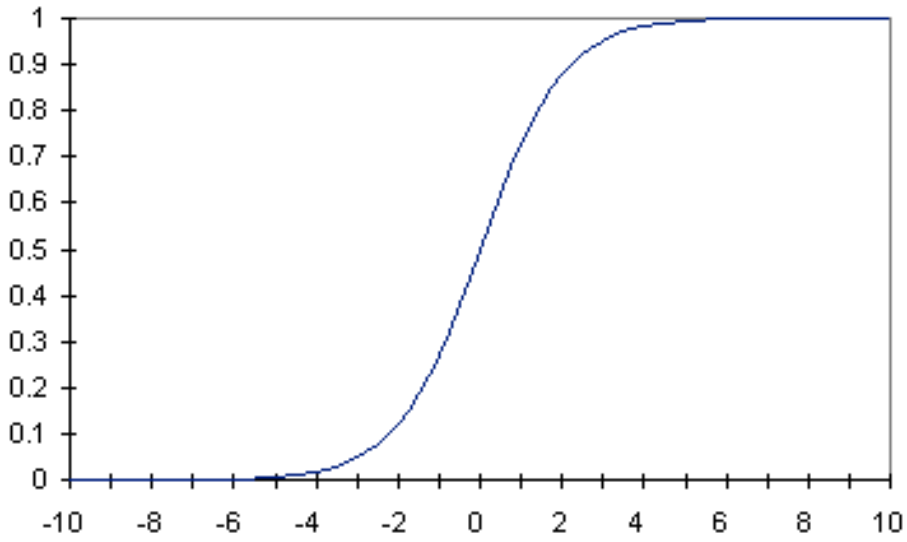

Figura 4.5. Curva sigmoide

$$f_1(x) = \frac{1}{1+e^{-x}} \qquad \text{(IV-1)}$$

4.7.3 Tangente hiperbólica

Al igual que en la función sigmoide, esta toma valores entre -1 y +1 para una variable independiente entre $-\infty\, y\, +\infty$. Como se observa en la ecuación IV-2, en este caso, con -1 se codificará la mínima actividad de la neurona, y con+1 la máxima.

$$f_2(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad \text{(IV-2)}$$

4.7.4 Función lineal a tramos

Es una función mas sencilla que las anteriores. Está formada por tres tramos lineales unidos para formar una función no lineal. Suele tener valores entre -1 y +1, como se muestra en la ecuación IV.3. De esta forma quedará mas sencilla, aunque también se puede considerar una función que varié entre 0 y 1, aunque esto segundo es menos habitual.

$$f(x) = \begin{cases} -1 \; si \; x < -1 \\ x \;\; si \;\; -1 < x < 1 \\ 1 \; si \; x < 1 \end{cases} \qquad \text{(IV-3)}$$

Capítulo 5

Desarrollo de la Red Neuronal Artificial con Matlab

5. Introducción

A continuación se explican los scripts y funciones de Matlab que se han creado para el desarrollo del TFG. Primero se explicarán las funciones mas pequeñas, que se utilizan para el tratamiento de datos para el posterior uso en la red neuronal.

Se mostraran los scripts tal como se ven el programa. En las funciones ya programas en Matlab se explicará su funcionamiento y las razones de su utilización.

5.1 Funciones creadas para el TFG

5.1.1 Función "spline_cubico"

Como vemos en la figura 5.1, en esta función se hace un suavizado de los datos de potencia demandada.

```
1
2     %              Función spline_cubico
3
4
5     %Una vez introducidos los datos de entrada en bruto, devuelve el spline
6     %cúbico de los datos de cada mes
7     % Los argumentos de entrada son el vector de potencias
8     % del mes a tratar y un parámetro de suavizado que determinará
9     % el peso relativo que queramos dar  a la aproximación.
10    % p puede tener valores entre 0 y 1
11
12
13
14
15    function spmes = spline_cubico(datos_mes,p)
16
17
18 -      xmes = datos_mes(:,1);
19 -      ymes = datos_mes(:,2);
20
21 -      spmes = csaps(xmes,ymes,p,xmes);
22
```

Figura 5.1.- Función spline cúbico

En esta función se utiliza la función de Matlab perteneciente a la toolbox llamada "Spline Toolbox".

El comando *'spmes = csaps(xmes,ymes,p,xmes)'* devuelve la forma polinómica a trozos (ppform) de un spline cúbico suavizado *'f'* para los datos

dados *'xmes, ymes'*, con el valor de *'f'* en el lugar de los datos *'xmes(j)'* aproximando el valor de los datos *'ymes(:, j)'*, para *'j=1:length(xmes)'*. Los valores pueden ser escalares, vectores o matrices. Los puntos de datos del mismo sitio se sustituirán por su media, ponderada por la suma de los pesos correspondientes. El parámetro *'p'* permite fijar el nivel de suavizado. El parámetro de suavización determina el peso relativo, según las exigencias contradictorias, de que *'f'* sea muy suavizado o de que *'f'* esté cerca de los datos. El último argumento de la función, *'xmes'*, es para obligar a la función a que nos devuelva los valores suavizados de *'f'* en los mismos valores de 'xmes', es decir, que nos devuelva *'f(xmes)'*.

5.1.2 Función "cambio_datos"

Como se observa en la función 5.2, con esta función se seleccionan los datos de demanda de las horas en punto en un vector con datos de la demanda cada 15 minutos.

```
1
2     %                 Función cambio_datos
3
4     % Transforma los excel con datos cada 15 min en vector con los datos de
5     % cada hora en punto con cinco filas más con las entradas de hora.
6     % El argumento de entradas será el vector de potencias ya ordenado del
7     % excel. La salida será un vector 4 veces más corto recogiendo los valores
8     % de cada hora.
9
10
11    function datos = cambio_datos(datos_mes)
12
13 -  n1=length(datos_mes);
14
15 -  k=0;
16
17 -  for i=1:4:n1
18
19 -      k=k+1;
20
21 -      m=rem(i,n1/4);
22
23 -      if m==0
24 -          break;
25 -      else
26 -          datos(k)=datos_mes(i);
27 -      end
28
29 -  end
30
31
```

Figura 5.2.- Función cambio_datos

Lo que hace esta función es recorrer el vector de cargas y seleccionar los datos de las horas enteras mediante un bucle "for".

Con la función rem obtenemos el resto de la división entera de sus argumentos, de esta forma el programa detecta el final del vector cargas y detiene el bucle “for”.

5.1.3 Función “extremos_temperatura”

Como se observa en la figura 5.3, se dispone de dos neuronas para introducir los extremos de temperatura, una será el valor mínimo diario, y la otra el máximo. Con esta función se crean los vectores correspondientes a esas dos variables de entrada.

```
1
2     %            Funcion extremos_temperatura
3     % Esta funcion toma la matriz de datos de la temperatura horaria
4     % y separa en un vector la temperatura máxima de cada día
5     % repetida 24 veces (24 horas al día) y en otro vector lo mismo con
6     % la temperatura minima.
7
8
9
10
11    function [Temp_Max, Temp_Min] = extremos_temperatura(Matriz_Temp)
12
13 -  [Filas, Columnas] = size(Matriz_Temp);
14
15 -  n=0;
16
17 -  for i=1:Filas
18 -      for j=1:Columnas
19 -          Temp_Max(j+Columnas*n) = max(Matriz_Temp(i,:));
20 -          Temp_Min(j+Columnas*n) = min(Matriz_Temp(i,:));
21 -      end
22 -      n=n+1;
23 -  end
```

Figura 5.3.- Función extremos_temperatura

Esta sencilla función recorre mediante un bucle “for” la matriz de datos de la temperatura, y se guarda en dos vectores distintos los valores máximos y mínimos diarios.

5.1.4 Función “findes”

Como se observa en la figura 5.4, esta función genera un vector (festivos), en el cual se incluye la información de que día es festivo, y cual no. Los días laborables tendrán un “1” como entrada, mientras que los festivos se marcarán con un “0”. Para poder realizar esta función es necesario tener la información que nos proporciona el vector de cargas horarias para conocer la longitud del mismo, así como el nombre del día de la semana que empieza el mes.

Utilizaremos caracteres en mayúscula o minúscula con la primera letra de cada día, excepto el miércoles, que se expresará con una “x”.

Con los primeros “if´s” asociamos cada día a un número, y en cada condición aparecen dos sentencias para aceptar tanto mayúsculas como minúsculas. Además nos informa por pantalla si el día introducido es valido.

Al final, la función imprime por pantalla las graficas del periodo de dato, diferenciando entre días laborables y festivos. El programa dibuja cuatro gráficas en un mismo plot.

```
%             Función findes

% Crea un vector de una fila y el mismo número de columnas que el vector
% de entrada, en el cual se escribirá un cero (0) si el día es sábado o
% domingo, y un uno (1) si no lo es.
% La entrada o input 1 de la función será el vector de potencias solamente
% con los datos horarios, el input 2 será el día de la semana en el que
% comienza el mes ('l' para lunes, 'm' para martes, 'x' para miércoles,
% etc.).
% Además esta función saca por pantalla las gráficas de los días
% según sean laborables o no, con el fin de comprobar si se ha realizado
% correctamente.

function festivo = findes(datos, day)


if (day=='L')||(day=='l')

sab=5;

else if (day=='M')||(day=='m')

     sab=4;

     else if (day=='X')||(day=='x')

          sab=3;

