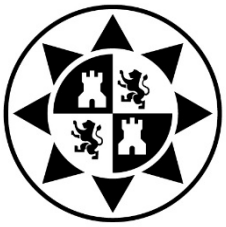


Universidad
Politécnica
de Cartagena

industriales
etsii UPCT

Estudio de la predictibilidad del rendimiento académico de alumnos en asignaturas de segundo curso.

Titulación: Grado en Ingeniería en Tecnologías Industriales.

Alumno: Juan Luis Muñoz Fernández.

Director: Mathieu Kessler.

Cartagena, 8 de Julio de 2014

Resumen

El rendimiento académico universitario es un asunto de capital importancia en la sociedad en que vivimos por las implicaciones socio-culturales y económicas que lleva asociadas. En este trabajo se abordará por un lado la predicción de la nota de egreso de alumnos para las titulaciones de Ingeniero Industrial e Ingeniero de Telecomunicaciones y, por otro lado, la clasificación de un alumno como potencial abandono o no abandono para las mismas titulaciones. Para ello, nos valdremos de variables relacionadas con las calificaciones del alumno en la Prueba de Acceso a la Universidad (PAU) y otras que se irán explicando a lo largo del trabajo. Se estudiará la capacidad de las Redes Neuronales Artificiales para la resolución de estas dos problemáticas, realizando una introducción a esta herramienta de modo general.

ÍNDICE

1. Capítulo I: Introducción a la Redes Neuronales Artificiales. 5
1.1. Historia. 6
1.2. Conceptos y clasificación. 8
1.2.1. Definición de Red Neuronal Artificial (RNA). 8
1.2.2. Fundamentos biológicos de las redes neuronales artificiales. 8
1.2.3. Elementos básicos de una red neuronal y estructura. 9
1.2.4. Clasificación de redes neuronales. Variantes. 11
1.3. Ventajas que ofrecen las redes neuronales. 14
1.4. Aplicaciones. 14
1.4.1. Aplicaciones generales. 14
1.4.2. Aplicaciones específicas. 15
2. Capítulo II: Metodología de resolución de problemas. 17
2.1. Resolución de un problema mediante RNA. Metodología. Matlab. 19
2.1.1. Recoger y preparar los datos. 19
2.1.2. División de los datos en tres grupos. 20
2.1.3. Creación, configuración y entrenamiento de la red. 21
2.1.4. Criterios de parada de entrenamiento. 33
2.2. Análisis de la influencia de las variables. 34
2.2.1. Análisis basado en la magnitud de los pesos. 34
2.2.2. Análisis de sensibilidad. 35
2.3. Ejemplos. 40
3. Capítulo III: Base de datos. 41
3.1. Variables para la predicción/clasificación de la nota media de egreso. 43
3.1.1. Variables PAU. 43
3.1.2. Datos de las asignaturas más complejas de cada titulación. 43
3.2. Variables para la clasificación de abandono. 44
3.2.1. Variables PAU. 44
3.2.2. Variables del primer año de matrícula. 45
3.2.3. Variables del segundo año de matrícula. 46
4. Capítulo IV: Estudio de alumnos egresados. 47
4.1. Clasificación. 48

4.1.1. Estudio para alumnos egresados de Ingeniería Industrial. 48
4.1.2. Análisis y conclusiones de los resultados de clasificación..... 54
4.2. Ajuste de funciones..... 60
4.2.1. Estudio para alumnos egresados de Ingeniería Industrial. 60
4.2.2. Conclusiones para ajuste de funciones. 64
4.3. Regresión lineal múltiple. 66
4.3.1. Estudio para alumnos egresados de Ingeniería Industrial. 66
5. Capítulo V: Estudio y predicción de tasa de abandono. 73
5.1. Objetivo..... 74
5.2. Clasificación a partir de variables PAU. 75
5.2.1. Alumnos de ingeniería industrial..... 75
5.3. Clasificación con variables de primer curso..... 80
5.3.1. Alumnos de Ingeniería Industrial. 80
5.4. Clasificación con variables de segundo curso..... 85
5.4.1. Abandono en Ingeniería Industrial..... 85
5.5. Análisis de la influencia de las variables. 90
5.5.1. Influencia de las variables PAU para Ingeniería Industrial. 91
5.5.2. Influencia de las variables de primer curso para Ingeniería Industrial. ... 93
6. Capítulo VI: Conclusiones generales. 96
6.1. Predicción de la nota de egreso..... 97
6.2. Clasificación de abandonos..... 98
6.3. Líneas de investigación. Actualidad. 99
7. Capítulo VII: Bibliografía..... 101
8. Capítulo VIII: Anexos. 103
8.1. Ejemplos de aplicación en Matlab. 104
8.1.1. Ejemplos para ajuste de funciones..... 104
8.1.2. Ejemplos para búsqueda de patrones y clasificación..... 116
8.2. Análisis para Ingeniería de Telecomunicaciones. 136
8.2.1. Estudio de nota media de alumnos egresados para Ingeniería de Telecomunicaciones. 136
8.2.2. Estudio del abandono en Ingeniería de Telecomunicaciones. 148

1. Capítulo I: Introducción a la Redes Neuronales Artificiales.

Los problemas a los que el hombre se ha enfrentado a lo largo de la Historia y su afán por la mejora de la calidad de vida han supuesto y siguen suponiendo en la actualidad el punto de partida para el desarrollo de muchos avances. Estos avances repercuten de manera crucial en el día a día de las personas en aspectos como la comodidad, el ahorro de tiempo y recursos o el bienestar. En este sentido, el ser humano se ve obligado a buscar soluciones a problemas mediante la toma de decisiones y a asumir las responsabilidades derivadas de tales decisiones. Una de las múltiples formas con que el hombre ha hecho frente a este triángulo "problema – decisión –responsabilidad", es la experiencia, lo que ya ha ocurrido. Y este concepto lo habremos de tener muy presente a lo largo del desarrollo de este trabajo, sobre todo de cara a las conclusiones.

¿Cómo podemos tomar decisiones en base a la experiencia con perspectiva y fiabilidad? Todo depende del problema que se esté abordando, de la información que se disponga y de la forma en que esa información se procese pero, en una primera aproximación, podemos decir que en esta vía de la "experiencia" de lo que se trata es de buscar patrones o tendencias para generar una "máquina" que nos permita predecir o reproducir comportamientos con un grado de incertidumbre asociado.

Sobre este eje conceptual girará nuestro trabajo, en el que nos valdremos de una herramienta matemática para la resolución de un problema. Dicha herramienta recibe el nombre de Redes Neuronales Artificiales (RNA) y surge de la intención del ser humano por generar una "máquina con memoria" de estructura similar a las conexiones neuronales del cerebro con el objetivo mencionado anteriormente: predecir o reproducir patrones a partir de datos.

1.1. Historia.

La historia de las redes neuronales comienza a principios del siglo XX. Veamos algunos de los eventos más importantes de su desarrollo, señalados por Damián Jorge Matich en [1], referencia en la que nos basamos para distintos puntos de esta introducción:

1936 - Alan Turing fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, quienes, en 1943, lanzaron "Un Cálculo Lógico de la Inminente Idea de la Actividad Nerviosa - Boletín de Matemática Biofísica 5: 115-133". Ellos modelaron una red neuronal simple mediante circuitos eléctricos.

1949 - Donald Hebb escribió un importante libro: "La organización del comportamiento", en el que se establece una conexión entre psicología y fisiología. Su idea fue que el aprendizaje ocurría cuando ciertos cambios en una neurona eran activados. También intentó encontrar semejanzas entre el aprendizaje y la actividad nerviosa. Los trabajos de Hebb formaron las bases de la Teoría de las Redes Neuronales.

1950 - Karl Lashley. En sus series de ensayos, encontró que la información no era almacenada en forma centralizada en el cerebro sino que era distribuida encima de él.

1956 - Congreso de Dartmouth. Este Congreso frecuentemente se menciona para indicar el nacimiento de la inteligencia artificial.

1957 - Frank Rosenblatt comenzó el desarrollo del Perceptrón. Esta es la red neuronal más antigua; utilizándose hoy en día para aplicación como reconocedor de patrones. Sin embargo, es incapaz de clasificar clases no separables linealmente.

1960 - Bernard Widrow/Marcial Hoff desarrollaron el modelo Adaline (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) que se ha utilizado comercialmente durante varias décadas.

1961 - Karl Steinbeck: Die Lernmatrix. Red neuronal para simples realizaciones técnicas (memoria asociativa).

1967 - Stephen Grossberg realizó una red: Avalancha, que consistía en elementos discretos con actividad que varía en el tiempo que satisface ecuaciones diferenciales continuas, para resolver actividades como reconocimiento continuo de habla y aprendizaje de los brazos de un robot.

1969 - Marvin Minsky/Seymour Papert. En este año surgieron críticas que frenaron, hasta 1982, el crecimiento que estaban experimentando las investigaciones sobre redes neuronales. Minsky y Papera, del Instituto Tecnológico de Massachussets (MIT) probaron (matemáticamente) que el Perceptrón era muy débil. A pesar de ello James Anderson desarrolló un modelo lineal, llamado Asociador Lineal, que consistía en unos elementos integradores lineales (neuronas) que sumaban sus entradas.

1974 - Paul Werbos desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (backpropagation); cuyo significado quedó definitivamente aclarado en 1985.

1977 - Stephen Grossberg. Teoría de Resonancia Adaptada (TRA), que simula otras habilidades del cerebro: memoria a largo y corto plazo.

1980 - Kunihiko Fukushima desarrolló un modelo neuronal para el reconocimiento de patrones visuales.

1985 - John Hopfield provocó el renacimiento de las redes neuronales con su libro: "Computación neuronal de decisiones en problemas de optimización".

1986 - David Rumelhart/G. Hinton redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation). A partir de 1986, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales. En la actualidad, son numerosos los trabajos que se realizan y publican cada año, las aplicaciones nuevas que surgen (sobre todo en el área de control) y las empresas que

lanzan al mercado productos nuevos, tanto hardware como software (sobre todo para simulación).

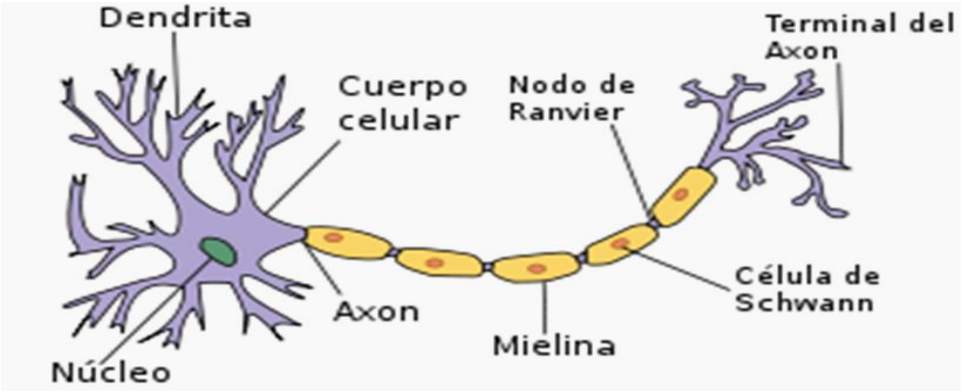
1.2. Conceptos y clasificación.

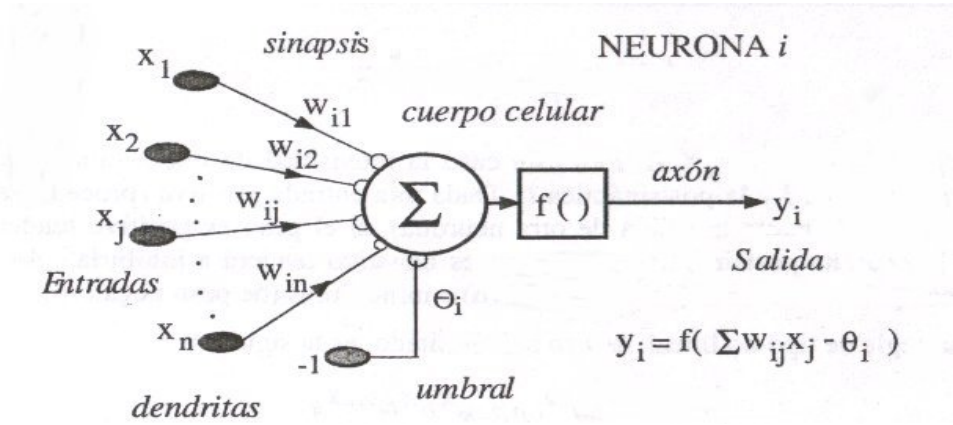
1.2.1. Definición de Red Neuronal Artificial (RNA).

Aunque el concepto engloba en sí muchas posibilidades, en el marco en que nos encontramos podemos definir Red Neuronal Artificial como sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

1.2.2. Fundamentos biológicos de las redes neuronales artificiales.

Como se mencionó en la introducción, la idea que supone el punto de partida para la creación de esta herramienta nace de la estructura del cerebro humano, de las conexiones que se generan. En las siguientes dos imágenes podemos ver, por un lado, las distintas partes de una neurona biológica y, por otro, las partes de una neurona artificial:





El cerebro consta de un gran número de neuronas (aproximadamente 10^{11}) altamente interconectadas (aproximadamente 10^4 conexiones por neurona). Tres son los elementos de las neuronas de los que emana conceptualmente la herramienta que en este trabajo nos ocupa:

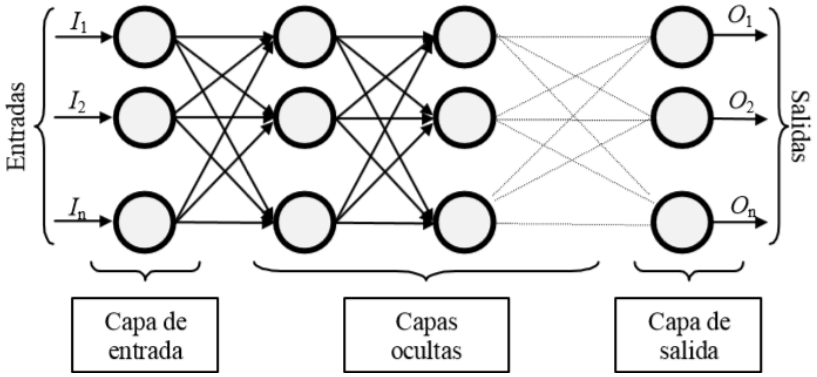
-Dendritas: Las dendritas son prolongaciones protoplásmicas ramificadas, bastante cortas de la neurona dedicadas principalmente a la recepción de estímulos y, secundariamente, también a la alimentación celular. Son terminales de las neuronas; y sirven como receptores de impulsos nerviosos provenientes desde un axón perteneciente a otra neurona. Serían por tanto, en la analogía matemática, las entradas.

-Cuerpo celular: contiene al núcleo y se encarga del procesado de la información para obtener una salida. Tiene asociado un umbral que supone la activación de la salida. A nivel biológico, estos procesos son el resultado de eventos electroquímicos.

-Axón: El axón, cilindroeje o neurita es una prolongación de las neuronas especializadas en conducir el impulso nervioso desde el cuerpo celular o soma hacia otra célula. En la analogía matemática, sería la salida de nuestra neurona.