```

```
28
29 -            else if (day=='J')|||(day=='j')
30
31 -                 sab=2;
32
33 -                 else if (day=='V')|||(day=='v')
34
35 -                      sab=1;
36
37 -                      else if (day=='S')|||(day=='s')
38
39 -                           sab=0;
40
41 -                           else if (day=='D')|||(day=='d')
42
43 -                               sab=6;
44
45 -                               else
46
47 -                               fprintf('\nError. Input 2 no es correcto.\n');
48 -                               return;
49
50 -                               end
51 -                           end
52 -                      end
53 -                 end
54 -            end
55 -       end
56 - end
57
58 - n2=length(datos);
59
60 - festivo=ones(1,n2);
61
62
63 - corrector=24*sab;   %cambio de días a horas para llegar al sábado
64
65 - if sab==6
66 -     horas=0;
67 - else
68 -     horas=sab*24;   %cambio de días a horas para llegar al sábado
69 - end
70
71 - for i=(horas+1):24*7:n2
72
73 -     if sab==6           %es domingo
74
75 -         for m=0:23
76
77 -             festivo(i+m)=0;                %asignamos los ceros al domingo
78
79 -             if (i+m+corrector)<=n2         %comprobamos que no nos
80                                              %salimos del array y asignamos
81 -                 festivo(i+m+corrector)=0;  %un 0 al sábado siguiente
82
83 -             else end
84 -         end
85 -     else                    %resto de días
86
```

```
87 -            for m=0:23
88
89 -                festivo(i+m)=0;                %asignamos los ceros al sábado
90
91 -                if (i+m+24)<=n2                %Comprobamos que no nos
92                                                 %salimos del array y asignamos
93 -                    festivo(i+m+24)=0;         %un cero al domingo
94
95 -                else end
96
97 -            end
98 -        end
99
100 -   end
101
102
103     %%%% Plotear resultados
104
105 -   m=0;
106 -   h=0;
107
108 -   for i=1:n2
109
110 -       if festivo(i)==1
111
112 -           m=m+1;
113 -           dia_lab(m)=datos(i);
114
115 -       else
116
117 -           h=h+1;
118 -           dia_fes(h)=datos(i);
119
120 -       end
121
122 -   end
123
124 -   figure
125
126 -   subplot(2,2,1)
127 -   plot(dia_lab,'b');
128 -   title('Días laborables');
129
130
131 -   subplot(2,2,2)
132 -   plot(dia_fes,'r');
133 -   title('Días festivos');
134
135
136 -   subplot(2,2,3)
137 -   plot(dia_lab,'b');
138 -   title('Laborables+festivos');
139 -   hold on;
140 -   plot(dia_fes,'r');
141
142
143 -   subplot(2,2,4);
144 -   plot(datos,'g');
145 -   title('Completo');
```

Figura 5.4.- Función findes

5.1.5 Función “fiesta”

Como se observa en la figura 5.5, con esta función se asignan los días que son festivos distintos al fin de semana.

```
%                  Función fiesta

% Modifica el vector 'festivo' según sea la entrada por el teclado,
% cambiando un día laborable por un día festivo. Tiene dos inputs, uno será
% el vector 'festivo' con la información de qué días son festivos o no, y
% el otro será un vector o un único número con la información del o de los
% días que queremos cambiar a festivos. Este último input es opcional, si
% no se introduce se pedirá por pantalla.

function festivo_nuevo=fiesta(festivo, day)

n=length(festivo);

festivo_nuevo=festivo;

if nargin==1

   while(1)

   disp(' ');
   d=input('Introduzca día festivo (1...31) "q" para salir: ');

   if d=='q'

      return;

   else if d>(n/24)||d<=0

       fprintf('\nError. Introduzca un día que se encuentre en el mes.\n');

       else

       indice=(d-1)*24;    %pasa de días a horas y corrige el sesgo

           for i=1:24

               festivo_nuevo(i+indice)=0;
```

```
38
39 -            end
40
41
42
43 -        fprintf('\n¿Algún cambio más? "s" o "n"\n');
44 -        g=input('s/n: ');
45
46 -            if g~='s'
47
48 -                return
49
50 -            else end
51
52 -        end
53
54
55 -    end
56
57 -    end
58
59 - else
60
61 -    if day==0
62
63 -        return;
64
65 -    else if (max(day)>n/24)||(min(day)<0)
66
67 -        fprintf('\nError. Input 2 es erróneo.\n');
68
69
70 -    else
71
72 -        n2=length(day);
73
74 -        for k=1:n2
75
76 -            indice=(day(k)-1)*24;    %pasa de días a horas y corrige el sesgo
77
78 -            for i=1:24
79
80 -                festivo_nuevo(i+indice)=0;
81
82 -            end
83
84 -        end
85
86 -        end
87
88 -    end
89
90
91 - end
92
```

Figura 5.5.- Función fiesta

Esta función puede tener uno o dos argumentos, el mas común es el vector festivos que creamos con la función "findes". El segundo argumento será un vector con los días que son festivos en formato numérico. El segundo argumento será un vector con los días que son festivos en formato numérico. En el caso de que no se incluya, la propia función lo pedirá por pantalla.

5.1.6 Función "tiempo_bin"

Como se observa en la figura 5.6, con esta función generamos una matriz con la hora en binario para cada elemento de la demanda. Todos los vectores de demanda comienzan a las 00:00 horas del primer día, hasta las 23:00 horas del último día.

```
1   %                       Función tiempo_bin
2
3   % Crea los inputs de la hora para un determinado vector de datos de carga
4   % horaria, empezando por las 00:00h hasta las 23:00. Por tanto, nuestro
5   % primer elemento corresponde a las 00:00, el segundo a las 01:00... el
6   % elemento 24 corresponde a las 23:00... y así hasta el final.
7   % Con esta matriz concatenada con la de la variable del día y demás inputs
8   % formaremos nuestra matriz de datos.
9
10  function tiempo = tiempo_bin(datos)
11
12  n2=length(datos);
13
14  for i=1:n2
15
16      hora=rem(i,24); %resto de la hora que nos da conjuntos de 24 horas
17      if hora==0
18          hora_corregida=23;
19      else
20          hora_corregida=hora-1;
21      end
22
23      hora_bin=dec2bin(hora_corregida,5);
24
25      for n=1:5               %% 5 variables binarias para el tiempo
26
27          tiempo(n,i)=str2num(hora_bin(n));
28
29      end
30  end
```

Figura 5.6.- Función tiempo_bin

La única función especial que usamos aparte de *rem* que explicamos anteriormente, es la función *dec2bin* que cambia un valor en base decimal al mismo en base binaria. El problema es que dicha función no crea un número sino una cadena de caracteres o *string*, por ello utilizamos la función *str2num* para transformar dicha cadena en un vector numérico.

El argumento de entrada es el vector de carga horaria, pero también puede ser el vector “festivos” ya que sólo necesitamos la longitud del mismo.

5.1.7 Función “escalar”

Como se observa en la figura 5.7, con esta función escalamos los valores de la demanda eléctrica o de la temperatura entre 0 y 1. Esta consta de tres argumentos de entrada y tres de salida:

```
1   %                  Función escalar
2
3   % Escala los valores de los vectores de cargas horarias y de temperaturas
4   % a valores entre 0 y 1. Tiene tres inputs, el vector de datos
5   % correspondiente y los límites superior e inferior.
6   % Tiene una única salida, que es el vector de datos ya escalado.
7
8
9   function [datos_escalados]=escalar(datos, D_max, D_min)
10
11
12  datos_escalados=(datos-D_min)/(D_max-D_min);
13
14
```

Figura 5.7.- Función fiesta

La primera entrada corresponde al vector de datos de la demanda eléctrica o de la temperatura.

Los otros dos *inputs* corresponden a los límites superior e inferior que creamos convenientes para escalar el vector de datos. En nuestro caso elegimos el máximo y el mínimo registrados en todo el histórico de datos de que disponemos.

La salida es el vector de datos ya escalado.

5.1.8 Función “ent_sal_red”

Como se observa en la figura 5.8, esta es la función para entrenar a la red. Dicha red contará con 13 entradas para alimentarla. Con esta función se crea una matriz de 13 filas y un numero de columnas que dependerá de la longitud del vector de cargas, el cual viene determinado por el conjunto de entrenamiento. Además, crea un vector que se corresponde con lo que sería la salida ideal (TARGETS) que tendría si la red es alimentada con dicha matriz de “INPUTS”. También hay que introducir la semana anterior al conjunto de entrenamiento para poder hacer la primera iteración. Como podemos ver, esta matriz tiene dos argumentos de entrada y dos de salida.

Entradas:

- La primera entrada será la matriz formada por la unión del vector de datos de la demanda eléctrica perteneciente a la semana anterior del conjunto de entrenamiento, continuamos con el vector "festivos", y después la matriz con las horas en formato binario. Terminamos colocando los vectores correspondientes a la temperatura máxima y mínima. De esta forma obtenemos una matriz de 9 filas y "n" columnas, donde n es el numero de horas del periodo de entrada.
- La segunda entrada será exactamente igual que al anterior, con la diferencia que aquí introduciremos los datos correspondientes al mes de entrenamiento. Obtendremos una matriz con las mismas filas y columnas que en la primera entrada.

Salidas:

- En primer lugar el vector INPUTS, que estará formado por 13 columnas y n2 filas, siendo n2 el numero de horas del conjunto de entrenamiento.
- La segunda salida será "TARGETS", y se corresponde a los valores que deberá tomar la red al introducir los valores de entrada anteriores.

En el script se explica como deben de ser los argumentos de entrada para que funcione correctamente, y las dimensiones de todos los vectores y matrices que se utilizan.