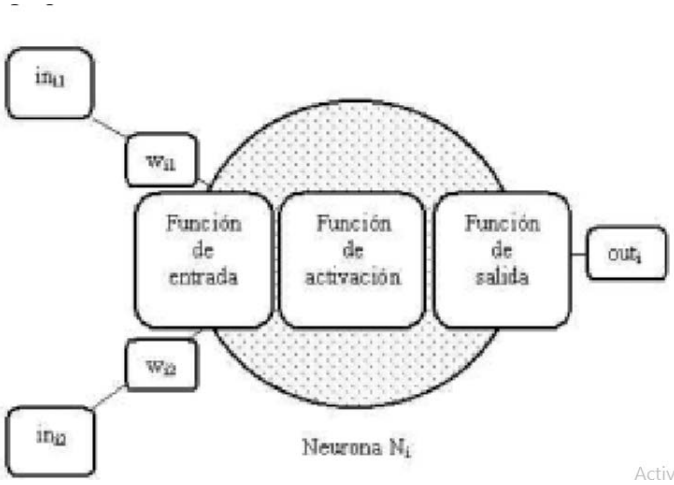
1.2.3. Elementos básicos de una red neuronal y estructura.

En la siguiente imagen podemos ver los principales elementos que componen una red neuronal artificial:



Como podemos observar en la imagen, una red neuronal es una combinación ordenada de neuronas en la que tenemos una serie de capas que en función de su posición son denominadas de un determinado modo: las capas de entrada y salida son siempre únicas, mientras que el número de capas ocultas puede variar. En la imagen se puede ver cómo ante un conjunto de entrada propuesto, la red genera un conjunto de salida.

En el siguiente esquema podemos ver a modo de ejemplo una neurona con dos entradas y una salida, distinguiéndose diversas partes en función de la actividad matemática que en ella se desarrolla: función de entrada, función de activación y función de salida.



- Función de entrada: Las entradas de la neurona modificadas por los pesos dan lugar mediante esta función a un valor único que será procesado en una función posterior.

$$input_i = (in_1 \cdot w_1) * (in_2 \cdot w_2) * \ldots * (in_n \cdot w_n)$$

donde * establece la función de entrada utilizada.

- Función de activación: calcula el estado de actividad de una neurona, transformando la entrada global (menos el umbral, θ_i) en un valor (estado) de activación, cuyo rango normalmente va de [0 a 1] o de [–1 a 1].

- Función de salida: El valor procedente de la función de activación atraviesa esta función que genera el valor de salida de la neurona.

1.2.4. Clasificación de redes neuronales. Variantes.

En este apartado de lo que se trata es de ofrecer una visión de la versatilidad de esta herramienta y de su capacidad de adaptación al problema propuesto así como de desarrollar una guía que permita comprender la gran variedad de redes neuronales que nos podemos encontrar y las decisiones que habremos de tomar en lo que a la estructura de la red se refiere, en función del problema y las partes de la red.

1.2.4.1. En función del problema:

-**Redes heteroasociativas**: La red aprende parejas de datos $[(A_1, B_1), (A_2, B_2) \dots, (A_n, B_n)]$, de tal forma que cuando se presente cierta información de entrada A_i, deberá responder generando la correspondiente salida asociada B_i.

-**Redes autoasociativas**: La red aprende ciertas informaciones A_1, A_2, … de manera que ante una entrada, se responde con la información A_1, A_2, … que más se le parezca.

1.2.4.2. Número de capas.

- Monocapa: Una sola capa.

- Multicapa: Varias capas de neuronas.

1.2.4.3. Conexión entre neuronas.

- Conexión hacia adelante: Es aquella en la que la salida de una neurona nunca es entrada de neuronas del mismo nivel o precedentes. Las redes de propagación hacia atrás (Backpropagation) siguen este tipo de conexión de manera que el ellas, durante el entrenamiento, la modificación de los pesos se produce mediante la propagación del error desde la capa de salida hacia atrás.

- Conexión hacia atrás (sistemas recurrentes): Es aquella en la que la salida de una neurona es entrada de neuronas del mismo nivel o precedentes. Las redes Hopfield usan este tipo de conexiones.

1.2.4.4. Tipo de neuronas.

- **Función de entrada**. Entre las más habituales encontramos:

1) Sumatorio de las entradas pesadas: es la suma de todos los valores de entrada a la neurona, multiplicados por sus correspondientes pesos.

$$\sum_j (in_n \cdot w_n) \;\; con\; j = 1, 2, \dots, n$$

2) Productorio de las entradas pesadas: es el producto de todos los valores de entrada a la neurona, multiplicados por sus correspondientes pesos.

$$\prod_j (in_n \cdot w_n) \;\; con\; j = 1, 2, \dots, n$$

3) Máximo de las entradas pesadas: solamente toma en consideración el valor de entrada más fuerte, previamente multiplicado por su peso correspondiente.

$$Max_j \; (in_n \cdot w_n) \;\; con\; j = 1, 2, \dots, n$$

- **Función de activación**: La función de activación calcula el estado de actividad de una neurona. Las más habituales son: función lineal, función sigmoidea y función hiperbólica.

1) Función lineal.

$$f(x) = \begin{cases} -1 \; si \; (input_i - \theta_i) \leq -\dfrac{1}{a} \\ a * (input_i - \theta_i) \;\; si \; -\dfrac{1}{a} < (input_i - \theta_i) < \dfrac{1}{a} \\ 1 \; si \; (input_i - \theta_i) \geq \dfrac{1}{a} \end{cases}$$

Donde θ_i es el umbral y a es un número positivo de cuyo valor depende la pendiente de la función.

2) Función sigmoidea

$$f(x) = \frac{1}{1 + e^{-a \cdot (input_i - \theta_i)}}$$

3) Función tangente hiperbólica.

$$f(x) = \frac{e^{a\cdot(input_i-\theta_i)} - e^{-a\cdot(input_i-\theta_i)}}{e^{a\cdot(input_i-\theta_i)} + e^{-a\cdot(input_i-\theta_i)}}$$

- **Función de salida**: Dos de las funciones de salida más comunes son:

1) Ninguna: este es el tipo de función más sencillo, tal que la salida es la misma que la entrada. Es también llamada función identidad.

2) Binaria:

$$Binaria \begin{cases} 1 & si\ f(x) \geq \xi_i \\ 0\ de\ lo\ contrario \end{cases}$$

donde ξ_i es el umbral.

1.2.4.5. Mecanismos de aprendizaje.

- **Online:** La red aprende durante su funcionamiento habitual. En las redes con aprendizaje on line no se distingue entre fase de entrenamiento y de operación, de tal forma que los pesos varían dinámicamente siempre que se presente una nueva información del sistema. En este tipo de redes, debido al carácter dinámico de las mismas, el estudio de la estabilidad suele ser un aspecto fundamental del estudio
- **Offline:** El aprendizaje supone que la red está desconectada de sus funciones. Cuando el aprendizaje es off line, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba que serán utilizados en la correspondiente fase. En las redes con aprendizaje off line, los pesos de las conexiones permanecen fijos después que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento
- **Supervisado:** Un agente externo controla el aprendizaje y permite la modificación de los pesos en caso de error en la salida. El aprendizaje supervisado puede ser por corrección de error, por refuerzo o estocástico.
- **No supervisado**: La corrección de los pesos no depende de un agente externo. Puede ser de dos tipos:
    - Hebbiano (Para neuronas binarias): Si dos neuronas Ni y Nj toman el mismo estado simultáneamente (activas o inactivas) el peso de la conexión entre ambas se incrementa.

- Competitivo y comparativo: Clasifica los datos de entrada modificando pesos asociados a patrones o ajustándolos para reconocer nuevos patrones.

1.3. Ventajas que ofrecen las redes neuronales.

- **Aprendizaje adaptativo**. Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial. En el proceso de aprendizaje, los enlaces ponderados de las neuronas se ajustan de manera que se obtengan ciertos resultados específicos.
- **Auto-organización**. Una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje. Cuando las redes neuronales se usan para reconocer ciertas clases de patrones, ellas autoorganizan la información usada. Por ejemplo, la red llamada backpropagation, creará su propia representación característica, mediante la cual puede reconocer ciertos patrones. Esta autoorganización provoca la generalización: facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a las que no había sido expuesta anteriormente.
- **Tolerancia a fallos**. Por un lado Las redes pueden aprender a reconocer patrones con ruido, distorsionados o incompletos. Esta es una tolerancia a fallos respecto a los datos. Por otro lado, las redes pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red.
- **Operación en tiempo real**. Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.
- **Fácil inserción dentro de la tecnología existente**. Se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilitará la integración modular en los sistemas existentes.

1.4. Aplicaciones.

1.4.1. Aplicaciones generales.

1.4.1.1. Asociación y clasificación.

En esta aplicación, los patrones de entrada estáticos o señales temporales deben ser clasificadas o reconocidas. Idealmente, un clasificador debería ser entrenado para que cuando se le presente una versión distorsionada ligeramente del patrón, pueda ser

reconocida correctamente sin problemas. De la misma forma, la red debería presentar cierta inmunidad contra el ruido, esto es, debería ser capaz de recuperar una señal "limpia" de ambientes o canales ruidosos. Esto es fundamental en las aplicaciones holográficas, asociativas o regenerativas.

1.4.1.2. Regeneración de patrones.

En muchos problemas de clasificación, una cuestión a solucionar es la recuperación de información, esto es, recuperar el patrón original dada solamente una información parcial. Hay dos clases de problemas: temporales y estáticos. El uso apropiado de la información contextual es la clave para tener éxito en el reconocimiento.

1.4.1.3. Regeneración y generalización.

El objetivo de la generalización es dar una respuesta correcta a la salida para un estímulo de entrada que no ha sido entrenado con anterioridad. El sistema debe inducir la característica saliente del estímulo a la entrada y detectar la regularidad. Tal habilidad para el descubrimiento de esa regularidad es crítica en muchas aplicaciones. Esto hace que el sistema funcione eficazmente en todo el espacio, incluso cuando ha sido entrenado por un conjunto limitado de ejemplos.

1.4.1.4. Optimización.

Las Redes Neuronales son herramientas interesantes para la optimización de aplicaciones, que normalmente implican la búsqueda del mínimo absoluto de una función de energía. Para algunas aplicaciones, la función de energía es fácilmente deducible; pero en otras, sin embargo, se obtiene de ciertos criterios de coste y limitaciones especiales.

1.4.2. Aplicaciones específicas.

Las redes neuronales pueden utilizarse en un gran número y variedad de aplicaciones, tanto comerciales como militares. Cuando se implementan mediante hardware (redes neuronales en chips VLSI), presentan una alta tolerancia a fallos del sistema y proporcionan un alto grado de paralelismo en el procesamiento de datos. Esto posibilita la inserción de redes neuronales de bajo coste en sistemas existentes y recientemente desarrollados. Hay muchos tipos diferentes de redes neuronales; cada uno de los cuales tiene una aplicación particular más apropiada. Algunas aplicaciones comerciales son:

- Biología: Permiten aprender más acerca del cerebro y otros sistemas y la obtención de modelos de la retina.
- Empresa: evaluación de probabilidad de formaciones geológicas y petrolíferas; identificación de candidatos para posiciones específicas; explotación de bases de datos, optimización de plazas y horarios en líneas de vuelo, optimización del flujo del tránsito controlando convenientemente la temporización de los semáforos, reconocimiento de caracteres escritos, modelado de sistemas para automatización y control.
- Medio ambiente: analizar tendencias y patrones; previsión del tiempo.
- Finanzas: previsión de la evolución de los precios, valoración del riesgo de los créditos, identificación de falsificaciones, interpretación de firmas.
- Manufacturación: robots automatizados y sistemas de control (visión artificial y sensores de presión, temperatura, gas, etc.), control de producción en líneas de procesos; inspección de la calidad.
- Medicina: analizadores del habla para ayudar en la audición de sordos profundos, diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalogramas, análisis sanguíneo, etc.), monitorización en cirugías, predicción de reacciones adversas en los medicamentos, entendimiento de la causa de los ataques cardíacos.
- Militares: clasificación de las señales de radar, creación de armas inteligentes, optimización del uso de recursos escasos, reconocimiento y seguimiento en el tiro al blanco.

La mayoría de estas aplicaciones consisten en realizar un reconocimiento de patrones, por ejemplo: buscar un patrón en una serie de ejemplos, clasificar patrones, completar una señal a partir de valores parciales o reconstruir el patrón correcto partiendo de uno distorsionado. Sin embargo, está creciendo el uso de redes neuronales en distintos tipos de sistemas de control. Desde el punto de vista de los casos de aplicación, la ventaja de las redes neuronales reside en el procesado paralelo, adaptativo y no lineal.

2. Capítulo II: Metodología de resolución de problemas.

Una vez introducidas las generalidades de las RNA, damos un paso más. El presente capítulo está dedicado a la aplicación práctica de esta herramienta. Mediante problemas sencillos presentaremos el Toolbox de Matlab (Hudson Beale, T. Hagan, & B. Demuth, 2010) [2] a grandes rasgos, haciendo especial énfasis en aquellos aspectos que, por la aplicación que en este trabajo se le dará, son de mayor interés: clasificación y ajuste de funciones.

En el capítulo anterior hablábamos de los elementos más importantes de las redes neuronales y sus variantes principales. La versatilidad de esta herramienta es tan grande que intentar condensarla en unas pocas páginas es prácticamente imposible ya que a lo largo de los años se han desarrollado una cantidad impresionante de algoritmos de aprendizaje que son más o menos útiles en función de la aplicación y que el "Toolbox" de Matlab incluye.

La guía del usuario del Matlab sobre redes neuronales ofrece una explicación avanzada sobre las distintas funciones y algoritmos desarrollados, de donde cabe destacar:

Capítulo 1: Comienzo. Se ofrece una visión general aplicada de las redes neuronales y presenta la interfaz gráfica del Toolbox de Matlab, que da la posibilidad de seleccionar la aplicación para la que vamos a utilizar nuestra red neuronal y establecer así por defecto una serie de parámetros y funciones que permiten optimizar a priori recursos de tiempo y memoria:

-"Ajuste de curvas".

-"Reconocimiento de patrones y clasificación".

-"Agrupamiento o *clustering*".

-"Series temporales".

Capítulo 2. Datos y estilos de entrenamiento. Lo más importante de este capítulo es la diferenciación que se establece entre redes estáticas y dinámicas y entre entrenamiento incremental y "en lotes". Las redes dinámicas se caracterizan por la presencia en ellas de realimentación, lo que implica la existencia de pesos que vinculan una determinada neurona, con neuronas de su misma capa. Los pesos de las redes estáticas, en cambio, únicamente unen las neuronas de una determinada capa, con neuronas de capas posteriores. El entrenamiento incremental es aquel en el que los pesos se actualizan con la introducción de cada uno de los ejemplos. En el entrenamiento *batch* o "en lotes", los pesos se actualizan tras la introducción de un número concreto de ejemplos.

Capítulo 3. Redes multicapa y entrenamiento de retropropagación. Quizá sea este uno de los capítulos más importantes de cara a este trabajo porque ofrece una secuencia de actuación de cara a la resolución de problemas mediante la retropropagación, desde la

preparación de los datos, hasta el análisis de los resultados, pasando por la exposición de los distintos algoritmos que podemos usar dependiendo de la situación.