El bucle for comienza en 169 debido a que tenemos que tenemos que coger la semana anterior al mes de entrenamiento.

```
%                   Función ent_sal_red

% Crea la matriz de entradas para nuestra red neuronal (cargas, día, hora,
% temperatura).
% Constará de 5+1+5+1 filas y n2 columnas, donde n2 será la longitud del
% vector de cargas target del conjunto de entrenamiento que será el segundo
% output.
% La primera fila tendrá la carga horaria de una semana anterior a esa
% misma hora, la segunda tendrá la carga horaria del día anterior a esa
% misma hora y las tres siguientes tendrán la carga de la tercera, segunda
% y anterior hora respectivamente. Nuestro training set será de un mes,
% luego la función tendrá dos inputs, dos matrices, una con la semana
% anterior al mes de entrenamiento junto con las cargas horarias más el
% vector de fin de semana, más la matriz de horas en binario mas el vector
% de temperaturas, y otra con el mes de entrenamiento mas el resto
% mencionado antes
%
% Ejemplo:
%
%   inputs:    [carga; festivo; hora; temperatura]
%                1xn    1xn    5xn     1xn
%
%   Donde n es el número de horas del período de entrada.
%
%   salida 1:  [cargas; festivo; hora; temperatura](en una misma matriz)
%                5*n2    1*n2    5*n2    1*n2
%
%   Donde n2 es el número de horas del conjunto de entrenamiento
%
%   En total, la salida 1 constará de n2 columnas y 12 filas
%
%   salida 2: [targets]
%               1*n2
%   Vector de targets para la red


function [entrada, targets] = ent_sal_red(datos_semana, datos_mes)

n1=length(datos_semana);
n2=length(datos_mes);

matriz=[datos_semana datos_mes];


% La primera semana se usará para poder arrancar el bucle.
% Por tanto, el bucle empezará desde la
% primera semana del mes de entrenamiento, o sea, 24*7+1=169

```

```
49 -    k=0;
50 -
51 -    for i=169:(n1+n2)
52 -
53 -        k=k+1;
54 -
55 -        entrada(1,k) = matriz( 1, i-24*7);
56 -        entrada(2,k) = matriz( 1, i-24);
57 -        entrada(3,k) = matriz( 1, i-3);
58 -        entrada(4,k) = matriz( 1, i-2);
59 -        entrada(5,k) = matriz( 1, i-1);
60 -        entrada(6,k) = matriz( 2, i);
61 -        entrada(7,k) = matriz( 3, i);
62 -        entrada(8,k) = matriz( 4, i);
63 -        entrada(9,k) = matriz( 5, i);
64 -        entrada(10,k) = matriz( 6, i);
65 -        entrada(11,k) = matriz( 7, i);
66 -        entrada(12,k) = matriz( 8, i);
67 -
68 -        targets(k) = matriz(1,i);
69 -
70 -    end
71 -
```

Figura 5.8.- Función ent_sal_red

5.1.9 Función “división_datos”

Como se observa en la figura 5.9, con esta función dividimos el conjunto de datos entre los tres grupos que se utilizaran durante el entrenamiento: training set; validation set; testing set.

Tiene cuatro entradas, la primera será el vector de datos, y nos sirve para medir la longitud de este. Las otras tres serán los porcentajes dedicados a cada uno de los subgrupos introducidos por el usuario.

El diseño de la función está enfocado para que el valor dedicado al training set sea el mayor de los tres. Aunque sirve para cualquier relación de porcentajes, lo mejor es utilizar siempre múltiplos. La división la hace de manera que redondea al valor de datos que mas se ajuste a nuestra elección.

Después de introducir estos datos, la función pedirá que le demos el valor del desplazamiento que queremos realizar en el conjunto de datos para que se vayan recorriendo los distintos grupos del conjunto de entrenamiento. Las posibilidades van desde 0 hasta 9. Esto debe a que la división de datos que hemos elegido es de 60-20-20, y habría 10 posibilidades de entrenamiento antes de que se volviera a repetir. Esta es la razón de la recomendación anterior de introducir siempre los porcentajes, sino esta parte del programa no sería válido para todos los casos.

Posteriormente el programa llama a la función “divideind”, que es la encargada de realizar la división de los índices del conjunto de datos en base a los anteriores.

Por ultimo, la función imprime una gráfica por pantalla donde podemos ver la división de los datos en cada grupo.

De las 7 salidas que tiene la función, 4 de ellas son imprescindibles para continuar con el programa. Son ‘trainInd’, ‘valInd’ y’ testInd’, que habrá que introducir como variables antes de entrenar la red, y ‘desplaz’, que recoge el valor del desplazamiento introducido por el usuario.

Las otras tres salidas, ’trainP’, ‘valP’ y ‘testP’ son el resultado obtenido por la función *“divideind”* pero no son necesarias fuera de la función *“división_datos”*. Se han incluido como salida por si se quieren utilizar para comprobar que la división se ha hecho correctamente. Dentro de la función *“divideind”* son necesarias para la representación gráfica.

```
%                   Funcion division_datos

% Esta funcion realiza la division de los datos de entrada de la potencia
% consumida en el mes a dividir (mesh) segun el porcentaje en que el
% usuario quiera dividir cada conjunto de datos (train, val y set).
% Es valido para cualquier relacion de porcentajes y para cualquier numero
% de dias que tenga el mes, puesto que la division la realiza de una manera
% aproximada ajustandose lo mas posible a los valores deseados de
% porcentajes.
% La funcion realiza tambien el desplazamiento de indices de cada
% subconjunto para el estudio de su robustez.
% Al final muestra por pantalla como ha dividido los subconjuntos, así como
% si se ha llevado a cambio el desplazamiento o no.
% La salida de la función son los conjuntos de valores que necesitará la
% red en el entrenamiento y que tenga en cuenta esta division de datos.


function [trainP,valP,testP,trainInd,valInd,testInd,desplaz] =
division_datos(mesh,trainRatio,valRatio,testRatio)

horas=0;
%%%%creo un vector para las horas
for i=1:length(mesh)
    vector_horas(1,i)=horas;
    horas=horas+1;
end

```

```
28
29      %%%%%creo los vectores índice
30 -    v=1;
31 -    k=1;
32 -    l=1;
33 -    cont=0;
34
35
36 -    ntrain=trainRatio*10;
37 -    nval=valRatio*10;
38 -    ntest=testRatio*10;
39
40
41
42 -    for i=1:round(trainRatio*length(vector_horas));
43          %el factor multiplicador depende del porcentaje dedicado
44          %a ntrain
45
46 -        trainInd(i) = v;
47
48 -        if rem(i,ntrain)==0
49 -            cont=cont+1;
50 -            v=v+1;
51 -            for j=1:(nval)
52 -                valInd(k) = v;
53 -                v=v+1;
54 -                k=k+1;
55 -            end
56
57 -            for j=1:(ntest)
58 -                testInd(l) = v;
59 -                v=v+1;
60 -                l=l+1;
61 -            end
62 -        else
63 -            v=v+1;
64 -        end
65 -    end
66
67
68      %ajuste de tamaño por las divisiones no enteras
69 -    if (length(trainInd)+length(valInd)+length(testInd))<length(vector_horas)
70 -        for i=((length(trainInd)+length(valInd)+length(testInd))+1):
71 -            length(vector_horas)
72 -            if (length(trainInd)-ntrain*cont)<ntrain
73 -                trainInd(length(trainInd)+1) = v;
74 -                v=v+1;
75 -            else if (length(valInd)-nval*cont)<nval
76 -                valInd(length(valInd)+1) = v;
77 -                v=v+1;
78 -            else
79 -                testInd(length(testInd)+1) = v;
80 -                v=v+1;
81 -                end
82 -            end
83 -        end
84 -    end
```

```
%%%Avanzar en línea temporal


i=1;
j=1;
k=1;

for n=1:length(mesh)
    if trainInd(i)== n
        vec_Ind(n)=1;
        if i<length(trainInd)
        i=i+1;
        end
    else if  valInd(j)== n
             vec_Ind(n)=2;
             if j<length(valInd)
             j=j+1;
             end
    else if testInd(k)== n
             vec_Ind(n)=3;
             if k<length(testInd)
             k=k+1;
             end
        end

        end

    end
end


%%despalazar


desplaz=input('\nIntroduzca desplazamiento (0~9): ');
if isempty(desplaz)
    desplaz = '1';
end


switch desplaz
    case 0, vec_Ind=vec_Ind;
    case 1, vec_Ind=[3 vec_Ind];
    case 2, vec_Ind=[3 3 vec_Ind];
    case 3, vec_Ind=[2 3 3 vec_Ind];
    case 4, vec_Ind=[2 2 3 3 vec_Ind];
    case 5, vec_Ind=[1 2 2 3 3 vec_Ind];
    case 6, vec_Ind=[1 1 2 2 3 3 vec_Ind];
    case 7, vec_Ind=[1 1 1 2 2 3 3 vec_Ind];
    case 8, vec_Ind=[1 1 1 1 2 2 3 3 vec_Ind];
    case 9, vec_Ind=[1 1 1 1 1 2 2 3 3 vec_Ind];
    otherwise vec_Ind=vec_Ind;
end

for i=1:(length(vec_Ind)-desplaz)
     vec_aux(i) = vec_Ind(i);
end
```

```
147 -     vec_Ind = vec_aux;
148 -     clear vec_aux
149
150       %%%%
151
152       %%%descodifico
153 -     i=1;
154 -     j=1;
155 -     k=1;
156
157 -   for n=1:length(vec_Ind)
158 -       if vec_Ind(n)==1
159 -           trainInd(i)=n;
160 -           i=i+1;
161 -       else if vec_Ind(n)==2
162 -           valInd(j)=n;
163 -           j=j+1;
164 -       else if vec_Ind(n)==3
165 -           testInd(k)=n;
166 -           k=k+1;
167 -           end
168 -           end
169 -       end
170 -   end
171       %%%
```

```
175       %calculo
176 -     [trainP,valP,testP] = divideind(vector_horas,trainInd,valInd,testInd);
177
178       %dibujo
179 -     figure;
180 -     plot(vector_horas,mesh,'k');
181 -     hold on
182 -   for n=1:length(trainP)
183 -       plot(trainP(n),mesh(trainInd(n)),'rx');
184 -   end
185
186 -   for n=1:length(valP)
187 -       plot(valP(n),mesh(valInd(n)),'gx');
188 -   end
189
190 -   for n=1:length(testP)
191 -       plot(testP(n),mesh(testInd(n)),'bx');
192 -   end
193
194 -     title('Uso de divideind:  Train = rojo    Val = verde   Test = azul');
195 -     xlabel('Tiempo(horas)');
196 -     ylabel('Energia(kW)');
197
198
199       %limpio las variables
200 -     clear i
201 -     clear j
202 -     clear n
```

```
203 -   clear v
204 -   clear k
205 -   clear l
206 -   clear cont
207 -   clear ntrain
208 -   clear nval
209 -   clear ntest
210 -   clear vector_horas
211 -   clear horas
```

Figura 5.9.- Función división_datos

5.2 Funciones de Matlab para la RNA

Vamos a utilizar la toolbox que Matlab tiene destinadas a utilizar en redes neuronales artificiales, que son las que hemos utilizado para la creación, entrenamiento y simulación de las red neuronal que estamos estudiando. Vamos a hacer una breve descripción de las funciones que hemos utilizado.

5.2.1 Función "newff"

Con esta función se crea la red neuronal. Las letras que van a continuación de la palabra en inglés "new" nos indica el tipo de red neuronal que vamos a utilizar, en nuestro caso la doble "f" hace referencia a "feed-foward", que como comentamos en otros puntos de la memoria, la red se alimenta desde la capa de entrada y la información va avanzando hasta el final pasando por cada capa sucesiva. Esta red es del tipo perceptrón multicapa "backpropagation", o propagación hacia atrás, y se refiere al método de aprendizaje, de adelante hacia atrás.

Su sintaxis es muy sencilla, igualando el nombre de la red a la función. Los argumentos de entrada variarían en función de las modificaciones que deseemos hacer respecto de la configuración que tiene por defecto. Eso sí, los mínimos argumentos que debemos indicar son el número de neuronas que deseamos en nuestra red en cada capa oculta, el vector o matriz de entradas de entrenamiento, y el vector o matriz de salida. La función de activación por defecto en esta función es la tangente hiperbólica, y como queremos utilizar la función sigmoide, se tiene que incluir un cuarto argumento. Este es un ejemplo de creación de una red:

$$red = newff(INPUTS, TARGETS, neuronas, \{'logsig'\});$$

Donde "red" es la variable de salida de la red que queremos crear, "INPUTS" es la matriz de datos de entrada, "TARGETS" la matriz de datos de salida, "neuronas" es un vector con el número de neuronas en la capa oculta,

y el último argumento es el nombre con el que Matlab asigna como función sigmoide de activación a las neuronas de la capa oculta.

5.2.2 Función "train"

Es la función que se encarga de entrenar la red neuronal. Matlab entrena la RN por defecto con el algoritmo Levenberg-Marquardt.

Lo que si que se ha modificado han sido los porcentajes de training set que se van a dedicar al entrenamiento, test y validación. Los porcentajes que se han fijado en el script han sido 60, 20 y 20% respectivamente. Esto se ha hecho con tres líneas simples en el script de entrenamiento y simulación, lo cual se puede ver ahí.

El uso de esta función es muy sencilla. Se ha igualado la red a la función teniendo ésta como argumentos la propia red, la matriz con los datos de entrada y la matriz con los datos de salida:

$$red = train(red, INPUTS, TARGETS);$$

Donde "red" es el nombre de la red que hemos creado con la función anterior, al igual que las entradas. No es preciso que sean las mismas, pero en todos los ejemplos de Matlab son iguales, puesto que es lo mas fácil si ya están cerradas.

5.2.3 Función "sim"

Esta función se utiliza para poder obtener los resultados de la red neuronal. Su sintaxis es muy simple y hemos optado por dejar las opciones que tiene por defecto.

Sus argumentos de entrada son la propia red neuronal y el vector o matriz seleccionadas que queremos que procese la red. La función tiene como salida un vector con las salidas correspondientes a las entradas.

Este seria el ejemplo de uso de la función:

$$prediccion = sim(red, INPUT);$$

Donde "red" es el objeto (neural network object) o red neuronal, "INPUT" es la matriz o vector de entradas y "predicción" es la salida de la red neuronal.

5.2.4 Función “init”

Con esta función reiniciamos todos los pesos sinópticos de la red neuronal, es decir, se vuelve al inicio de la red neuronal. Se recurre a esta función porque nuestras estimaciones en el estudio principal resultaban de la media de 20 estimaciones. La red es entrenada nuevamente para cada nueva estimación, pero para que los resultados sean diferentes cada vez, hay inicializar la red, puesto que si no se hace, el primer entrenamiento fija los parámetros y los demás entrenamientos llegan a la misma solución dando como resultado la misma estimación y volviendo inútil el sistema de las 20 estimaciones. Parece ser que, con los pesos sinópticos de la primera iteración, los algoritmos de optimización del error llegan rápidamente a la solución que es exactamente la misma. Por tanto, esta función es de vital importancia en nuestro trabajo.

La sintaxis es muy sencilla, sin argumentos de salida y con un único argumento de entrada que es la propia red:

```
red=init(red);
```

Donde “red” es el nombre de la red neuronal, como en las anteriores funciones.

Capítulo 6

Resultados

6 Introducción

En este capítulo vamos a dar una explicación de los resultados obtenidos, donde mostraremos tanto gráficas como tablas para fundamentar las conclusiones, la validez de la red neuronal, y para terminar, el número de neuronas de la capa oculta junto con el grado de suavizado del spline cúbico.

Para todo comparar los resultados nos basaremos en un diagrama tipo box plot. En cada diagrama que representemos aparecerán diez box plot, uno por cada desplazamiento de la división de datos que se realice. Estos box plot recogerán los valores del error medio relativo obtenidos a partir de la comparación hora a hora de las estimaciones y los datos reales en todo el conjunto de entrenamiento.

Estas son las expresiones matemáticas que se han utilizado:

- **Error Medio Relativo**

$$Error\ _i(\%) = \frac{|Demanda\ Estimada_i - Demanda\ Real_i|}{Demanda\ Real_i} x100 \qquad \text{(VI-1)}$$

donde i nos indica la hora del cálculo del error

- **Desviación Típica**

$$Desviación\ T\ _j(kW) = \sum_i^n (Demanda\ Estimada_i - Demanda\ Real_i)^2\ n \qquad \text{(VI-2)}$$

donde i nos indica la hora i,

n nos indica el número de horas del vector error

Con esta variable calculamos a dispersión de la estimación frente a la demanda real.

Los resultados del error medio relativo y los de la desviación típica de cada desplazamiento de cada estimación los recogeremos en una tabla que acompañará a los diagramas box plot.

6.1 Número de Neuronas Artificiales

Para finalizar el estudio hemos hecho un total de 10 simulaciones en tres estudios distintos: 10, 15 y 20 neuronas en la capa oculta. A continuación se muestran los resultados y las conclusiones. Para ver cual es el número mas adecuado, se ha analizado en primer lugar el error medio, ya que es el dato mas significativo. Posteriormente se ha repetido el proceso con la desviación típica, ya que con esta tolerancia se puede dar una conclusión mas afinada. Es muy importante resaltar que la muestra de datos con la que he realizado el estudio no es lo suficientemente grande como para obtener unas conclusiones del todo fiables, pero si que vale pare tener una idea de la calidad de nuestra red neuronal y cuales son los mejores argumentos con los que debe de contar esta.

También hay que tener en cuenta que el mes a predecir es el de diciembre, el cual tiene bastantes días de fiesta y también días “atípicos”, debido a la navidad y a diferentes eventos que también se podrían tener en cuenta como variable de entrada a la red neuronal. A continuación iremos desgranando cuales son esos días, las posibles causas y soluciones.

La semana a estimar ha sido la primera de diciembre de 2015, la cual empezó un martes y terminó un lunes. Dicho lunes era festivo debido a que este año el día de la constitución ha coincidido con un domingo y la fiesta se trasladó al lunes. Como se ha mencionado en el capítulo 1, debido al problema con el analizador de redes solo disponíamos de datos suficientes para tres meses, por eso no se ha podido simular más semanas. Para hacer un estudio más completo se necesitan más meses de datos, pudiendo hacer más simulaciones, consiguiendo de esta forma una RNA más robusta y fiable.

Como podremos observar a continuación en las tablas y gráficas veremos que apenas hay diferencia entre tener 10, 15 ó 20 neuronas en la capa oculta, pero hay alguna pequeña diferencia que nos hace intuir cual será la mejor de las tres muestras en el caso de simular para un periodo mas largo de una semana, ya que como explicamos a lo largo de esta memoria, cuanto más tiempo queramos predecir, mayor será el error que tendrá esta.

6.1.1 Simulaciones con 10 neuronas

Después de realizar un total de 10 simulaciones con un total de 10 neuronas en la capa oculta, se ha procedido a construir la tabla 6.1 donde se muestran los resultados obtenidos.

Tabla 6.1-Resultados simulación con 10 neuronas en la capa oculta

Simulación	Error Medio	Desviación típica(kW)
1	7,0390	161087
2	7,9121	183123
3	7,5386	176938
4	7,2347	176708
5	7,0027	162808
6	7,1367	158835
7	7,4891	161912
8	7,4153	163039
9	7,0030	153904
10	7,3023	168987