Capítulo 12. Funciones de referencia. Este capítulo muestra la amplia gama de funciones que podemos usar, su aplicación dentro de las redes neuronales y su forma de actuación:

Funciones de datos, funciones de distancia, funciones de la interfaz gráfica, funciones de inicialización de capa, funciones de aprendizaje, funciones de búsqueda de línea, funciones de entrada a la red, funciones de inicialización de red, funciones de nuevas redes, funciones de optimización, funciones de representación, funciones de proceso, funciones para Simulink, funciones de topología, funciones de entrenamiento, funciones de transferencia, funciones de inicialización para pesos y sesgo, funciones de peso.

2.1. Resolución de un problema mediante RNA. Metodología. Matlab.

La metodología a emplear para la resolución de un problema mediante RNA es, en lo que a aspectos generales se refiere, independiente de la naturaleza del problema en cuestión. Sin embargo, existen particularidades, puntos clave en los que unos problemas difieren de otros como es, por ejemplo, el algoritmo de aprendizaje empleado. En este apartado trataremos de presentar dichas similitudes y diferencias y se describen los pasos a seguir para resolver el problema.

2.1.1. Recoger y preparar los datos.

El objetivo de esta primera etapa es, en base al conjunto de variables disponibles y a los resultados que queremos obtener, hacer una primera selección de aquellas variables que consideramos de mayor importancia en la salida de nuestra red. Esta decisión es importante en la medida en que un número alto de variables podría ayudar a obtener mejores predicciones en lo que al encuentro de nuestros patrones se refiere (si es esa, por ejemplo, la aplicación que se le desea dar) pero puede suponer también un costo computacional alto. El proceso de optimización de tiempo y fiabilidad de este tipo de herramientas se basa en una decisión inicial fundamentada en una serie de restricciones y cuestiones: ¿Qué error estoy dispuesto a asumir? ¿Hasta qué punto el tiempo es un factor crítico? La respuesta a estas preguntas supone el punto de partida de un proceso de realimentación de tal manera que mediante la observación de los resultados habremos de ver si es interesante o no aumentar el número de variables o incluso

reducirlo, buscando ahorrar tiempo a cambio de una reducción controlada de la fiabilidad.

Es importante destacar que las bases de datos utilizadas en general en problemas de redes neuronales son bastante amplias. Ello implica la necesidad de llevar a cabo un buen análisis de los mismos para evitar errores asociados a aspectos como campos sin rellenar, cifras incoherentes, mala colocación de comas o puntos decimales, etcétera. Generamos así una matriz "input" con ejemplos y variables.

Tras esta preparación es necesario que los valores numéricos de variables para cada uno de los ejemplos sean introducidos mediante algún tipo de función dentro de un intervalo teniendo en cuenta la función de activación que usaremos en la neurona. Ello lo que permite es que la consideración de la influencia de una determinada variable o característica se haga de manera proporcional y equitativa, independientemente del valor numérico que lleve asociado. En principio, la función de activación que usaremos nosotros será la tangente sigmoidea y lo que esta función requiere es que introduzcamos en ella valores situados en el intervalo [-1,1].

La función que implementará por defecto el toolbox de Matlab para la aplicación que nosotros seleccionaremos es "mapminmax", que precisamente, acota en [-1,1] los valores numéricos de nuestros "inputs".

2.1.2. División de los datos en tres grupos.

De todos los ejemplos que tengamos habremos de seleccionar tres grupos:

-Grupo de entrenamiento: Supone el 60% del total de los ejemplos disponibles. Se utiliza para entrenar la red, es decir, permiten que los pesos varíen en función de sus características y valores.

-Grupo de validación: Supone el 20% del total de los ejemplos disponibles. Se utiliza para ajustar ciertos valores de parámetros asociados al entrenamiento.

-Grupo de test: Supone el 20% del total de los ejemplos disponibles. Se utiliza para comprobar la fiabilidad de la red creada.

Los porcentajes señalados anteriormente son orientativos y pueden ser variados en caso de considerarse necesario. Comúnmente, también se suele utilizar un 70% de los datos para entrenar la red, un 15% para validación, y el 15% restante para test.

Para la selección de los ejemplos que pertenecerán a cada uno de los grupos se usará una función random que selecciona aleatoriamente de acuerdo con los porcentajes marcados.

2.1.3. Creación, configuración y entrenamiento de la red.

Existen múltiples maneras de llevar a cabo este proceso. Para el caso de redes multicapa entrenadas por retropropagación una forma sería introducir el siguiente código:

net=feedforwardnet;

net=configure(net, X,y);

net=init(net);

El primer comando crea la red, el segundo la configura otorgándole a los parámetros valores por defecto que podemos modificar y establece aleatoriamente unos valores para los pesos y los sesgos dentro de un rango fijando, y el tercer comando reinicializa los valores asociados a pesos y sesgos.

Sin embargo, teniendo clara la naturaleza del problema a resolver, podemos condensar estos tres comandos en uno solo con el que además, definimos la arquitectura de la red y una serie de funciones y parámetros que, a priori, optimizan el proceso. Los problemas que se abordarán a lo largo de este trabajo serán o bien de ajuste de funciones o bien de clasificación.

Para el problema de clasificación usaremos el siguiente comando:

net =patternnet(hiddenLayerSize,'trainscg');

donde *hiddenLayerSize* es la variable que almacena el número de neuronas de la capa oculta mediante un escalar si es única o en forma de vector si es múltiple y *trainscg* es la función de entrenamiento.

Para el problema de ajuste de funciones usaremos el siguiente comando:

net = fitnet(hiddenLayerSize);

donde *hiddenLayerSize* representa lo mismo que para el caso de clasificación. En el problema de ajuste no es necesario seleccionar una función de entrenamiento concreta pues Matlab toma, por defecto, *trainlm*.

El entrenamiento que se establece por defecto es el *batch*, ya explicado anteriormente. De acuerdo con la guía del tooblox de Matlab, es más rápido y genera menos errores que el entrenamiento incremental.

Como también se ha señalado anteriormente la técnica que usaremos nosotros para el entrenamiento de la red es la retropropagación, dentro de la cual hay distintos algoritmos de entrenamiento:

Función	Descripción
`trainb`	Entrenamiento *batch* con reglas de entrenamiento para pesos y umbrales
`trainbfg`	*BFGS quasi-Newton backpropagation*
`trainbfgc`	*BFGS quasi-Newton backpropagation* para controladores adaptativos de referencia de modelos ANN
`trainbr`	Regularización bayesiana
`trainbuwb`	Entrenamiento *batch* no supervisado para pesos y umbrales
`trainc`	Actualización incremental de orden cíclico
`traincgb`	*Powell-Beale backpropagation* de gradiente conjugado
`traincgf`	*Fletcher-Powell backpropagation* de gradiente conjugado
`traincgp`	*Polak-Ribiére backpropagation* de gradiente conjugado
`traingd`	*Backpropagation* por descenso de gradiente
`traingda`	*Backpropagation* por descenso de gradiente con reglas de entrenamiento adaptativas
`traingdm`	*Backpropagation* por descenso de gradiente con momento
`traingdx`	*Backpropagation* por descenso de gradiente con momento y reglas de entrenamiento adaptativas
`trainlm`	*Levenberg-Marquardt backpropagation*
`trainoss`	*One step secant backpropagation*
`trainr`	Entrenamiento incremental de orden aleatorio con funciones de aprendizaje
`trainrp`	*Resilient backpropagation* (Rprop)
`trains`	Entrenamiento incremental de orden secuencial con funciones de aprendizaje
`trainscg`	*Backpropagation* por descenso de gradiente conjugado y escalado

Dependiendo de la aplicación, será mejor utilizar una función u otra. La función de entrenamiento más rápida es *trainlm*. *Trainbfg* también es bastante rápida. Sin embargo, estas funciones son menos eficientes cuanto más grandes son las redes porque necesitan mucha memoria. Funciones mucho más eficientes son *trainscg* o *trainrp*, que no gastan mucha memoria y son más rápidas que el gradiente descendente. En principio, tal y como ya hemos señalado, nosotros usaremos *trainlm* para ajuste de funciones y *trainscg* para clasificación.

2.1.3.1. Algoritmo LM.

El algoritmo Levenberg-Marquardt (Yu & M. Wilamowski) [3] proporciona una solución numérica al problema de minimizar una función no lineal. Es rápido y proporciona una convergencia estable.

En [3] se explica la minimización de la función Suma del Error Cuadrático (SSE) y Matlab emplea como función a minimizar el Error Cuadrático Medio (MSE), aunque en ambos casos el procedimiento algorítmico es análogo.

$$SSE = E(x,w) = \frac{1}{2} \cdot \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2$$

Donde P representa el número de patrones, M representa el número de salidas de la red, *x* es el vector de entrada, *w* representa los vectores de pesos, y $e_{p,m}$ representa el error de entrenamiento de la salida *m* cuando aplicamos el patrón *p* y se define como:

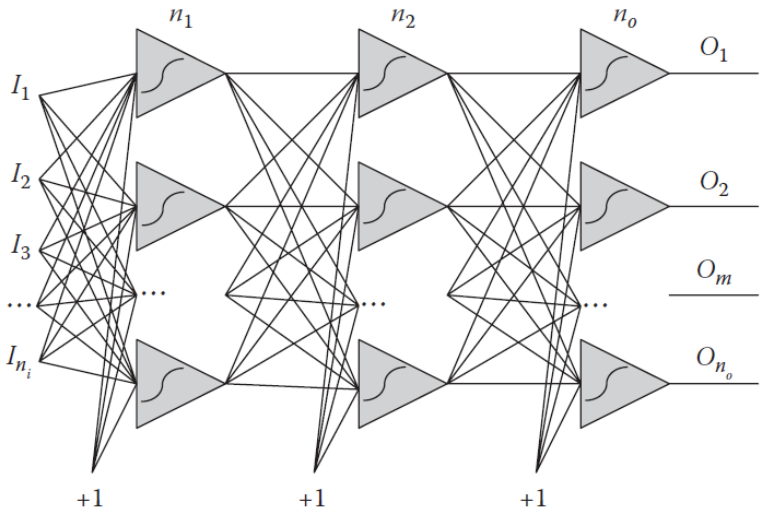
$$e_{p,m} = d_{p,m} - o_{p,m}$$

Donde $d_{p,m}$ es el vector de salida deseado y $o_{p,m}$ el vector de salida actual.

De todos los algoritmos mencionados, en relación con el de Levenberg-Marquardt, cabe destacar el algoritmo del gradiente descendente, que dispersó la oscura nebulosa que se cernía sobre el campo de las redes neuronales artificiales y aún hoy es muy usado. Sin embargo, es un algoritmo ineficiente porque provoca una convergencia lenta. Esta lenta convergencia es mejorada por el algoritmo Gauss-Newton. Usando derivadas de segundo orden de la función de error para evaluar la curvatura de la superficie error, el algoritmo de Gauss-Newton puede encontrar tamaños de paso apropiados para cada iteración y converger bastante rápido. Pero esta mejora solo ocurre cuando la aproximación cuadrática de la función de error es razonable. En otro caso, el algoritmo de Gauss-Newton diverge.

El algoritmo de Levenberg-Marquardt combina el método del gradiente descendente y el algoritmo de Gauss-Newton dando lugar a un algoritmo con la estabilidad de aquel y la rapidez de convergencia de éste. Es más robusto que el algoritmo de Gauss-Newton porque, en muchos casos, converge bien incluso si la superficie de error es mucho más compleja que una función cuadrática. Aunque el algoritmo LM tiende a ser un poco más lento que el algoritmo GN, converge mucho más rápido que el gradiente descendente.

Para explicar el algoritmo LM, nos basaremos en el entrenamiento de la siguiente red, formada por esta red neuronal de tres capas:



Cada iteración del proceso de aprendizaje lleva una etapa de computación hacia adelante, en la que se calcula la salida de la red con los pesos actuales, y una etapa de computación hacia atrás, en la que en base a los errores obtenidos a la salida de la red, se varían los pesos para minimizar la función de error.

En primer lugar describimos los pasos de la computación hacia adelante:

a) Mediante la función de entrada, calculamos un valor que recoge las influencias de todas las entradas a la neurona correspondiente, así como las pendientes y las salidas de todas las neuronas de la primera capa:

$$net_j^1 = \sum_{i=1}^{ni} I_i \cdot w_{j,i}^1 + w_{j,0}^1$$

$$y_j^1 = f_j^1\left(net_j^1\right)$$

$$s_j^1 = \frac{\partial f_j^1}{\partial net_j^1}$$

Donde I_i son las entradas a la red, el superíndice 1 hace referencia a la primera capa y el subíndice j hace referencia a la neurona que se está calculando dentro de la primera capa.

b) Usando las salidas de la primera capa de neuronas como entradas de todas las neuronas de la segunda capa, hacemos un cálculo similar:

$$net_j^2 = \sum_{i=1}^{n_1} y_i^1 \cdot w_{j,i}^2 + w_{j,0}^2$$

$$y_j^2 = f_j^2\left(net_j^2\right)$$

$$s_j^2 = \frac{\partial f_j^2}{\partial net_j^2}$$

c) Y de la misma manera procedemos para la última capa:

$$net_j^3 = \sum_{i=1}^{n_2} y_i^2 \cdot w_{j,i}^3 + w_{j,0}^3$$

$$o_j = f_j^3\left(net_j^3\right)$$

$$s_j^3 = \frac{\partial f_j^3}{\partial net_j^3}$$

Partiendo de los resultados de la computación hacia adelante, iniciamos la computación hacia atrás:

d) Calculamos el error de la salida j y la δ inicial como la pendiente de la salida j:

$$e_j = d_j - o_j$$

$$\delta_{j,j}^3 = s_j^3$$

$$\delta_{j,k}^3 = 0$$

Donde

d_j es la salida deseada para la salida j.
o_j es la actual salida en j obtenida de la computación hacia adelante.
$\delta_{j,j}^3$ es la pendiente asociada a la propia salida.
$\delta_{j,k}^3$ es la pendiente asociada a las otras neuronas de la misma capa.

e) Propagamos hacia atrás δ desde las entradas de la tercera capa hasta las salidas de la segunda capa:

$$\delta_{j,k}^2 = w_{j,k}^3 \cdot \delta_{j,j}^3$$

Donde k es el índice de las neuronas de la segunda capa, desde 1 hasta n_2.

f) Propagamos hacia atrás δ desde las salidas de la segunda capa hasta las entradas de la segunda capa:

$$\delta_{j,k}^2 = s_k^2 \cdot \delta_{j,k}^2$$

Donde k es el índice de las neuronas de la segunda capa, desde 1 hasta n_2.

g) Propagamos hacia atrás δ desde las entradas de la segunda capa hasta las salidas de la primera capa:

$$\delta_{j,k}^1 = \sum_{i=1}^{n_2} w_{j,k}^2 \cdot \delta_{j,j}^2$$

Donde k es el índice de las neuronas de la segunda capa, desde 1 hasta n_1.

h) Propagamos hacia atrás δ desde las salidas de la primera capa hasta las entradas de la primera capa:

$$\delta_{j,k}^1 = s_k^1 \cdot \delta_{j,k}^1$$

Donde k es el índice de las neuronas de la segunda capa, desde 1 hasta n_1.

i) Los correspondientes elementos de fila de la matriz Jacobiana pueden ser calculados usando

$$\frac{\partial e_{p,m}}{\partial w_{j,i}} = -\delta_{m,j} \cdot y_{j,i}$$

Donde $y_{j,i}$ se calcula en la computación hacia adelante (señal propagada desde las entradas hacia las salidas) y $\delta_{m,j}$ es obtenido en la computación hacia atrás desde la capa de salida hasta la capa de entrada. En la neurona de salida m $(j = m), \delta_{m,j} = s_m$.

j) Una vez calculada la matriz Jacobiana estamos en disposición de actualizar los pesos mediante:

$$w_{k+1} = w_k - (J_k^T \cdot J_k + \mu \cdot I)^{-1} \cdot J_k \cdot e_k$$

Donde e_k es el vector de error, I es la matriz identidad y el coeficiente μ recibe el nombre de coeficiente de combinación y es siempre positivo.