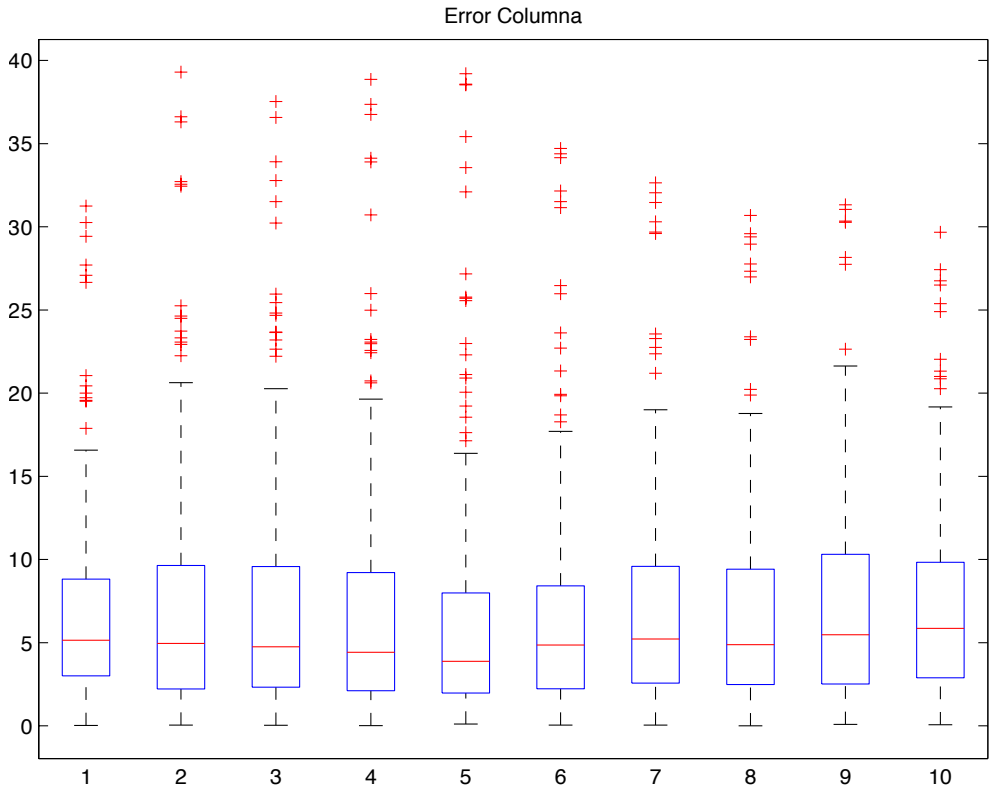

Figura 6.1.-Resultados BoxPlot para 10 neuronas

En el diagrama BoxPlot de la figura 6.1 se observa de una manera gráfica la distribución de los resultados, así como los valores atípicos que presenta la simulación.

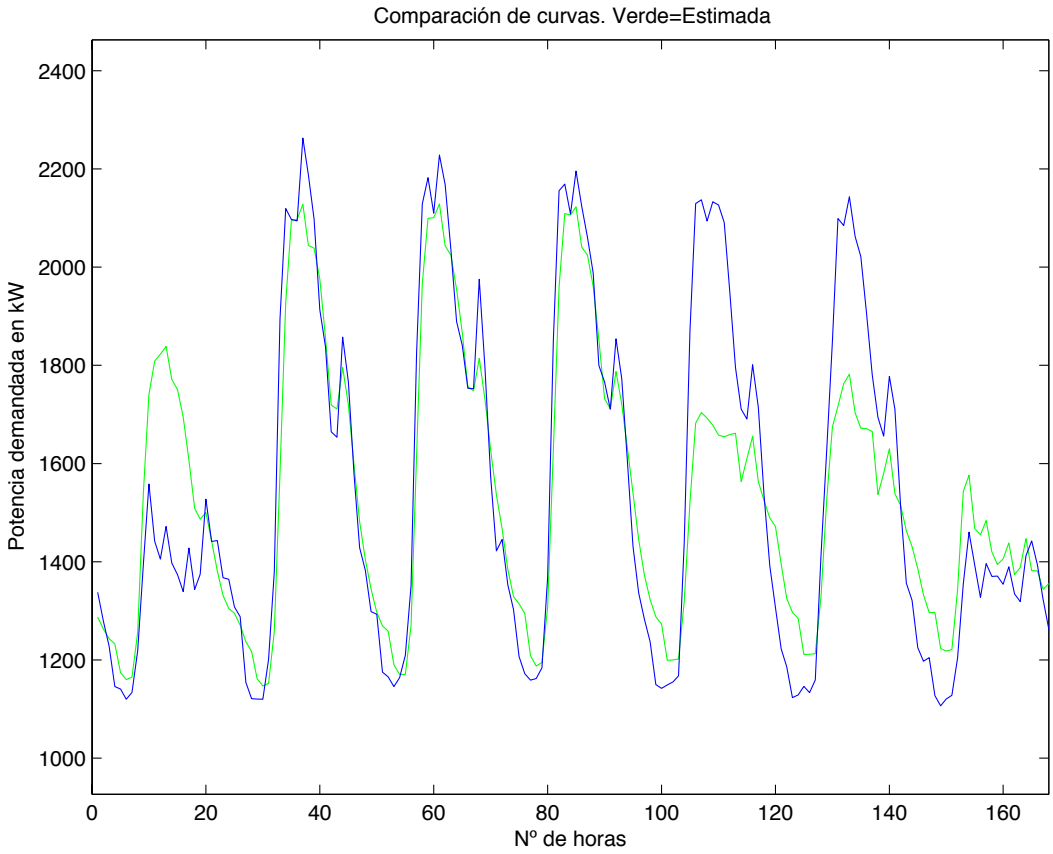

Figura 6.2.- Resultados simulación para 10 neuronas. En verde la curva estimada, en azul la curva real.

En el gráfico de la figura 6.2 se observa la representación de dos curvas. Se trata de la curva real (color azul) y de la simulada (color verde). De manera muy rápida se puede ver que hay tres días que se ajustan muy poco a la curva real, estos son el martes, el sábado y el domingo. El último también se sale un poco de la curva real, aunque de forma menos pronunciada que los casos anteriores. Esto es muy posible que se deba a que sería conveniente introducir una variable para tener en cuenta este tipo de días, y a la problemática comentada anteriormente con el mes de diciembre. Vemos que el primer día de predicción la curva simulada es algo mas alta, luego tenemos tres días que son prácticamente idénticos a los reales, continuamos con otros dos que se quedan cortos y finalmente el último, también es bastante aceptable.

6.1.2 Simulaciones con 15 neuronas

Después de realizar un total de 10 simulaciones con un total de 15 neuronas en la capa oculta, se ha procedido a construir la tabla 6.2 donde se muestran los resultados obtenidos.

Tabla 6.2-Resultados simulación con 15 neuronas en la capa oculta

Simulación	Error Medio	Desviación típica(kW)
1	7,1186	113858
2	6,7533	121232
3	6,5413	133990
4	7,0583	128127
5	7,0329	140859
6	6,6402	118113
7	7,1295	110587
8	7,0615	129701
9	7,0177	118549
10	6,4215	114401

Como podemos observar, si comparamos la tabla 6.1, donde se simulaba con 10 neuronas, con la 6.2 los errores son algo menores en esta última. Es muy poca la diferencia, pero al igual que se ha comentado anteriormente, el error se va acumulando en las simulaciones cuanto mayor es el tiempo a predecir. Si observamos la desviación típica sucede lo mismo, nos encontramos con una cifras más bajas que en el caso de 10 neuronas.

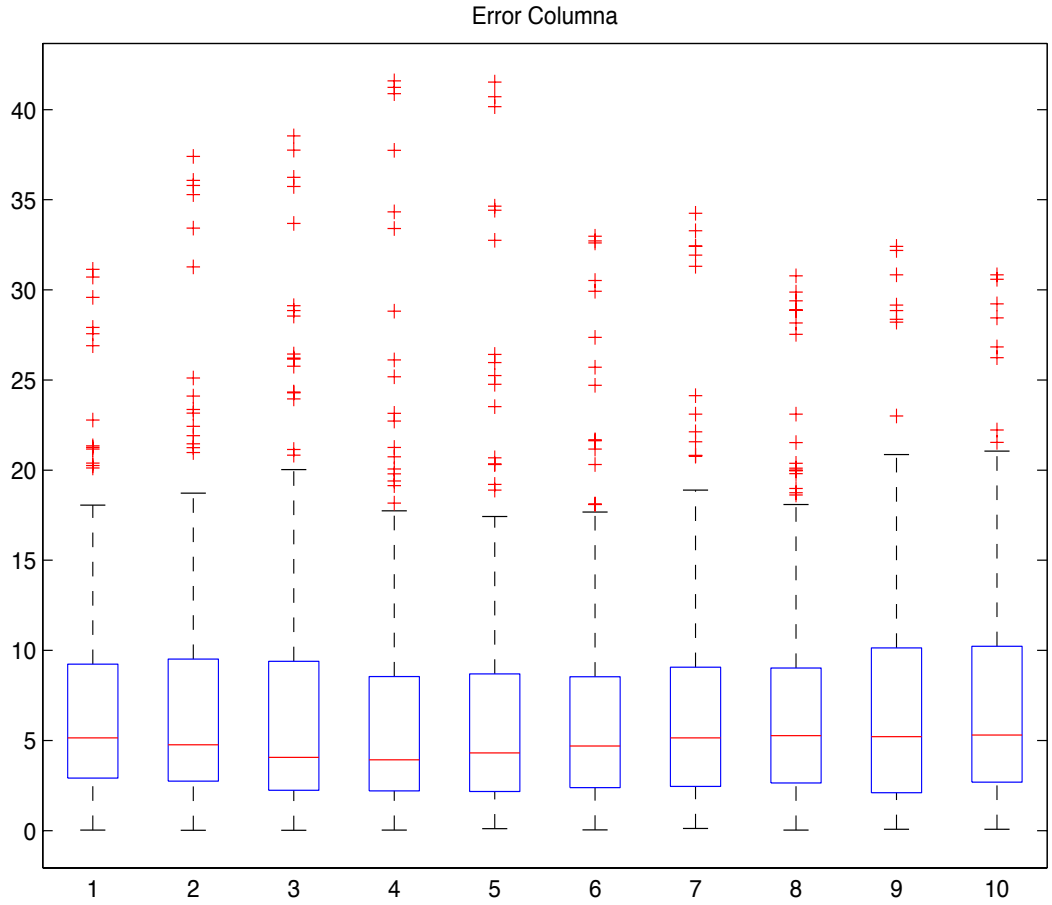

Figura 6.3-Resultados BoxPlot para 15 neuronas

En el diagrama BoxPlot de la figura 6.3 se observa de una manera gráfica la distribución de los resultados, así como los valores atípicos que presenta la simulación.

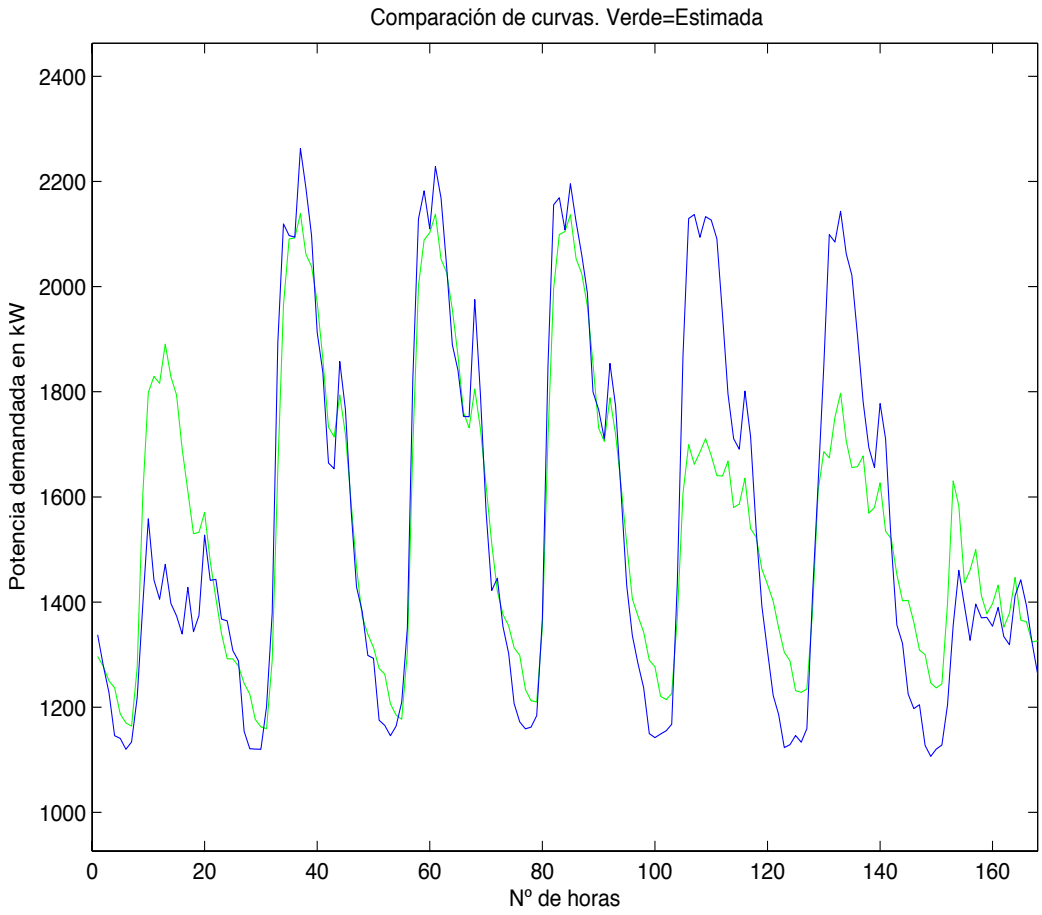

Figura 6.4.- Resultados simulación para 15 neuronas. En verde la curva estimada, en azul la curva real.

Como podemos observar el gráfico de la figura 6.2 y el 6.4 son prácticamente iguales. Esto se debe a que trabajamos con los mismos datos, con la única diferencia de las neuronas en la capa oculta y a que las diferencias son muy pequeñas.

6.1.3 Simulaciones con 20 neuronas

Después de realizar un total de 10 simulaciones con un total de 20 neuronas en la capa oculta, se ha procedido a construir la tabla 6.3 donde se muestran los resultados obtenidos.

Tabla 6.3-Resultados simulación con 20 neuronas en la capa oculta

Simulación	Error Medio	Desviación típica(kW)
1	7,0869	161724
2	7,6895	178289
3	7,2434	180186
4	7,3583	178856
5	6,6417	162483
6	7,0806	158002
7	7,6642	159879
8	6,9634	159124
9	7,6129	171127
10	7,2188	161588

La tabla 6.1 muestra el error medio y la desviación típica para una red neuronal con 20 neuronas en la capa oculta y un total de 10 simulaciones. Como podemos observar, los errores son algo mayores que los obtenidos para las simulaciones con 15 neuronas.

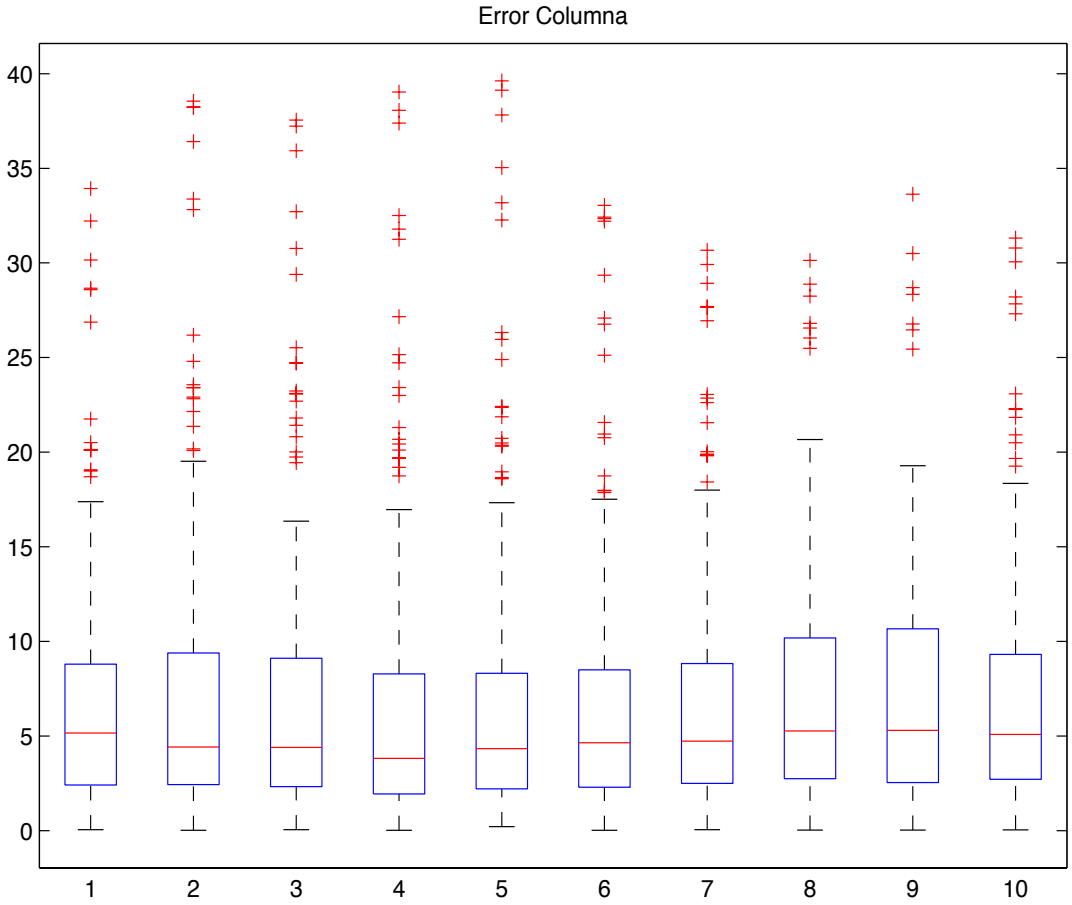

Figura 6.5.-Resultados BoxPlot para 20 neuronas

En el diagrama BoxPlot de la figura 6.5 se observa de una manera gráfica la distribución de los resultados, así como los valores atípicos que presenta la simulación.

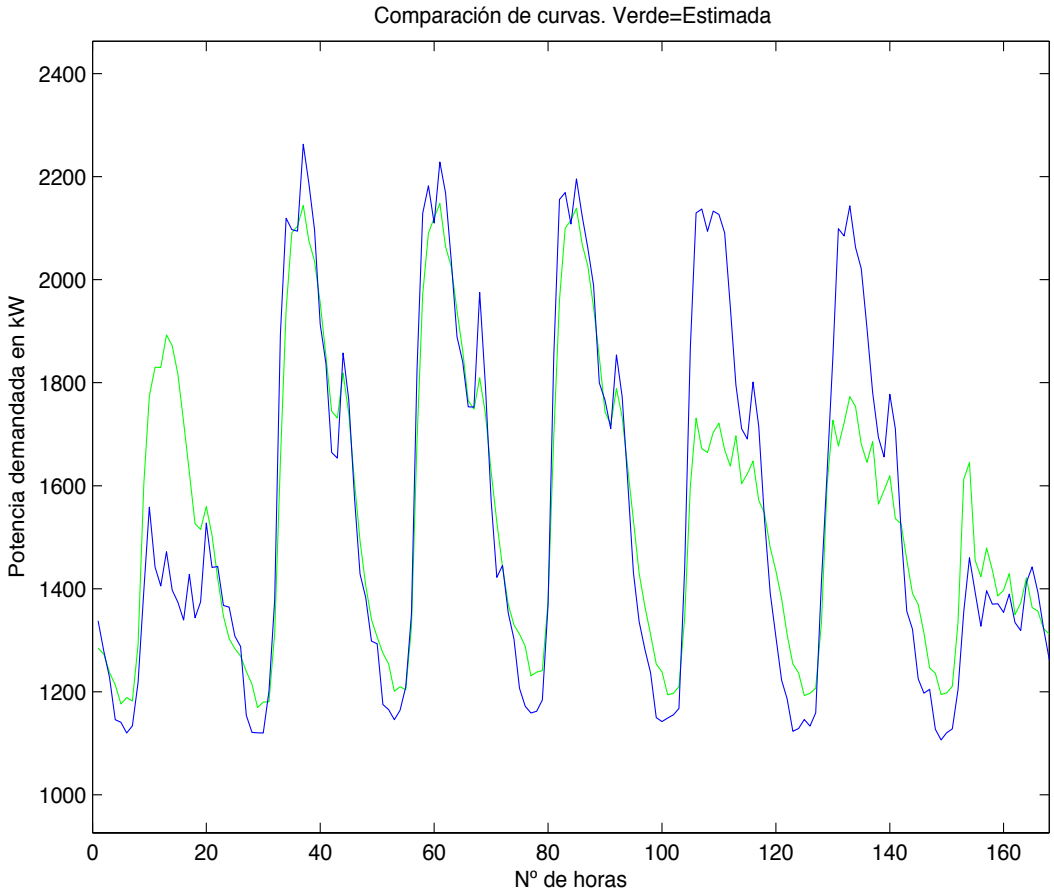

Figura 6.6- Resultados simulación para 20 neuronas. En verde la curva estimada, en azul la curva real.