En la fórmula de actualización de los pesos se ha considerado la siguiente aproximación para la matriz Hessiana:

$$H \approx J^T \cdot J + \mu \cdot I$$

Además, fijándonos en la mencionada fórmula de actualización de los pesos nos percatamos de que para valores del coeficiente μ de combinación pequeños (próximos a cero), dicha fórmula toma la forma empleada por el algoritmo de Gauss-Newton para actualizar los pesos:

$$w_{k+1} = w_k - (J_k^T \cdot J_k)^{-1} \cdot J_k \cdot e_k$$

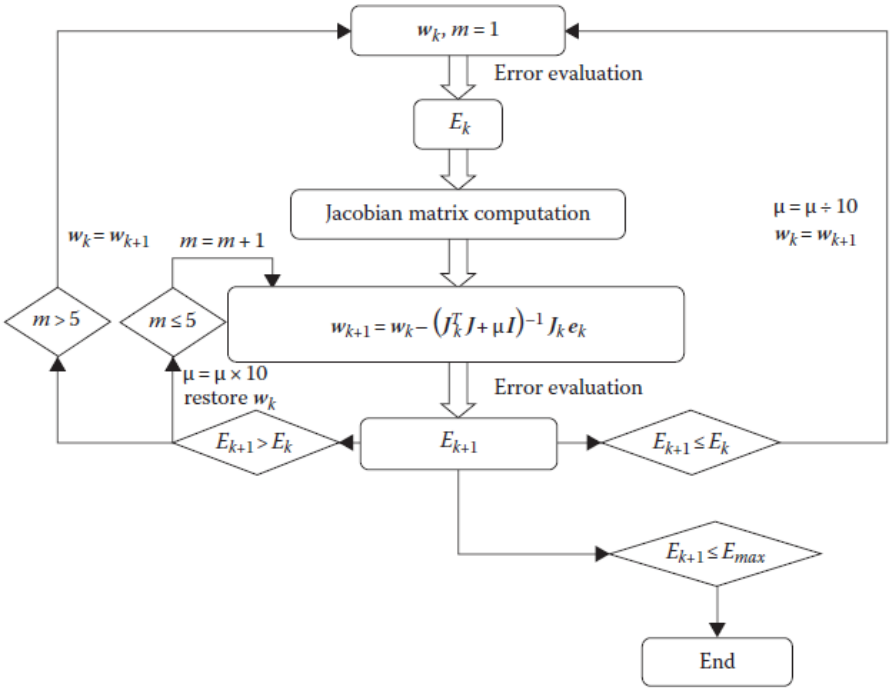
Mientas que si el coeficiente μ tiene valores muy grandes, la fórmula adquiere la forma empleada por el algoritmo del gradiente descendente para actualizar los pesos:

$$w_{k+1} = w_k - \alpha \cdot g_k$$

Donde se puede establecer la siguiente relación con el coeficiente de aprendizaje:

$$\alpha = \frac{1}{\mu}$$

El flujograma que da forma a este algoritmo viene recogido a continuación:



En resumen, el proceso de entrenamiento usando el algoritmo de aprendizaje de Levenberg-Marquardt presenta la siguiente forma:

1) Con los pesos iniciales (generados aleatoriamente), se evalúa el error total.
2) Se actualizan los pesos mediante la fórmula descrita para este algoritmo.
3) Con los nuevos pesos, se vuelve a evaluar el error.
4) Si el error actual ha aumentado como consecuencia de la actualización, entonces se retira el paso dado (restaurando los valores de los pesos de la iteración anterior) y se aumenta el coeficiente μ de combinación mediante un factor de 10 u otro factor distinto. Se vuelve entonces al paso 2 y se actualizan de nuevo los pesos.
5) Si el error actual ha disminuido como resultado de la actualización, entonces se acepta el paso (aceptando los nuevos pesos como los actuales) y se disminuye el coeficiente μ de combinación dividiéndolo entre un factor de 10 u otro (ha de coincidir con el del paso 4).
6) Se vuelve al paso 2 hasta que los nuevos pesos no generen un error total menor que el valor requerido o hasta que otro criterio de parada sea alcanzado.

2.1.3.2. Algoritmo SCG.

Para explicar el proceso de entrenamiento (Fodslette Moller, 1993) [4], primero explicaremos en qué consiste un gradiente conjugado. A continuación, veremos por qué surge la idea de escalarlo.

Varios algoritmos de gradiente conjugado han sido utilizados, como el gradiente descendente estándar con búsqueda de línea (CGL) y BFGS. Sin embargo, estos dos métodos aumentan la complejidad de cálculo por iteración de aprendizaje ya que tienen que realizar una búsqueda de línea con el fin de determinar un tamaño de paso apropiado. Para superar este lastre surge el gradiente conjugado escalado: este algoritmo evita la búsqueda de línea por iteración de aprendizaje mediante el uso de un enfoque de Levenberg-Marquardt (trainlm, ya explicado antes), que permite escalar el tamaño de paso.

Conviene recordar la estrategia general de optimización en redes neuronales vista para el algoritmo de Levenberg-Marquardt:

1) Inicializamos (aleatoriamente, por ejemplo) la matriz de pesos w y ponemos a 1 el contador del número de iteraciones (k=1).
2) Seleccionamos la dirección de búsqueda p_k y el tamaño de paso α_k para conseguir que $E(w_k + \alpha_k \cdot p_k) < E(w_k)$, donde E es la función de error que hemos seleccionado para optimizar. En el caso de Matlab, dicha función es el error cuadrático medio (MSE):

$$MSE = \frac{1}{P \cdot M} \cdot \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2$$

Donde P (diferente de p_k) representa el número de patrones, M representa el número de salidas de la red, y $e_{p,m}$ representa el error de entrenamiento de la salida *m* cuando aplicamos el patrón *p* y se define como:

$$e_{p,m} = d_{p,m} - o_{p,m}$$

Donde $d_{p,m}$ es el vector de salida deseado y $o_{p,m}$ el vector de salida actual.

3) Actualizamos el vector $w_{k+1} = w_k + \alpha_k \cdot p_k$.
4) Si $E'(w_k) \neq 0$, hacemos k=k+1 y volvemos al paso 2.

En el algoritmo del gradiente descendente $p_k = -E'(w)$ y $\alpha_k = constante$. En cambio, en los métodos de dirección de gradiente conjugados tanto el valor de α_k como el de p se van actualizando en las sucesivas iteraciones, para lo que es necesario recurrir a información de la aproximación de segundo orden. Definimos la aproximación cuadrática de E en el entorno de un punto w como:

$$E_{qw}(y) = E(w) + E'(w)^T \cdot y + \frac{1}{2} \cdot y^T \cdot E''(w) \cdot y$$

Dos son los teoremas fundamentales en los que se basan este tipo de algoritmos:

Teorema 1. Sean $p_1, \dots, p_n$ un sistema conjugado e y_1 un punto en el espacio de pesos. Sean los puntos $y_2, \dots y_{N+1}$ definidos iterativamente mediante

$$y_{k+1} = y_k + \alpha_k \cdot p_k$$

Donde $\alpha_k = \frac{\mu_k}{\delta_k}$, $\mu_k = -p^T \cdot E'_{qw}(y_k), \delta_k = p_k^T \cdot E''(w) \cdot p_k$.

Entonces y_{k+1} minimiza E_{qw} restringido al k-plano π_k dado por y_1 y $p_1, \dots, p_k$.

El algoritmo asegura que el mínimo global de la función cuadrática será detectado en al menos N iteraciones. Si todos los valores propios del Hessiano $E''_{qw}(w)$ caen en varios grupos con valores del mismo orden, entonces hay una gran posibilidad de que el algoritmo termine en muchas menos iteraciones.

Teorema 2. Sea y_1 un punto en el espacio de pesos y p_1 y r_1 igual al vector descendente $-E'_{qw}(y_1)$ de más pendiente. Definimos p_{k+1} iterativamente mediante

$$p_{k+1} = r_{k+1} + \beta_k \cdot p_k$$

Donde $r_{k+1} = E'_{qw}(y_{k+1}), \beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T \cdot r_k}{p_k^T \cdot r_k}$ y y_{k+1} es el punto generado mediante el teorema 1.

Entonces p_{k+1} es el vector descendente de más pendiente para E_{qw} restringido al (N-K)-plano π_{k+1} conjugado con π_k dado por y_1 y $p_1, \dots, p_k$.

Partiendo de la estrategia de optimización general, de las definiciones expuestas para gradientes conjugados y de estos dos teoremas, se puede establecer el siguiente proceso iterativo de optimización de los pesos para minimizar el error de la red neuronal:

1) Inicializamos el vector w de pesos. Establecemos inicialmente $p_1 = r_1 = -E'(w)$ y ponemos a uno el contador de iteraciones k=1.
2) Calculamos la información de segundo orden:
$$s_k = E''(W_k) \cdot p_k$$
$$\delta_k = p_k^T \cdot s_k$$
3) Calculamos el tamaño de paso:
$$\mu_k = p_k^T \cdot r_k$$
$$\alpha_k = \mu_k / \delta_k$$
4) Actualizamos el vector de pesos:
$$w_{k+1} = w_k + \alpha_k \cdot p_k$$
$$r_{k+1} = -E'(w_{k+1})$$
5) Si k *mod* N=0, entonces reiniciamos el algoritmo $p_{k+1} = r_{k+1}$, de lo contrario se crea una nueva dirección conjugada:
$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T \cdot r_k}{p_k^T \cdot r_k}$$
$$p_{k+1} = r_{k+1} + \beta_k \cdot p_k$$

6) Si la dirección descendente de mayor pendiente $r_k \neq 0$, hacemos k=k+1 y volvemos al paso dos, comenzando una nueva iteración. Si $r_k = 0$, el proceso iterativo ha terminado y tenemos los pesos w que minimizan el error.

Sin embargo, el gradiente conjugado sólo funciona bien si la matriz Hessiana es definida positiva y si la aproximación cuadrática es buena. De estas limitaciones surge la idea de para desarrollar un gradiente conjugado escalado:

1) Para solucionar el problema asociado al Hessiano se utilizará el enfoque de Levenberg-Marquardt para calcular s_k así:

$$s_k = \frac{E'(w_k + \sigma_k \cdot p_k) - E'(w_k)}{\sigma_k} + \lambda_k \cdot p_k$$

Donde $0 < \sigma_k \ll 1$.

El valor de λ_k se ajusta en cada iteración de acuerdo con δ_k. Si $\delta_k \leq 0$, el hessiano no es definido positivo, por lo que tengo que aumentar el valor de λ_k y estimar s_k otra vez como

$$s'_k = s_k + (\lambda'_k - \lambda_k) \cdot p_k$$

Donde λ'_k es el nuevo valor de λ_k y s'_k es el nuevo valor de s_k.

Pero ¿cuánto debo aumentar λ_k? Pues sabiendo que lo que quiero es un nuevo δ_k, δ'_k, que sea mayor que cero:

$$\delta'_k = p_k^T \cdot s'_k = p_k^T \cdot (s_k + (\lambda'_k - \lambda_k) \cdot p_k) = \delta_k + (\lambda'_k - \lambda_k) \cdot |p_k|^2 > 0$$

Por tanto, en principio, una buena aproximación es:

$$\lambda'_k = 2 \cdot (\lambda_k - \frac{\delta_k}{|p_k|^2})$$

Esto conduce a:

$$\delta'_k = \delta_k + (\lambda'_k - \lambda_k) \cdot |p_k|^2 = \delta_k + \left(2 \cdot \lambda_k - 2 \cdot \frac{\delta_k}{|p_k|^2} - \lambda_k\right) \cdot |p_k|^2$$
$$= -\delta_k + \lambda_k \cdot |p_k|^2$$

Y por tanto el tamaño de paso queda como:

$$\alpha_k = \frac{\mu_k}{\delta_k} = \frac{\mu_k}{p_k^T \cdot s_k + \lambda_k \cdot |p_k|^2}$$

2) Problema de la aproximación cuadrática:

La aproximación cuadrática E_{qw}sobre la cual trabaja el algoritmo puede no ser siempre una buena aproximación para E(w) ya que λ_k escala la matrix Hessiana de una manera artificial. Se necesita, por tanto, un mecanismo para aumentar o disminuir λ_k y conseguir una buena aproximación cuadrática, incluso cuando el Hessiano es definido positivo. Definimos:

$$\Delta_k = \frac{E(w_k) - E(w_k + \alpha_k \cdot p_k)}{E(w_k) + E_{qw}(\alpha_k \cdot p_k)} = \frac{2 \cdot \delta_k \cdot [E(w_k) - E(w_k + \alpha_k \cdot p_k)]}{\mu_k^2}$$

Aquí, Δ_k es una medida de lo bien que $E_{qw}(\alpha_k \cdot p_k)$ aproxima $E(w_k + \alpha_k \cdot p_k)$ en el sentido de que cuanto más cerca de uno esté Δ_k, mejor es la aproximación. λ_k es aumentada y disminuida mediante las siguientes fórmulas:

Si $\Delta_k > 0.75$ entonces $\lambda_k = \frac{1}{4} \cdot \lambda_k$

Si $\Delta_k < 0.25$ entonces $\lambda_k = \lambda_k + \frac{\delta_k \cdot (1 - \Delta_k)}{|p_k|^2}$

La fórmula para $\Delta_k < 0.25$ aumenta lambda de manera que el nuevo tamaño de paso es igual al mínimo para un polinomio cuadrático ajustado por $E'(w_k)^T \cdot p_k, E(w_k)$ y $E(w_k + \alpha_k \cdot p_k)$.

A continuación se describen los paso a seguir para la implementar este algoritmo:

1) Inicializo el vector w_1 de pesos y los escalares: $0 < \sigma \leq 10^{-4}$, $0 < \lambda_k \leq 10^{-6}$, $\lambda_k' = 0$.
   Establezco $p_1 = r_1 = -E'(w_1)$, k=1 y success=true.

2) Si success=true, calculamos la información de segundo orden:

$$\sigma_k = \sigma / |p_k|$$
$$s_k = \frac{E'(w_k + \sigma_k \cdot p_k) - E'(w_k)}{\sigma_k}$$
$$\delta_k = p_k^T \cdot s_k$$

3) Escalamos δ_k: $\delta_k = p_k^T \cdot s_k + \lambda_k \cdot |p_k|^2$

4) Si $\delta_k \leq 0$, hago la matriz hessiana definida positiva:

$$\lambda_k' = 2 \cdot (\lambda_k - \frac{\delta_k}{|p_k|^2})$$

$$\delta_k' = -\delta_k + \lambda_k \cdot |p_k|^2$$

$$\lambda_k = \lambda_k'$$

5) Cálculo del tamaño de paso:

$$\mu_k = p_k^T \cdot r_k$$
$$\alpha_k = \mu_k / \delta_k$$

6) Cálculo del parámetro de comparación:

$$\Delta_k = \frac{2 \cdot \delta_k \cdot [E(w_k) - E(w_k + \alpha_k \cdot p_k)]}{\mu_k^2}$$

7) Si $\Delta_k \geq 0$, se puede conseguir una corrección del error:

$$w_{k+1} = w_k + \alpha_k \cdot p_k$$
$$r_{k+1} = -E'(w_{k+1})$$
$$\lambda_k' = 0$$
$$success = true$$

8) Si k *mod* N=0, recomienza el algoritmo con $p_{k+1} = r_{k+1}$, de lo contrario se crea una nueva dirección conjugada:

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T \cdot r_k}{\mu_k}$$

$$p_{k+1} = r_{k+1} + \beta_k \cdot p_k$$

Si $\Delta_k > 0.75$, reducimos el parámetro de escala: $\lambda_k = \frac{1}{4} \cdot \lambda_k$, de lo contrario: $\lambda_k' = \lambda_k$

$success = false.$

9) Si $\Delta_k < 0.25$ entonces disminuimos el parámetro de escala: $\lambda_k = \lambda_k + \frac{\delta_k \cdot (1 - \Delta_k)}{|p_k|^2}$

10) Si la dirección descendente de mayor pendiente $r_k \neq 0$, entonces hacemos k=k+1 y vamos al paso 2. Si $r_k = 0$, el proceso iterativo ha terminado y hemos alcanzado los valores de los pesos que minimizan el error.

En Matlab ya vienen implementados estos algoritmos, para activarlos y entrenar la red por épocas, simplemente tenemos que ejecutar los siguientes comandos:

a) Para ajuste de funciones:
net = fitnet(hiddenLayerSize);
[net, tr]=train(net, inputs, targets);

b) Para reconocimiento de patrones y clasificación:
net =patternnet(hiddenLayerSize,'trainscg');
[net, tr]=train(net, inputs, targets);

Donde *net* hace referencia a la red creada, *tr* es el conjunto de características de la red entrenada, *trainscg* es la función de entrenamiento, *inputs* son las entradas y *targets* las salidas objetivo.

2.1.4. Criterios de parada de entrenamiento.

Se distinguen cinco criterios distintos para detener el proceso de entrenamiento:

2.1.4.1. Épocas.

El código que permite controlar el número de épocas (iteraciones del entrenamiento) máximo es:

net.trainParam.epochs =… ;

2.1.4.2. Tiempo.

net.trainParam.time=… ;

Permite introducir el máximo tiempo en segundos que deseamos que dure el entrenamiento.

2.1.4.3. Rendimiento.

El código que permite fijar un valor límite para la función de optimización (MSE, en nuestro caso) es:

net.trainParam.goal=… ;

Cuando, tras una iteración, se calcula el error entre la salida de la red y la salida objetivo, y su valor es menor o igual que el establecido mediante este parámetro, el entrenamiento se detiene, pues se considera que se ha alcanzado un valor óptimo.

2.1.4.4. Gradiente.

Para elegir el valor de gradiente a partir del cual se desea que el entrenamiento pare se utiliza:

net.trainParam.min_grad=… ;

Durante el proceso de entrenamiento llega un punto en que el gradiente es tan pequeño que no tiene sentido continuar con el proceso ya que se consumen tiempo y recursos sin conseguir grandes disminuciones del error.

2.1.4.5. Comprobaciones de validación.

Podemos establecer el número de comprobaciones que queremos que se hagan de la red antes de parar el entrenamiento mediante el conjunto de datos de validación. Esta herramienta permite controlar el sobreajuste de la red, que supone una pérdida de generalidad. Se aplica la técnica conocida como 'early stopping': Cuando se observa que tras una serie de iteraciones, el error ha comenzado a crecer, se interrumpe el entrenamiento y se recuperan para los pesos los valores de la iteración que generaba menor error.

net.trainParam.max_fail=… ;

2.2. Análisis de la influencia de las variables.

En general, el mayor esfuerzo en la investigación sobre RNA se ha centrado en el desarrollo de nuevos algoritmos de aprendizaje, nuevas arquitecturas de redes y nuevos campos de aplicación. Sin embargo, se ha dedicado poca atención a desarrollar procedimientos que permitan entender lo que ocurre dentro de la red neuronal. Ésta se ha presentado muchas veces como una especie de "caja negra" cuyo complejísimo trabajo, de alguna forma mágico, transforma las entradas en salidas concretas.

En este apartado se pretende exponer de modo muy genérico una serie de metodologías para ver la influencia de las variables del conjunto de entrada, en las distintas salidas de acuerdo con (Montaño Moreno, 2002) [5]:

2.2.1. Análisis basado en la magnitud de los pesos.

El análisis basado en la magnitud de los pesos agrupa aquellos procedimientos que se basan exclusivamente en los valores almacenados en la matriz estática de pesos con el propósito de determinar la influencia relativa de cada variable de entrada sobre cada una de las salidas de la red. Este tipo de análisis tiene su origen en el examen de los pesos de conexión entre las neuronas de entrada y ocultas. En principio, se decía que las entradas con pesos de valor absoluto más alto eran más importantes. Este método tan rudimentario acabó desechándose pues la presencia de pesos más altos en las conexiones entre la capa de entrada y la primera capa oculta no implican mayor importancia de las variables. Se han propuesto en este sentido expresiones más elaboradas que tienen en cuenta no solo los pesos de conexión entre la capa de entrada y la primera oculta sino también los pesos de conexión entre la capa oculta y la de salida.

A continuación se presenta una de las ecuaciones más utilizadas, la propuesta por Garson:

$$Q_{ik} = \frac{\sum_{j=1}^{L}\left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} v_{jk}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{L}\left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} v_{jk}\right)}$$

Donde w_{ij} hace referencia a los pesos que conectan la capa de entrada con la capa oculta, v_{jk} hace referencia a los pesos que conectan la capa oculta con la capa de salida, $\sum_{r=1}^{N} w_{rj}$ es la suma de los pesos de conexión entre las i neuronas de entrada y la neurona oculta j, L es el número de neuronas de la capa oculta y N es el número de variables de entrada.

En este ecuación debemos tener en cuenta, por una parte, que el valor de los pesos se usa en valor absoluto para que los pesos positivos y negativos no se cancelen y, por otra parte, que el valor umbral de las neuronas ocultas y de salida no se tienen en cuenta, asumiendo que su inclusión no afecta al resultado final. El índice Q_{ik} representa el porcentaje de influencia de la variable de entrada i sobre la salida k, en relación a las demás variables de entrada, de forma que la suma de este índice para todas las variables de entrada debe dar como valor el 100%.

2.2.2. Análisis de sensibilidad.

El análisis de sensibilidad está basado en la medición del efecto que se observa en una salida y_k o en el error cometido debido al cambio que se produce en una entrada x_i. Así, cuanto mayor efecto se observe sobre la salida, mayor sensibilidad podemos deducir que presenta respecto a la entrada. A continuación se presentan diferentes formas de realizar el análisis de sensibilidad:

2.2.2.1. Análisis de sensibilidad basado en el error.

Imaginemos que la función de error que se utiliza es la raíz cuadrada de la media cuadrática del error:

$$RMC_{error} = \sqrt{\frac{\sum_{p=1}^{P}\sum_{k=1}^{M}\left(d_{pk} - y_{pk}\right)^2}{P \cdot M}}$$

Donde d_{pk} es la salida deseada para el patrón p en la neurona de salida k.

La aplicación de este tipo de análisis sobre un conjunto e datos consiste en ir variando el valor de una de las variables de entrada a lo largo de todo su rango mediante la

aplicación de pequeños incrementos, mientras que se mantienen los valores originales de las demás variables de entrada. Una vez aplicados los incrementos a una determinada variables de entrada se procede a entrenar la red neuronal calculando el valor de RMC error. Siguiendo este procedimiento para todas las variables de entrada, se puede establecer una ordenación en cuanto a la importancia sobre la salida. Así , la variable de entrada que proporcione el mayor RMC error será la variable con más influencia en la variable, respuesta, mientras que la variable de entrada con menor RMC error asociado será la que menos contribuya en la predicción de la red.

2.2.2.2. Análisis de sensibilidad basado en la salida.

Otra forma de realizar el análisis de sensibilidad consiste en estudiar el efecto que se observa directamente en una variable de salida debido al cambio que se produce en una variable de entrada. De esta forma, sobre la red entrenada se fija el valor de todas las variables de entrada a su valor medio y, a continuación, se puede optar por añadir ruido o ir variando el valor de una de las variables a lo largo de todo su rango mediante la aplicación de pequeños incrementos Esto permite registrar los cambios producidos en la salida de la red y aplicar sobre estos cambios un índice resumen que dé cuenta de la magnitud del efecto de las variaciones producidas en la entrada x_i sobre la salida y_k. Esta forma de proceder es sencilla de aplicar. Sin embargo, desde esta aproximación se debe tomar una decisión bastante arbitraria acerca de la cantidad de ruido a añadir o de la magnitud del incremento y también del valor al que quedan fijadas las demás variables. A continuación se presentan dos métodos basados en el análisis de sensibilidad sobre la salida que gozan de un fundamento matemático más sólido: la matriz de sensibilidad Jacobiana y el método de sensibilidad numérico.

2.2.2.3. Matriz de sensibilidad Jacobiana.

El algoritmo de aprendizaje backpropagation aplicada a un perceptrón multicapa se basa en el cálculo de la derivada parcial del error con respecto a los pesos para averiguar qué dirección tomar para modificar los pesos con el fin de reducir el error de forma iterativa. En este sentido, los elementos que componen la matriz Jacobiana S proporcionan, de forma analítica, una medida de la sensibilidad de las salidas a cambios que se producen en cada una de las variables de entrada. En la matriz Jacobiana S, cada fila representa una entrada de la red y cada columna representa una salida de la red, de forma que el elemento S_{ik} de la matriz representa la sensibilidad de la salida k con respecto a la entrada i. Cada uno de los elementos S_{ik} se obtiene calculando la derivada parcial de una salida y_k con respecto a una entrada x_i, esto es, $\frac{\partial y_k}{\partial x_i}$. En este caso, la derivada parcial representa la pendiente instantánea de la función subyacente entre x_i e y_k para unos

valores dados de ambas variables. Aplicando la regla de la cadena sobre $\frac{\partial y_k}{\partial x_i}$ tenemos que:

$$S_{ik} = \frac{\partial y_k}{\partial x_i} = f'(net_k) \cdot \sum_{j=1}^{L} v_{jk} \cdot f'(net_j) \cdot w_{ij}$$

Así, cuanto mayor sea el valor absoluto de S_{ik}, más importante es x_i en relación a y_k. El signo de S_{ik} indica si el cambio observado en y_k va en la misma dirección o no que el cambio provocado en x_i. Cuando la discrepancia entre la salida y_k calculada por la red y la salida deseada d_k para un patrón dado es mínima, el término derivativo $f'(net_k)$ es próximo a cero con funciones sigmoidales y, como consecuencia, el valor de la derivada parcial queda anulado. Con el fin de solucionar este problema, se ha propuesto suprimir el término derivativo $f'(net_k)$ asumiendo que no afecta a la comparación entre las sensibilidades de las diferentes entradas respecto a la salida.

Como se puede observar, los valores de la matriz Jacobiana dependen no solo de la información aprendida por la red neuronal, que está almacenada de forma estática en las conexiones w_{ij} y v_{ij}, sino también de la activación de las neuronas de la capa oculta y de salida que, a su vez, dependen de las entradas de la red. Como diferentes patrones de entrada pueden proporcionar diferentes valores de pendiente, la sensibilidad necesita ser evaluada a partir de todo el conjunto de entrenamiento. Considerando el valor de sensibilidad entre i y k para todo el patrón X_p como $S_{ik}(p)$, podemos definir la sensibilidad a partir de la esperanza matemática, $E(S_{ik}(p))$, y la desviación estándar, $SD(S_{ik}(p))$, que pueden ser calculadas mediante:

$$E(S_{ik}(p)) = \frac{\sum_{p=1}^{P} S_{ik}(p)}{P}$$

$$SD(S_{ik}(p)) = \sqrt{\frac{\sum_{p=1}^{P}(S_{ik}(p) - E(S_{ik}(p)))\text{^}2}{P-1}}$$

En (Montaño Moreno, 2002) [5] se recogen una serie de referencias bibliográficas donde se hacen pruebas de validación de esta metodología.

2.2.2.4. Método de sensibilidad numérico.

Los métodos expuestos hasta ahora cuentan con una serie de limitaciones que comentamos brevemente. El análisis basado en la magnitud de los pesos no ha demostrado ser sensible a la hora de ordenar las variables de entrada en función de su importancia sobre la salida y, en los estudios comparativos, el análisis de sensibilidad

basado en el cálculo de la matriz Jacobiana ha demostrado ser siempre superior. El análisis consistente en añadir incrementos o perturbaciones se basa en la utilización de variables de entrada cuya naturaleza es continua, ya que no sería del todo correcto añadir incrementos a variables nominales, esto es, variables que toman valores discretos. Por su parte, el cálculo de la matriz Jacobiana parte del supuesto de que todas las variables implicadas en el modelo son continuas. Este supuesto limita el número de campos de aplicación de las RNA si se desea aplicar este método de sensibilidad.

En (Montaño Moreno, 2002) [5] se presenta un método que denominan análisis de sensibilidad numérico (NSA, *numeric sensitivity análisis*), trata de superar las mencionadas limitaciones. El método se basa en las pendientes que se forman entre entradas y salidas, respetando la estructura inicial de los datos.

Para analizar el efecto de una variable de entrada x_i sobre una variable de salida y_k mediante el método NSA, en primer lugar, debemos ordenar ascendentemente los p patrones a partir de los valores de la variable de entrada x_i. En función de tal ordenación, se genera un número G determinado de grupos de igual o aproximado tamaño. El número idóneo de grupos dependerá del número de patrones disponible y de la complejidad de la función que se establece entre la entrada y la salida, aunque en la mayoría de los casos será suficiente un valor de G=30 o similar. Para cada grupo formado se calcula la media aritmética de la variable x_i y la media aritmética de la variable y_k. A continuación, se obtiene el índice NSA basado en el cálculo numérico de la pendiente formada entre cada par de grupos consecutivos g_r y g_{r+1} de x_i sobre y_k mediante la siguiente expresión:

$$NSA_{ik}(g_r) = \frac{\overline{y_k}(g_{r+1}) - \overline{y_k}(g_r)}{\overline{x_i}(g_{r+1}) - \overline{x_i}(g_r)}$$

Donde

$\overline{x_i}(g_{r+1})$ y $\overline{x_i}(g_r)$ son las medias de la variable x_i correspondientes a los grupos g_r y g_{r+1}, respectivamente.

$\overline{y_k}(g_{r+1})$ y $\overline{y_k}(g_r)$ son las medias de la variable y_k correspondientes a los grupos g_r y g_{r+1}, respectivamente.