Lo explicado en las imágenes 6.2 y 6.4 vale para la interpretación de la imagen 6.6, ya que los gráficos son muy similares en los tres casos.

6.2 Análisis de los resultados

Una vez que se han analizado los resultados obtenidos , la conclusión ha sido que con 15 neuronas en la capa oculta, la red es más robusta que para 10 y 20 neuronas. Esta elección es muy importante porque el número de neuronas en la capa oculta, incide directamente en los resultados de los siguientes estudios que se van a realizar a continuación.

6.3 Grado de Suavizado del Spline Cúbico

Una vez que sabemos cual es el mejor caso en función del número de neuronas en la capa oculta, se va a hacer lo propio para el suavizado del spline cúbico. Se harán pruebas con un grado de suavizado de 0,75, 0,85 y 0,95. Para las pruebas anteriores determinamos un grado de suavizado de 0,95 con el fin de que todos los datos se pudieran comparar.

Tabla 6.4.-Resultados simulaciones con los tres grados de suavizado: 0,75, 0,85, 0,95

Simulación/suavizado	Error	DT	Error	DT	Error	DT
	0,75		0,85		0,95	
1	7,6464	172160	7,4827	169591	7,1186	113858
2	7,8243	195540	7,387	188664	6,7533	121232
3	7,3485	175117	7,8211	184341	6,5413	133990
4	7,5739	186188	7,5891	187273	7,0583	128127
5	7,2345	188850	7,1942	168407	7,0329	140859
6	7,8097	189488	7,7679	170442	6,6402	118113
7	7,3546	178911	8,0325	167855	7,1295	110587
8	6,9127	169357	7,6907	167411	7,0615	129701
9	8,1345	179500	7,8732	176176	7,0177	118549
10	8,3446	178866	7,2589	164635	6,4215	114401
11	7,8978	179857	8,3333	190057	7,335	165326
12	8,0238	177403	7,4843	197147	6,9274	116976
13	7,5334	175590	7,5927	184193	7,1262	149995
14	7,3656	177914	7,4534	180820	7,0144	146285
15	7,0661	185853	6,8841	168259	6,9421	152462
16	7,5979	190015	7,5304	168739	7,0186	173911
17	7,4569	185587	8,1802	176599	7,3206	168617
18	7,2838	177223	7,6354	163536	7,4445	168177
19	6,8176	169782	7,1918	161889	7,1438	135367
20	8,1992	179498	7,855	168432	7,3246	131230

Como podemos apreciar visualmente en la tabla 6.4, donde se han realizado un total de 20 simulaciones para cada uno de los suavizados de spline cúbico mencionados anteriormente, se obtiene un menor error medio y desviación típica para el caso de 0,95. Esto quiere decir que como intuíamos antes de realizar el estudio, el suavizado de los datos afecta de una forma elevada en la predicción de la curva de consumo.

6.4 Añadiendo una entrada nueva: La Humedad Relativa Media Diaria

Con el fin de seguir avanzando en la precisión de nuestra red neuronal se ha decidido introducir en esta una nueva entrada, pasando a tener 14 entradas. La elección de esta variable se debe al alto grado de incidencia que tiene la humedad en el consumo de energía eléctrica en el hospital. Esto se debe, a grandes rasgos, a que si para una temperatura variamos el índice de humedad relativa, necesitaremos menos energía para aumentar o reducir la temperatura. Como inconveniente tenemos que es necesario mantener la humedad siempre dentro de ese rango, en el caso del Hospital Rio Hortega se mantiene entre el 45% y 55%. También se debe a necesidades de seguridad en un hospital, y aquí influyen varios factores. En los quirófanos hay una serie de medidas de seguridad obligatorias entre las cuales tenemos la necesidad de mantener la humedad relativa del aire en un 55%. Las razones son las siguientes¨

- La electricidad estática: Cuanto mas baja es la humedad relativa, menor es el riesgo de chispazo.
- El control de bacterias: Se ha demostrado que en el rango de humedad de entre el 45%-55% es donde menos proliferan estas.

Mantener esta en los rangos descritos anteriormente requiere un importe gasto de energía. Por todas estas razones hemos considerado introducir esta nueva variable en nuestro estudio.

Como ya sabemos que el mejor parámetro de suavizado del spline cúbico es 0,95, voy a realizar 20 simulaciones y comparar los resultados con los obtenidos en la tabla 6.2-1 del capítulo anterior.

6.4.1 Resultados Obtenidos

A continuación se muestran los resultados obtenidos después de introducir la nueva variable de entrada: La humedad.

Tabla 6.5.-Resultados con variable humedad

Simulación	Error Medio	Desviación típica(kW)
1	7,1659	158262
2	8,1982	188934
3	7,4152	183040
4	7,4463	186394
5	7,5619	182352
6	6,9215	165268
7	6,9502	156144
8	6,9594	153307
9	7,6612	166995
10	7,7934	165935
11	7,2671	162149
12	7,2355	161567
13	7,1984	157890
14	7,4752	162562
15	7,1273	152908
16	7,0203	157768
17	6,812	164729
18	7,5886	183147
19	7,5661	187275
20	7,1095	178706

Comparando los resultados con los de la tabla 6.5 y los de la tabla 6.4 del capítulo anterior, observamos que los resultados son algo peores. Tanto en la columna del error relativo medio como en la de la desviación típica no hemos conseguido reducir los valores. Esto puede deberse a que el parámetro que hemos introducido depende de otros valores de entrada en nuestra red neuronal.

Como hemos explicado a lo largo de la memoria, las entradas de una red neuronal artificial tienen que ser lo mas independientes posibles unas de otras. Quizás la humedad en nuestra zona esté muy relacionada con la temperatura, y al introducir esta nueva variable obtenemos un resultado contraproducente a pesar de tener más datos.

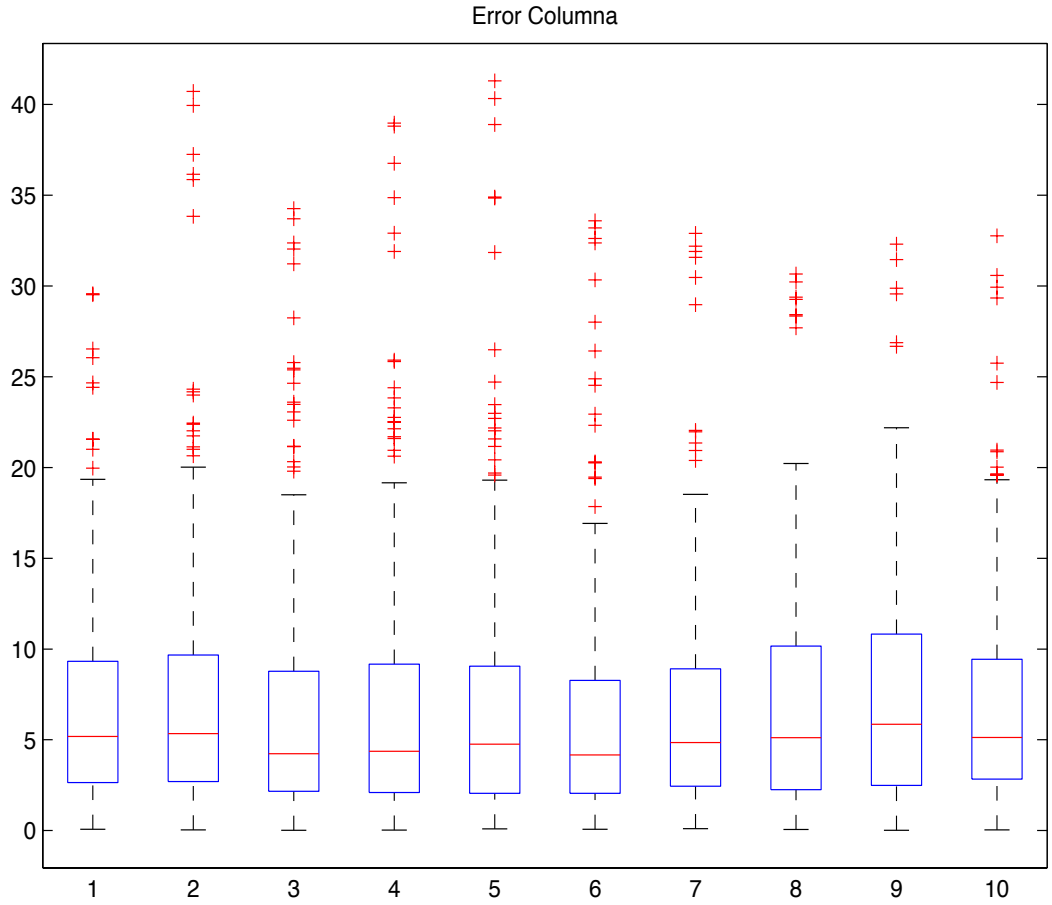

Figura 6.7-Resultados BoxPlot para 15 neuronas con variable humedad

La imagen 6.7 representa el BoxPlot de las 10 últimas simulaciones, donde se observa la distribución de los resultados y los puntos atípicos.