Una vez calculados los G-1 valores de NSA, se puede obtener el valor de la esperanza matemática del índice NSA o pendiente entre la variable de entrada i y la variable de salida k mediante:

$$E\big(NSA_{ik}(g_r)\big) = \sum_{r-1}^{G-1} NSA_{ik}(g_r) \cdot f\big(NSA_{ik}(g_r)\big) = \frac{\overline{y_k}(g_G) - \overline{y_k}(g_1)}{\overline{x_i}(g_G) - \overline{x_i}(g_1)}$$

Donde:

$f\big(NSA_{ik}(g_r)\big) = \frac{\overline{x_i}(g_{r+1}) - \overline{x_i}(g_r)}{\overline{x_i}(g_G) - \overline{x_i}(g_1)}$

$\overline{x_i}(g_G)$ y $\overline{x_i}(g_1)$ son los valores promedio de la variable x_i para el último grupo de g_G y el primer grupo g_1, respectivamente.

$\overline{y_k}(g_G)$ y $\overline{y_k}(g_1)$ son los valores promedio de la variable $\overline{y_k}$ para el grupo g_G y el grupo g_1, respectivamente.

Cuando la variable de entrada x_i es binaria, la esperanza del índice NSA se obtiene calculando la media de la variable y_k cuando la variable x_i toma el valor mínimo y la media de la variable y_k cuando la variable x_i toma el valor máximo, y aplicando la siguiente expresión:

$$E\left(NSA_{ik}(g_r)\right) = \frac{\overline{y_k}(x_{imax}) - \overline{y_k}(x_{imin})}{x_{imax} - x_{imin}}$$

Donde:

x_{imax} y x_{imin} son los valores máximo y mínimo que toma la variable x_i, respectivamente.

$\overline{y_k}(x_{imax})$ e $\overline{y_k}(x_{imin})$ son la media de la variable y_k cuando la variable x_i toma el valor máximo y la media de la variable y_k cuando la variable x_i toma el valor mínimo, respectivamente.

El valor de la esperanza matemática del índice NSA representa el efecto promedio que tiene un incremento de x_i sobre y_i. Cuando la variable de entrada es binaria, la esperanza matemática representa el efecto promedio provocado por el cambio del valor mínimo al calor máximo en la variable x_i. Al igual que en el caso de la matriz Jacobiana, cuanto mayor sea el valor absoluto de $E(NSA_{ik}(g_r))$, más importante es x_i en relación a y_k. El signo de $E(NSA_{ik}(g_r))$ indica si el cambio observado en y_k va en la misma dirección o no que el cabio provocado en x_i.

En (Montaño Moreno, 2002) [5] además, se destaca que han realizado diversos estudios piloto y que la forma más adecuada de representar las variable de entrada de naturaleza discreta (binarias o politómicas) es mediante la utilización de codificación ficticia, tal y como se hace habitualmente en el modelado estadístico. La introducción de los valores originales de una variable politómica en la red neuronal no permite reflejar de forma adecuada el efecto o pendiente que tiene el cambio de una categoría a otra categoría. Por otro lado, se afirma que es conveniente que las variables de entrada y salida sean reescaladas al mismo rango de posibles valores (por ejemplo, entre 0 y 1). Esto último evita posibles sesgos en la obtención del índice NSA debido a la utilización de escalas de medida diferentes entre las variables de entrada y , además, permite obtener un valor de esperanza matemática estandarizado, a diferencia de la matriz Jacobiana. Así el rango de valores que puede adoptar $E(NSA_{ik}(g_r))$ oscila entre -1 y +1. Estos dos límites indicarían un efecto máximo de la variable de entrada sobre la salida, con una relación negativa en el primer caso (-1) y una relación positiva en el segundo (+1). Los valores iguales o próximos a cero indicarían ausencia de efecto de la variable de entrada.

El cálculo de la desviación estándar del índice NSA, cuando las variables implicadas son de naturaleza continua se puede realizar mediante:

$$SD\left(NSA_{ik}(g_r)\right) = \sqrt{E\left(NSA_{ik}^2(g_r)\right) - \left(E\left(NSA_{ik}(g_r)\right)\right)^2}$$

El valor de la desviación estándar se debe interpretar como el grado de oscilaciones que ha sufrido la pendiente que se establece entre x_i e y_k, de manera que a mayor valor de la desviación estándar, mayor comportamiento caótico o aleatorio tiene la función entre las dos variables implicadas.

2.3. Ejemplos.

En el correspondiente capítulo del Anexo de este trabajo se recogen una serie de ejemplos interesantes tanto de aplicación de redes neuronales para ajuste de funciones como para clasificación.

3. Capítulo III: Base de datos.

Los datos que se emplean a lo largo de este proyecto (salvo que se indique lo contrario) han sido facilitados por la Oficina de Prospección y Análisis de Datos de la Universidad Politécnica de Cartagena.

El perfil de los alumnos considerados en todo este estudio es: por un lado, alumnos titulados (con 300 ECTS) como Ingenieros superiores de Telecomunicaciones o Ingenieros superiores Industriales, ingresados por PAU en el DURM (Distrito Único de la Región de Murcia), excluyendo del conjunto a aquellos que accedieron a la titulación superior por la posesión de una titulación técnica; por otro lado, se han considerado alumnos de las dos titulaciones mencionadas que abandonan la carrera tras su primer año de matrícula o que no están matriculados en su año teórico de egreso o al siguiente.

Es preciso destacar ahora que en este texto principal se recogen los resultados obtenidos para los alumnos de Ingeniería Industrial. El mismo estudio se ha realizado también para los alumnos de Ingeniería de Telecomunicaciones y los resultados se recogen en un anexo del presente trabajo.

En el capítulo cuatro del presente trabajo trataremos de predecir la nota media de egreso de la carrera para alumnos con el perfil anteriormente descrito a partir de dos bloques de variables: variables relacionadas con la prueba de acceso a la universidad (PAU) y variables relacionadas con las asignaturas más complejas de la carrera.

En el capítulo cinco trataremos de predecir si un alumno abandonará o no a partir de tres bloques de variables: variables PAU, variables explicativas del rendimiento del alumno durante su primer año matriculado y variables explicativas del rendimiento del alumno durante su segundo año matriculado. Se tratará de predecir el abandono desde dos perspectivas: el abandono entendido como el alumno que tras un año matriculado, no vuelve a matricularse y el abandono oficial, entendido como el alumno que no está matriculado en su año teórico de egreso ni al siguiente. De lo que se trata es de predecir el abandono de la forma más prematura posible. Por tanto, veremos los resultados que se obtienen introduciendo en el conjunto de Inputs las variables de manera cronológica: primero veremos los resultados que se obtienen teniendo en el conjunto de Inputs únicamente las variables PAU, después veremos los resultados teniendo en el conjunto de Inputs, variables PAU y de primer curso y, finalmente, veremos los resultados que se obtienen usando como conjunto de Inputs, variables PAU, más variables de primer curso, más variables de segundo curso.

Veamos las variables implicadas con más detenimiento.

3.1. Variables para la predicción/clasificación de la nota media de egreso.

3.1.1. Variables PAU.

Las variables PAU que se consideran son las siguientes:

Variable	Codificación	Escala
Nota PAU en la parte 1	P1	0-10
Ranking parte 1/Total presentados	R_P1	0-1
Nota PAU en la parte 2	P2	0-10
Ranking parte 2/Total presentados	R_P2	0-1
Calificación PAU	C	0-10
Ranking calificación PAU/Total presentados	R_C	0-1
Calificación física PAU	F	0-10
Ranking calificación física/Total presentados	R_F	0-1
Calificación Matemáticas PAU	M	0-10
Ranking calificación matemáticas/Total presentados	R_M	0-1

Como se puede apreciar tenemos, por un lado, para cada alumno notas de distintos apartados de la PAU y, por otro lado, tenemos el ranking que ocupa el alumno en ese apartado, es decir, su posición relativa respecto al resto de alumnos. El apartado de ranking ha sido calculado dividiendo la posición del alumno (en una lista ordenada decrecientemente: el primero es el que más nota tiene y el último el que menos) entre el número total de alumnos presentados a ese apartado ese año.

A partir de estos datos se lanzan tres metodologías de actuación para poder ver cuál de ellas proporciona mejores resultados: RNA para clasificación, RNA para ajuste de funciones y regresión lineal múltiple.

3.1.2. Datos de las asignaturas más complejas de cada titulación.

Para cada una de las asignaturas introduciremos tres variables: nota obtenida por el alumno (N), número de convocatorias presentadas (CP), número de convocatorias transcurridas hasta que el alumno se presenta por primera vez (CT).

3.1.2.1. Ingeniería de Telecomunicaciones.

Las asignaturas consideradas son:

Asignatura	Curso
Electrónica analógica	1º
Campos electromagnéticos	2º
Diseño de circuitos y sistemas electrónicos	4º
Redes de ordenadores	4º
Tratamiento digital de señales	5º

3.1.2.2. Ingeniería Industrial.

Se considerarán las mismas variables que para Telecomunicaciones, en este caso de las siguientes asignaturas:

Asignatura	Curso
Mecánica	2º
Mecánica de fluidos general	2º
Mecánica de fluidos aplicada	3º
Teoría de sistemas	3º
Tecnología de fabricación y tecnología de máquinas	4º
Teoría de estructuras y construcciones industriales	4º
Ingeniería térmica y de fluidos	4º

Las metodologías y técnicas que se emplearán para analizar estos datos son las ya señaladas anteriormente: RNA para clasificación, RNA para ajuste de funciones y regresión lineal múltiple.

3.2. Variables para la clasificación de abandono.

3.2.1. Variables PAU.

Se verán involucradas las mismas variables descritas en el apartado 3.1.1.

3.2.2. Variables del primer año de matrícula.

Las variables de primer año que se han empleado para la predicción de abandono son las siguientes:

En primer lugar, veamos las variables que se han considerado como influyentes en el abandono de un alumno:

- Créditos matriculados: Es el número de créditos de los que el alumno se matricula inicialmente en su primer año de universidad.
- Tasa evaluados: Esta variable representa el tanto por uno de créditos evaluados respecto a los créditos matriculados. Se considera que los créditos de una asignatura han sido evaluados cuando el alumno se ha presentado por lo menos una vez a la correspondiente evaluación. Lógicamente, este indicador ha de ser menor que 1.
- Tasa presentados: Representa la proporción de créditos presentados respecto a los créditos matriculados. Si el alumno se presenta dos veces (con esta primera matrícula) a una asignatura, los créditos contarán por partida doble. Este indicador puede ser mayor que uno.
- Tasa rendimiento: Proporción de créditos aprobados respecto a los matriculados.
- Tasa éxito: Proporción de créditos aprobados respecto a los evaluados.
- Número de convocatorias: Número de convocatorias a las que se podría haber presentado el alumno ese año. Representa el número de convocatorias por asignatura en las que el alumno aparece en actas. Por ejemplo, si se trata de una asignatura del primer cuatrimestre y el alumno aprueba en Febrero, esa asignatura sumará 1 a esta variable. En cambio, si la aprueba en Junio, sumará 2 y si la aprueba en Septiembre, 3.
- Tasa NP: proporción de convocatorias no presentadas respecto al número de convocatorias total.
- Tasa 0: proporción de convocatorias suspensas respecto al número de convocatorias total.
- Tasa 1: proporción de convocatorias con calificación Aprobado respecto al número de convocatorias total.
- Tasa 2: proporción de convocatorias con calificación Notable respecto al número de convocatorias total.
- Tasa 3: proporción de convocatorias con calificación Sobresaliente respecto al número de convocatorias total.
- Tasa 4: proporción de convocatorias con calificación Matrícula de Honor respecto al número de convocatorias total.

3.2.3. Variables del segundo año de matrícula.

Se han empleado las mismas variables descritas en el apartado 3.2.2., pero con los valores correspondientes al segundo año de matrícula del alumno.

4. Capítulo IV: Estudio de alumnos egresados.

4.1. Clasificación.

4.1.1. Estudio para alumnos egresados de Ingeniería Industrial.

El objetivo de esta sección es el estudio de la predictibilidad del rendimiento académico de alumnos de Ingeniería Industrial mediante Redes Neuronales Artificiales para clasificación. Se empleará Matlab para la resolución de los problemas, siguiendo la metodología, algoritmo y código descritos en el capítulo II.

4.1.1.1. Estudio a partir de datos PAU.

a) Estudio previo de los datos.

La Red Neuronal clasificará, en función de las variables de entrada, a los alumnos en uno de los siguientes grupos, determinando la pertenencia a estos grupos que su nota media en la titulación se halla dentro de los extremos establecidos:

Extremo inferior	Extremo superior	Caracterización
1.0	1.5	Aprobado bajo
1.5	2.0	Aprobado alto
2.0	2.5	Notable bajo
2.5	3.0	Notable alto
3.0	4.0	Sobresaliente

La condición de igualdad se halla en el extremo inferior para todos los grupos excepto para el último, en el que dicha condición se halla en ambos extremos.

En primer lugar, veamos algunos parámetros característicos de las variables de entrada:

Variable	Media	Desviación típica
Nota PAU en la parte 1	7.1296	1.1466
Ranking parte 1/Total presentados	0.2417	0.2226
Nota PAU en la parte 2	7.5679	1.3167
Ranking parte 2/Total presentados	0.1595	0.1817
Calificación PAU	8.0302	0.9851
Ranking calificación PAU/Total presentados	0.1740	0.1852
Calificación física PAU	7.2830	1.7824
Ranking calificación física/Total presentados	0.2464	0.2054
Calificación Matemáticas PAU	7.8057	1.8784
Ranking calificación matemáticas/Total presentados	0.2490	0.2311

Vemos que los alumnos DURM que ingresan en Ingeniería Industrial presentan características similares a los alumnos de Telecomunicaciones [Ver anexo] con una media en la PAU bastante alta (8 puntos sobre 10). Además su posición respecto al resto de alumnos DURM es bastante buena pues la media de los apartados de ranking es siempre baja: de cada cien alumnos DURM, la posición media de un alumno de nuevo ingreso en ingeniería de telecomunicaciones para las variables consideradas estaría en torno a la 25.

Indicador de la variable de salida:

	Valor medio	Desviación típica
Nota egresado.	1.636	0.3059

Vemos que el valor medio para la calificación media de los egresados de la titulación de Ingeniería Industrial es de aprobado alto, con una desviación típica pequeña (0.3059).

b) Entrenamiento:

El proceso de entrenamiento se llevará a cabo con un conjunto de datos correspondiente a 159 alumnos de la Escuela Técnica Superior de Ingeniería Industrial, en las condiciones establecidas en el capítulo III.

La matriz de entrada (Inputs) presentará un aspecto similar al representado en la siguiente tabla:

	Alumno 1	Alumno 2	Alumno […]	Alumno 159
P1	4,0000	6,2000	[…]	6,7000
R_P1	0,9016	0,4170	[…]	0,2733
P2	7,6000	3,6000	[…]	7,4000
R_P2	0,0790	0,8245	[…]	0,0950
C	7,0000	6,1000	[…]	8,4000
R_C	0,3370	0,6658	[…]	0,0585
F	8,8000	5,0000	[…]	10,0000
R_F	0,1159	0,5699	[…]	0,0064
M	5,8000	2,8000	[…]	4,4000
R_M	0,1855	0,9089	[…]	0,3710

Mientras que la matriz de Targets, presentará el siguiente aspecto:

	Alumno 1	Alumno 2	Alumno […]	Alumno 170
Aprobado bajo	1	1	[…]	0
Aprobado alto	0	0	[…]	1
Notable bajo	0	0	[…]	0
Notable alto	0	0	[…]	0
Sobresaliente	0	0	[…]	0

Donde 1 determina la pertenencia al grupo en cuestión y 0 representa la no pertenencia.

Utilizaremos para el entrenamiento el algoritmo del Gradiente Escalado Conjugado, descrito en el capítulo II. El conjunto de entrenamiento estará formado por el 70% de los datos disponibles, el conjunto de validación estará compuesto por el 15% de los datos, y el conjunto de test estará compuesto por el restante 15%.

Se llevarán a cabo cuatro entrenamientos, teniendo lugar el primero y el segundo a igualdad de condiciones entre sí, y el tercero y el cuarto también a igualdad de condiciones entre sí, tal y como recogen las tablas siguientes. El objetivo de realizar este emparejamiento es ver la variabilidad de los resultados de la red por la aleatoriedad en la inicialización de los pesos y la aleatoriedad en la formación de los grupos de entrenamiento, validación y test.

b.1) Entrenamientos 1 y 2:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	159
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

b.2) Entrenamiento 3 y 4:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	159
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/50

c) Comparación de resultados:

Entrenamiento	1	2	3	4
Épocas	27	22	15	36
Tiempo (s)	1	2	1	0
Rendimiento	0.102	0.0896	0.0906	0.0816
Gradiente	0.0225	0.0255	0.0375	0.0240
Comprobaciones de validación	6	6	6	6
% aciertos (1.0-1.5)	33.3	75.0	50.0	0.0
% aciertos (1.5-2.0)	84.6	44.4	75.0	61.1
% aciertos (2.0-2.5)	0.0	0.0	0.0	50.0
% aciertos (2.5-3.0)	0.0	NaN	0.0	NaN
% aciertos (3.0-4.0)	NaN	NaN	NaN	NaN
% aciertos General	58.3	45.8	58.3	50.0

d) Conclusiones:

Vemos por un lado que la inicialización aleatoria de los pesos y la selección aleatoria de los conjuntos de entrenamiento, validación y test genera variaciones que ronda en torno al 15%. Además, al aumentar el número de capas ocultas no se producen mejoras significativas.

Los resultados obtenidos presentan las mismas características que para el caso de los alumnos de Telecomunicaciones [Ver anexo]. La nota media ronda el 1.6 para ambas titulaciones, teniendo la mayoría de los ejemplos disponibles una nota media de egreso inferior a 2. Ello incapacita a la red para predecir el rendimiento de alumnos que obtengan notable o sobresaliente pues no existen suficientes ejemplos de entrenamiento de estos grupos, que permitan que los pesos alcancen un valor general óptimo. Este aspecto, unido a la gran variabilidad que introduce la inicialización aleatoria de los pesos y a la selección aleatoria de los ejemplos, hace que los resultados obtenidos no sean satisfactorios.

Por tanto, concluimos que para los alumnos de Ingeniería Industrial e Ingeniería de Telecomunicaciones no es posible predecir mediante Redes Neuronales Artificiales de clasificación la nota media de la carrera a partir de las variables de acceso a la universidad consideradas.

4.1.1.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

En este caso introduciremos las asignaturas por cursos: primero las dos de segundo, después las dos de tercero y finalmente las tres asignaturas de cuarto.

a) Estudio previo de los datos.

Asignatura	Variable	Media	Desviación típica
Mecánica	N	5.8196	1.0503
	CP	1.9860	1.5428
	CT	2.8182	2.4137
Mecánica de fluidos general	N	6.0273	0.9944
	CP	1.5944	1.0430
	CT	2.1259	1.8227
Mecánica de fluidos aplicada	N	5.9490	1.1106
	CP	1.5245	0.9629
	CT	2.2238	2.5019
Teoría de sistemas	N	5.6343	0.9752
	CP	2.3147	1.5310
	CT	1.7972	1.7262
Tecnología de fabricación y tecnología de máquinas	N	5.9385	1.0332
	CP	1.8112	0.9926
	CT	2.7483	2.2719
Teoría de estructuras y construcciones industriales	N	6.1063	1.1037
	CP	1.5455	0.8618
	CT	1.8951	1.7551
Ingeniería térmica y de fluidos	N	6.1308	0.9932
	CP	1.6154	0.8301
	CT	2.6084	2.1690

Vemos que las notas medias obtenidas en estas asignaturas se hallan en torno a los 6.0 puntos, ligeramente inferiores a las obtenidas por los ingenieros de telecomunicaciones. En cuanto a convocatorias presentadas y convocatorias transcurridas hasta presentarse por primera vez, destaca por un lado Teoría de sistemas, ya que los alumnos requieren de media más de dos convocatorias para aprobar; por otro lado, en cuanto a convocatorias que se dejan pasar antes de presentarse por primera vez, destacan sobre todo Mecánica, Tecnología de fabricación y tecnología de máquinas e Ingeniería Térmica y de Fluidos, con cerca de tres.

b) Entrenamiento:

Se llevarán a cabo siete entrenamientos introduciendo cronológicamente las asignaturas, en cada uno de los cuales se considerarán como variables de entrada todas las variables PAU, más las tres variables (N, CP, CT) de la nueva asignatura que se introduce en el conjunto, más las tres de las que se han introducido previamente. Por tanto, se usarán 13, 16, 19, 22, 25, 28, 31 variables respectivamente. Se usarán los

mismos alumnos que para el entrenamiento sólo con variables PAU, habiéndose eliminado 16 que no tenían todos los campos completos (143 ejemplos: 70% para entrenamiento, 15% para validación y 15% para test). Se empleará el gradiente escaldo conjugado y como arquitectura se empleará una red con una única capa oculta de 25 neuronas.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	143
Variables consideradas	13 (16, 19, 22, 25, 28, 31)
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

c) Comparación de resultados.

Entrenamiento	1	2	3	4	5	6	7
Épocas	21	23	33	24	7	23	21
Tiempo (s)	0	2	0	0	0	0	0
Rendimiento	0.0832	0.0788	0.0611	0.0638	0.143	0.0384	0.0502
Gradiente	0.0332	0.0329	0.0269	0.0298	0.0305	0.0764	0.0422
Comprobaciones de validación	6	6	6	6	6	6	6
% aciertos (1.0-1.5)	57.1	40.0	40.0	40.0	0.0	77.8	80.0
% aciertos (1.5-2.0)	60.0	83.3	87.5	80.0	100.0	88.9	60.0
% aciertos (2.0-2.5)	0.0	0.0	0.0	0.0	NaN	100.0	0.0
% aciertos (2.5-3.0)	0.0	0.0	100	0.0	NaN	100.0	NaN
% aciertos (3.0-4.0)	NaN	NaN	NaN	NaN	NaN	NaN	NaN
% aciertos General	47.6	42.9	57.1	47.6	61.9	85.7	61.9

d) Conclusiones.

Vemos que los resultados tienen similares características a los que obtuvimos para los ingenieros de Telecomunicaciones [Ver Anexo]. Son malos en el sentido de que no proporcionan tasas de acierto razonablemente buenas.

En este conjunto de entrenamientos, parece que al introducir en el conjunto de inputs, la asignatura de Teoría de Estructuras y Construcciones industriales los resultados mejoran. Sin embargo, al repetir el entrenamiento (lo que supone una nueva inicialización aleatoria de los pesos y una nueva selección aleatoria de los ejemplos) en las condiciones de la columna 6, hemos obtenido resultados similares al resto, con tasas de acierto de en torno al 60%. Ello significa que en el primer entrenamiento en la condiciones de la columna 6 se han alcanzado unos valores para los pesos que permiten clasificar ejemplos de manera bastante general por lo que, en principio, parece que la

asginatura de “Teoría de estructuras y construcciones industriales” representa bastante bien los patrones de nota media de egreso de los alumnos.

4.1.2. Análisis y conclusiones de los resultados de clasificación.

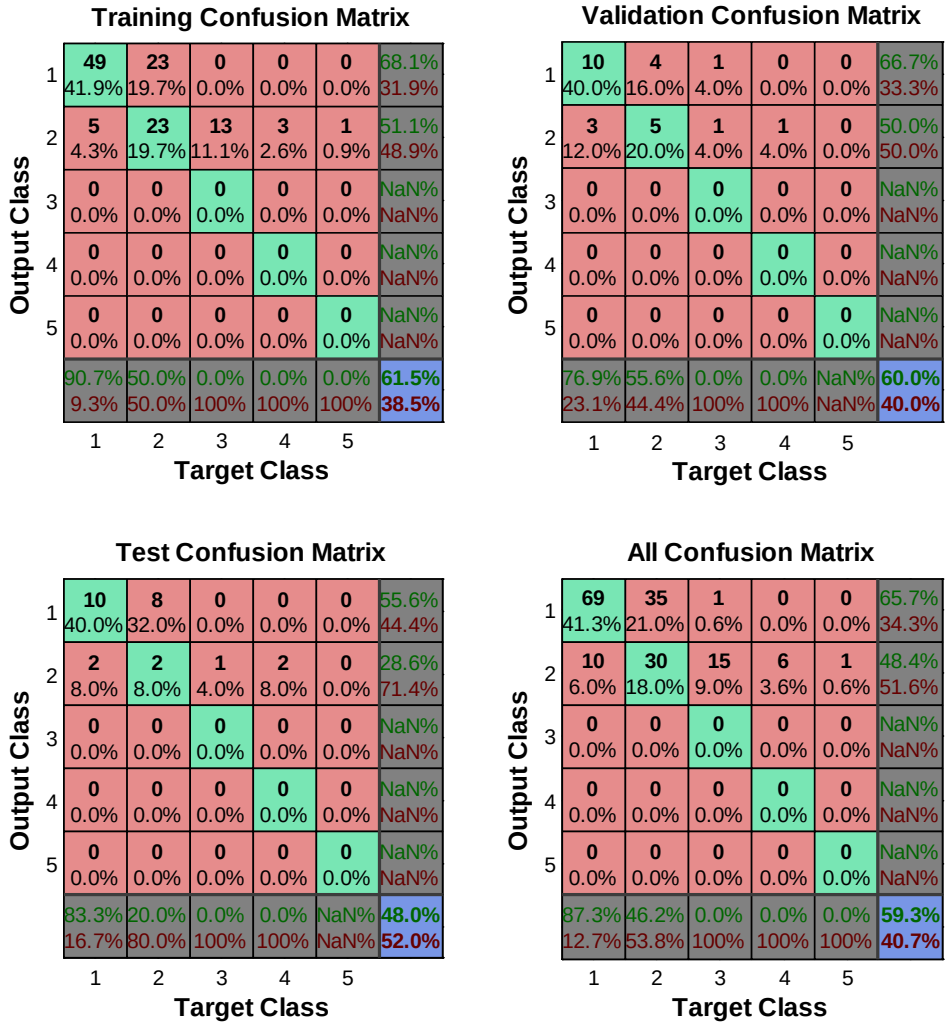
Los resultados obtenidos hasta ahora han sido malos ¿qué sucede? ¿Por qué no somos capaces de clasificar a los alumnos con ciertas garantías? Podemos plantear varias respuestas:

- Número insuficiente de ejemplos: Al enfrentarnos a un caso real la disponibilidad de datos que tenemos es limitada pues los alumnos egresados de la titulaciones empleadas son los que son y en las condiciones ya presentadas. Conjuntos más amplios de entrenamiento quizá permitirían a la red una búsqueda más exhaustiva de patrones; con conjuntos de validación más grandes, la generalización de la red sería más precisa; con conjuntos de test más grandes, los resultados presentarían menos variabilidad y seríamos capaces de captar el potencial real de la red entrenada.
- Incapacidad de las Redes Neuronales Artificiales para llevar a cabo la tarea que aquí se acomete. Este punto parece clave y en él haremos especial hincapié. Parece que intentar clasificar a los alumnos en clases en función de las notas no es la mejor manera de afrontar la problemática que aquí se define, pues estamos discretizando las notas (que son un continuo) y, además, de una manera grosera en el sentido de que englobamos en un mismo grupo o clase por ejemplo a dos alumnos que tienen 1.51 y 1.99 de nota media de egreso respectivamente, atribuyéndoles las mismas características, mientras que establecemos en clases separadas por ejemplo a dos alumnos que tienen 1.99 y 2.01 de nota media de egreso respectivamente, diciendo con ello a la Red Neuronal que poseen características totalmente distintas. Analicemos dos situaciones concretas:

a) Entrenamiento con datos PAU de ingenieros de Telecomunicaciones:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	170
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

Matriz de resultados:



Tenemos 25 ejemplos reservados para test

La salida de la red es la siguiente:

Alumno	1	2	3	4	5	6
Nota media	1.6157	1.6809	1.7281	1.4538	1.8009	1.5348
Clase real	2	2	2	1	2	2
Clase RNA	1	2	1	1	1	1
1-1.5 (1)	0.8477	0.2188	0.4487	0.8529	0.5316	0.5448
1.5-2 (2)	0.3207	0.4648	0.4064	0.4083	0.4606	0.4070
2-2.5 (3)	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
2.5-3 (4)	0.0050	0.0071	0.0038	0.0094	0.0038	0.0033
3-4 (5)	0.0019	0.0020	0.0024	0.0135	0.0031	0.0041
Fallo	x	-	x	-	x	x

Alumno	7	8	9	10	11	12
Nota media	1.5023	2.6380	1.5911	2.1861	1.4174	1.5200
Clase real	2	4	2	3	1	2
Clase RNA	1	2	1	2	1	1
1-1.5 (1)	0.9379	0.1423	0.8183	0.3565	0.9102	0.9830
1.5-2 (2)	0.2526	0.4315	0.4992	0.3923	0.2732	0.2310
2-2.5 (3)	0.0000	0.0000	0.0005	0.0001	0.0000	0.0001
2.5-3 (4)	0.0094	0.0081	0.0131	0.0061	0.0046	0.0034
3-4 (5)	0.0080	0.0023	0.0296	0.0107	0.0046	0.0080
Fallo	x	x	x	x	-	x

Alumno	13	14	15	16	17	18
Nota media	1.4518	1.3799	1.3559	1.8761	1.4240	1.3465
Clase real	1	1	1	2	1	1
Clase RNA	1	1	1	2	2	1
1-1.5 (1)	0.6027	0.5490	0.9731	0.1547	0.0975	0.5169
1.5-2 (2)	0.4654	0.4089	0.1998	0.5325	0.4434	0.2735
2-2.5 (3)	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000
2.5-3 (4)	0.0024	0.0032	0.0215	0.0045	0.0107	0.0107
3-4 (5)	0.0089	0.0087	0.0106	0.0032	0.0013	0.0006
Fallo	-	-	-	-	x	-

Alumno	19	20	21	22	23	24	25
Nota media	1.4128	1.3901	1.2534	2.6727	1.3891	1.3946	1.7281
Clase real	1	1	1	4	1	1	2
Clase RNA	1	1	2	2	1	1	1
1-1.5 (1)	0.8297	0.8089	0.2304	0.0319	0.8305	0.9263	0.8267
1.5-2 (2)	0.2390	0.2156	0.4266	0.5211	0.4358	0.2316	0.4475
2-2.5 (3)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001
2.5-3 (4)	0.0068	0.0111	0.0065	0.0137	0.0018	0.0397	0.0059
3-4 (5)	0.0022	0.0013	0.0013	0.0013	0.0053	0.0079	0.0114
Fallo	-	-	x	x	-	-	x

En la tabla se ve claramente que el entrenamiento no da resultados útiles:

Por un lado podríamos pensar que cuando la clase llevara asociada una salida bastante alta (próxima a 1), tendríamos más probabilidad de éxito: ello es lo que ocurre en alumnos como el 11, 15 y 24, con salidas asociadas de 0.9102, 0.9731 y 0.9263

respectivamente. En cambio encontramos alumnos como el 1, 7 y 12 con salidas asociadas de 0.8477, 0.9379 y 0.9830 respectivamente que, a pesar de ello, dan error.