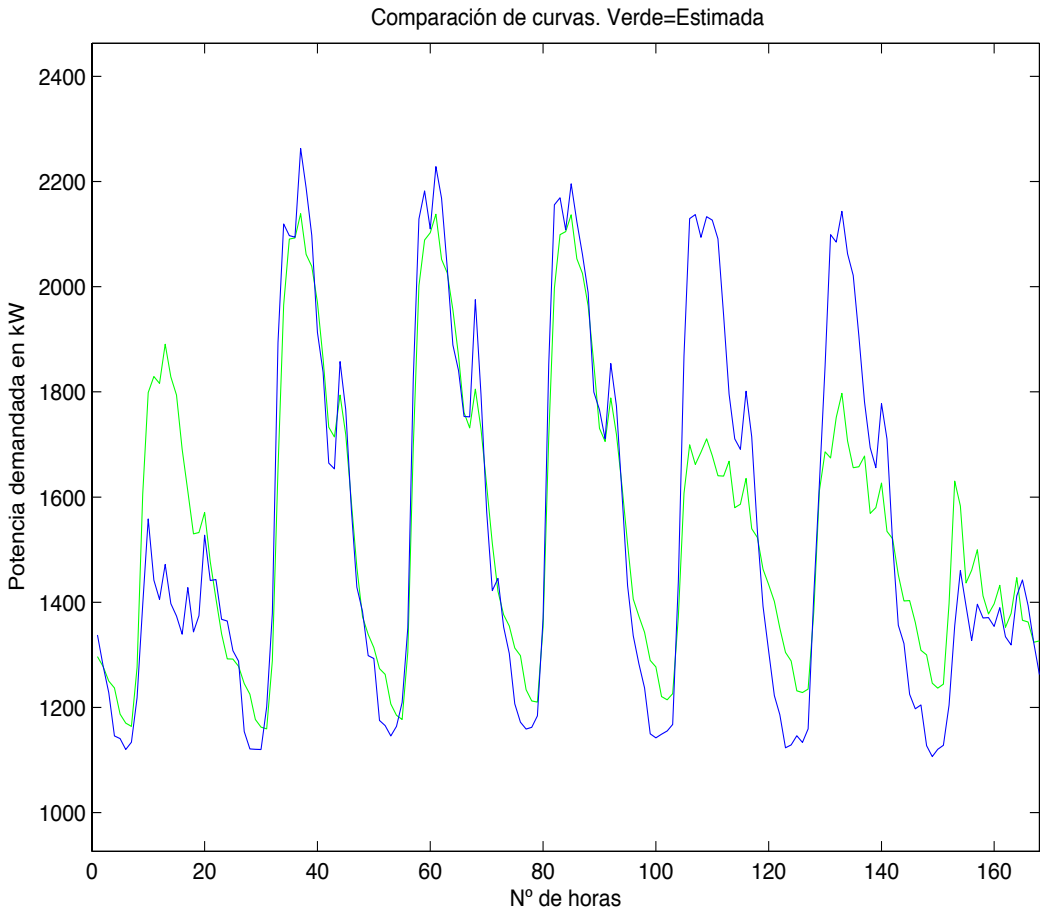

Figura 6.8.- Resultados simulación para 15 neuronas con la variable humedad. En verde la curva estimada, en azul la curva real.

En la figura 6.8 podemos observar que está algo menos aproximada que la figura 6.6 del apartado anterior. La diferencia es mínima, pero con el apoyo de las tablas 6.5 y 6.2 para observar las diferencias entre las desviaciones típicas y los errores medios, hace que se confirmen los datos obtenidos en la gráfica, que eran peores que los de la red neuronal sin la entrada de la variable humedad.

Capítulo 7

Conclusiones y Líneas Futuras

7.1 Conclusiones

Para finalizar el presente proyecto, se pueden obtener una serie de conclusiones que mostramos a continuación.

El primer objetivo de este proyecto era analizar cual es el número más adecuado de neuronas en la capa oculta. Una vez realizadas las pertinentes simulaciones para 10, 15 y 20 neuronas y comparados los resultados, se ha llegado a la conclusión de que lo mejor es tener 15 neuronas, pues tanto el error relativo medio como la desviación era la que mejores números arrojaba.

Una vez elegido el número de neuronas en la capa oculta, el siguiente objetivo era obtener el grado de suavizado de spline cúbico que mejor se ajustara a la red. Se optó por comparar un suavizado de 0,75, 0,85 y 0,95. Para ello se hicieron un total de 20 simulaciones para cada grado de suavizado, y comparando los resultados obtenidos de la misma forma que hicimos anteriormente se llegó a la conclusión de que el mejor valor para nuestra red era el de 0,95.

El último objetivo era introducir una nueva variable. Una vez analizadas las posibles nuevas entradas a la red, se decidió optar por la humedad relativa media diaria. Para ello se hicieron los cambios necesarios en el programa. El procedimiento a seguir fue el mismo que para elegir el número de neuronas en la capa oculta. A continuación se analizaron los resultados que arrojaron las simulaciones y se pudo ver claramente que los resultados eran peores, con un error medio relativo y una desviación típica claramente superiores. Es posible que esto se deba a que la humedad depende de otra variable ya introducida en la red. El resultado de una RNA no depende del número de entradas que esta reciba, sino de que se introduzcan las que de verdad influyen en los resultados que queremos obtener.

Para realizar la RNA se han utilizado los datos proporcionados por los analizadores de redes situados en los Centros de Transformación del Hospital, y mediante el software de cálculo y programación Matlab y la RNA creada, se han realizado las comparaciones mostradas anteriormente.

A pesar del problema en la recogida de datos, se han podido demostrar todas las posibilidades que puede dar la RNA al Hospital.

Esta herramienta podría ser utilizada para aplicaciones como alisar la curva de carga, que los transformadores de los Centros de Transformación no trabajen en vacío, y además que estos se mantengan en la medida de lo posible en su curva óptima de trabajo.

Todo este ahorro de energía que se puede conseguir con estas aplicaciones, se pueden traducir en un ahorro económico, el cual variará en función del precio de la energía.

Además del ahorro económico, también hay que valorar el ahorro en forma de la prolongación de la vida útil de los transformadores.

7.2 Líneas Futuras de Actuación

La RNA creada se acerca bastante a los resultados esperados, pero convendría revisar que pasa con los días que podemos considerar "atípicos" introduciendo una nueva variable.

La mayor parte de las mejoras deberían de ser en torno a las nuevas variables de entrada a la red.

Una vez comprobado que la variable de entrada de la humedad media diaria no satisface las necesidades de la red, convendría introducir alguna nueva. Como posible mejora se puede actuar en la temperatura, que a pesar de ser una variable con la que contamos, se podría introducir en forma de temperatura media diaria del día anterior.

BIBLIOGRAFÍA

En esta sección se muestra la bibliografía consultada para la realización de este Trabajo Fin de Grado, ordenada por orden alfabético del apellido del primer autor. Esta bibliografía no se ha citado expresamente en el texto, en algunos casos.

Libros

- Box G., Jenkins G. "Time Series Analysis: Forecasting and Control", publicado por Scientific Computing Associates, ISBN: 978-1-118-67502-1, Chicago 2015.
- Burden L. Richard, Douglas F." Numerical analysis", Editorial Cengage Learning, ISBN: 13-978-0-538-73351-9, Boston 2011.
- Douglas C. M., George C.R. "Probabilidad y Estadística aplicadas a la ingeniería", Editorial Mc Graw Hill, ISBN: 968-26-1232-2, México 1993.
- Enders, W. "Applied Econometric Times Series", Editorial John Wiley & Sons, Inc. United States 1995.
- Hilera JR. "Redes neuronales artificiales: Fundamentos, modelos y aplicaciones", Editorial Astro Data, ISBN: 0-201-87895-X, Delaware 1995.
- Haykin S., "Neural Networks", Editorial Pearson Prentice Hall, ISBN 81-7808-300-0, Singapur 2001.
- Johansen S. "Likelihood-Based inference in cointegrated vector autorregressive models", Oxford University Press. Oxford 1995.
- Otero, J. M., " Econometría. Series temporales y predicción", Editorial AC, Madrid 1993.
- Smeral E., Witt S., "Econometric forecasts of tourism demand to 2005. Annals of Tourism Research", Vol. 23, n° 4, pp. 891-907.
- Steven C. Chapara, Raymond P. "Métodos numéricos para ingenieros", Editorial Mc Graw Hill, ISBN: 970-103965-3, México D.F 2006
- Willis H. Lee. "Spatial electric load forecasting", Editorial Marcel Dekker, ISBN: 978-0-8247-0840-5, North Carolina 2002.
- Wooldrige J. "Introducción a la econometría. Un enfoque moderno", editorial Thomson Paraninfo, ISBN: 0-324-11364-1, Madrid 2008.

Tesis y Proyectos

- Talaya E. "Análisis de la Demanda. Aplicación a la actividad turística de las técnicas de predicción", Tesis Doctoral. Universidad Complutense, Madrid 1987.

- Ariza A. M. “Métodos utilizados para el pronóstico de demanda de energía eléctrica en sistemas de distribución” Proyecto de la Universidad Tecnológica de Pereira, 2013.
- Jiménez F. M. “Redes neuronales y pre procesado de variables para modelos y sensores en bioingeniería “ Tesis doctoral, Universidad de Valencia 2012.
- Méndez Pérez J., González G., Torres S. “Análisis del rendimiento de un sistema de transformadores en paralelo en un gran edificio” Universidad de la Laguna.

Páginas web

- Inforriego, Junta de Castilla y León Consulta: 13 de junio 2016]
  Disponible en: http://www.inforiego.org/opencms/opencms
- Universidad de los Andes, Venezuela [Consulta: 13 de junio 2016].
  Disponible en:
  http://tesis.ula.ve/postgrado/tde_busca/arquivo.php?codArquivo=7243
- Universidad Autónoma de Madrid, Consulta: 13 de junio 2016].
  Disponible en:
  https://www.uam.es/personal_pdi/economicas/anadelsur/pdf/Box-Jenkins.PDF
- Hospital Universitario Rio Hortega, SACYL [Consulta: 13 de junio].
  Disponible en: http://www.saludcastillayleon.es/HRHortega/es
- BOE, Directiva 2002/91/CE relativa a la eficiencia energética de los edificios [Consulta: 13 de junio].
  Disponible en: http://www.boe.es/buscar/doc.php?id=DOUE-L-2003-80006