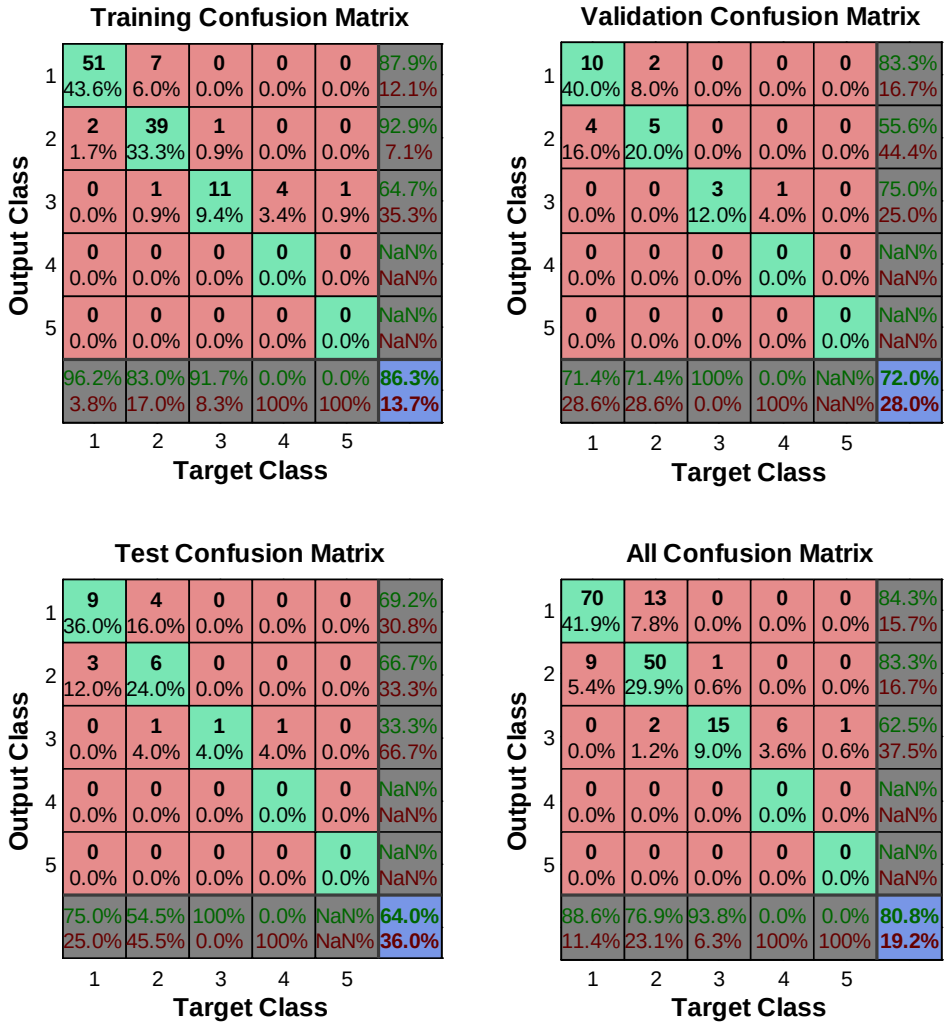
Otro punto de análisis sería ver si los errores se dan para alumnos cuya nota media se haya próxima a la zona de división entre dos clases. Para los alumnos 1, 7 y 12, con notas medias de 1.6157, 1.5023 y 1.5200 respectivamente ocurre así. En cambio para alumnos como el 3, 5 y 25, con notas medias de 1.7281, 1.8009 y 1.7281 respectivamente, situadas en la zona intermedia del intervalo, también se produce error.

Como conclusión podemos afirmar que las Redes Neuronales Artificiales no se pueden usar para clasificar a los alumnos en clases dependiendo de su nota media, a partir de variables PAU por las grandes tasas de error obtenidas y por las grandes dificultades que se presentan para sacar conclusiones fidedignas.

b) Introducimos como variables tanto las variables PAU como las relacionadas con las asignaturas más difíciles de la titulación.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	167
Variables consideradas	25
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

Matrices de resultados:



Alumno	1	2	3	4	5	6
Nota media	1.7436	1.4123	1.3777	1.4538	1.5219	1.6563
Clase real	2	1	1	1	2	2
Clase RNA	1	1	1	1	1	2
1-1.5 (1)	0.9407	0.9928	0.6552	0.9144	0.8215	0.0856
1.5-2 (2)	0.1405	0.0578	0.1746	0.2134	0.2490	0.9187
2-2.5 (3)	0.0005	0.0002	0.0036	0.0003	0.0007	0.0036
2.5-3 (4)	0.0047	0.0011	0.0126	0.0062	0.0084	0.0122
3-4 (5)	0.0088	0.0029	0.0019	0.0024	0.0025	0.0032
Fallo	x	-	-	-	x	-

Alumno	7	8	9	10	11	12
Nota media	1.8661	1.9130	1.4978	1.5628	1.3200	1.3167
Clase real	2	2	1	2	1	1
Clase RNA	2	3	2	1	1	1
1-1.5 (1)	0.0087	0.0021	0.0395	0.9973	0.9838	0.9628
1.5-2 (2)	0.8288	0.3705	0.9063	0.0049	0.1516	0.0736
2-2.5 (3)	0.0515	0.5480	0.0047	0.0004	0.0003	0.0002
2.5-3 (4)	0.0424	0.0806	0.0096	0.0067	0.0031	0.0045
3-4 (5)	0.0091	0.0087	0.0033	0.0162	0.0044	0.0022
Fallo	-	x	x	x	-	-

Alumno	13	14	15	16	17	18
Nota media	1.3799	1.1972	1.7444	1.3261	1.7345	1.4483
Clase real	1	1	2	1	2	1
Clase RNA	2	1	2	1	2	2
1-1.5 (1)	0.4169	1.0000	0.0167	0.9993	0.0599	0.0274
1.5-2 (2)	0.7341	0.0010	0.9324	0.0192	0.7881	0.9209
2-2.5 (3)	0.0018	0.0002	0.0139	0.0001	0.0206	0.0145
2.5-3 (4)	0.0187	0.0173	0.0165	0.0073	0.0093	0.0121
3-4 (5)	0.0056	0.0836	0.0016	0.0082	0.0044	0.0068
Fallo	x	-	-	-	-	x

Alumno	19	20	21	22	23	24	25
Nota media	1.7662	1.4128	1.5854	1.8929	1.3600	2.4565	2.5043
Clase real	2	1	2	2	1	3	4
Clase RNA	2	1	1	2	1	3	3
1-1.5 (1)	0.0994	0.9997	0.6048	0.0257	0.9985	0.0012	0.0015
1.5-2 (2)	0.7763	0.0007	0.4540	0.8482	0.0151	0.0446	0.0802
2-2.5 (3)	0.0118	0.0015	0.0012	0.0132	0.0001	0.9152	0.8327
2.5-3 (4)	0.0121	0.0177	0.0150	0.0255	0.0014	0.2851	0.2263
3-4 (5)	0.0031	0.0079	0.0019	0.0095	0.0100	0.0078	0.0060
Fallo	-	-	x	-	-	-	x

En este entrenamiento tenemos nueve alumnos mal clasificados: 1, 5, 8, 9, 10, 13, 18, 21, 25. De ellos, 5, 8, 9, 10, 18, 21, 25, con notas medias de 1.5219, 1.9130, 1.4978, 1.5628, 1.4483, 1.5854 y 2.5043 respectivamente se hallan cerca de una frontera de cambio de clase por lo que el error es, en principio, lógico. Sin embargo otros alumnos que también se hallaban en la frontera como el 20 o el 24 con notas medias de 1.4128 y 2.4565 respectivamente, han sido bien clasificados. La introducción de las asignaturas más difíciles de la titulación clarifica un poco la extracción de conclusiones y mejora

significativamente los resultados aunque estos todavía distan mucho de ser satisfactorios.

Respecto a los valores de salida de la red, situados en el intervalo [0,1], las conclusiones son similares a las obtenidas en el caso en el que sólo considerábamos las variables PAU.

Por tanto, vemos que definir el problema del estudio de la predictibilidad de la nota media de alumnos egresados como un problema de clasificación no es adecuado, por lo que abordaremos el problema desde otros puntos de vista.

4.2. Ajuste de funciones.

El estudio que se propone a continuación se desarrolla con los mismos datos que el del apartado anterior, aplicando la arquitectura, funciones y algoritmo de aprendizaje adecuados para optimizar el entrenamiento, de acuerdo con el capítulo II del presente trabajo. El objetivo es proponer una alternativa para la predicción de nota media de egreso para alumnos de Ingeniería de Telecomunicaciones e Industrial que dé mejores resultados que las técnicas empleadas anteriormente.

4.2.1. Estudio para alumnos egresados de Ingeniería Industrial.

4.2.1.1. Estudio a partir de datos PAU.

a) Entrenamiento.

Se desarrollan tres entrenamientos, en idénticas condiciones. Se ha decidido no introducir más de una capa porque no mejora los resultados [Ver resultados en Anexo para Ingeniería de Telecomunicaciones]. Se utilizará como algoritmo de aprendizaje el de Levenberg-Marquardt. Se dispone de 159 ejemplos (70% para test, 15% para validación y 15% para test%).

Función de entrenamiento.	Tranlm
Ejemplos de entrenamiento	159
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

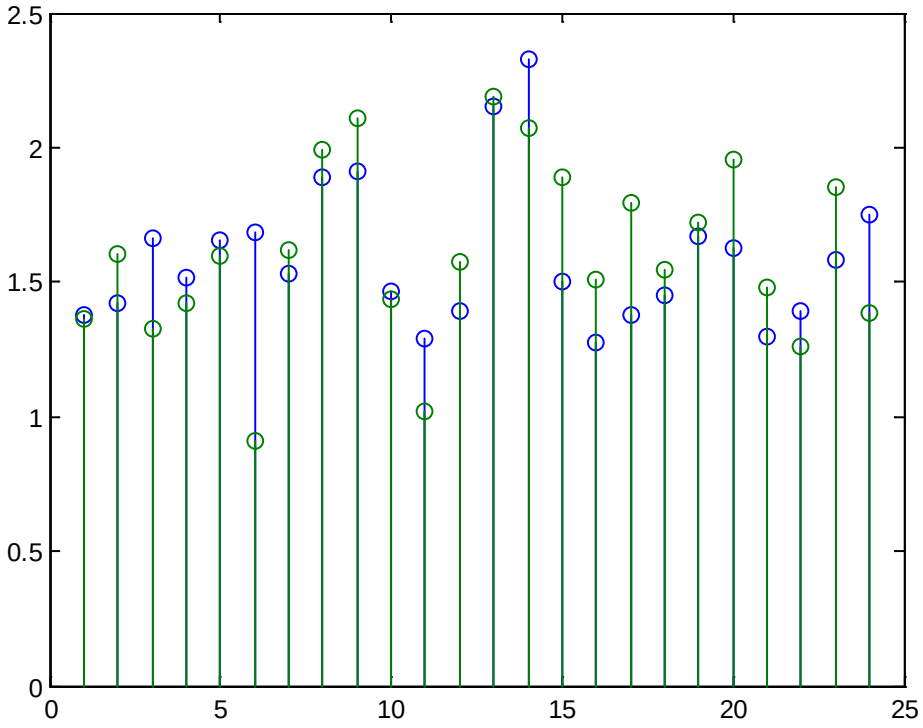
b) Resultados.

Entrenamiento	1	2	3
Épocas	11	9	11
Tiempo (s)	1	0	0
Rendimiento	0.00887	0.0222	0.0121
Gradiente	0.0372	0.0819	0.0608
Mu	0.0100	0.0100	0.0100
Comprobaciones de validación	6	6	6
|Error máximo|	1.026	0.7938	1.116
MSE	0.1352	0.0729	0.1022

c) Conclusiones.

Los resultados que se obtienen mediante ajuste de funciones son más adecuados que los que se obtenían mediante clasificación, más acordes a la naturaleza del problema abordado en el sentido de que permiten ofrecer salidas de la red continuas en un intervalo [0-4], lo que se asemeja más a los "Target" que tenemos. Es cierto que los "Target" son continuos en el intervalo [1-4] y que algunas salidas de la red son menores que 1. Con ello lo que la red nos indica es que de acuerdo con el entrenamiento recibido (y todo lo que ello implica: variables de entrada, arquitectura, función de entrenamiento,…) el alumno con menos de 1 de nota media no debería haber conseguido aprobar la carrera.

En la mejor iteración (la segunda), la comparación de los "Targets" con los "Outputs" de la red presenta el siguiente aspecto:



Vemos que hay casos en los que la predicción de la red es mala con errores por encima de los 0.25 puntos. Es lo que ocurre con los casos 6, 11, 14, 15, 16 ,17 20 y 23. En el resto de casos las predicciones son bastante buenas, destacando especialmente los ejemplos 1, 4, 7, 10, 13 y 19.

Los resultados obtenidos no son tan malos como los obtenidos en clasificación y en principio, parece que hemos sido capaces de crear a partir de los datos existentes, un programa que permita con cierto margen de error dar una aproximación más o menos veraz de la nota media de egreso del alumno a partir de sus características PAU. Es positivo ver que las salidas de la red tienen la tendencia, para muchos de los ejemplos de test, de los “targets” y que la situación relativa de dichas salidas es similar a la situación relativa de los “targets”.

Sin embargo, es preciso destacar que la mayor parte de los alumnos están comprendidos en una franja de notas realmente estrecha. Para el caso de Ingeniería de Telecomunicaciones [Ver Anexo] 111 de los 170 ejemplos empleados tienen notas medias entre 1.2 y 1.7 (más del 65%); 151 de los 170 alumnos tienen notas medias entre 1.2 y 2.2 puntos (el 88.82%), y 168 de los 170 alumnos tienen notas medias entre 1.1 y 2.8. Por ello, es lógico que los errores obtenidos sean bajos y errores por encima de los 0.25 puntos implican una predicción bastante mala porque el 65% de los alumnos tienen su nota media en un rango de cinco décimas (1.2-1.7).

4.2.1.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

a) Entrenamiento:

Se desarrollan siete entrenamientos y en cada uno de ellos se añaden las variables (N, CP, CT) de una asignatura (mecánica, mecánica de fluidos general, mecánica de fluidos aplicada, teoría de sistemas, tecnología de fabricación y tecnología de máquinas, teoría de estructuras y construcciones industriales e ingeniería térmica y de fluidos) al conjunto de entrada existente previamente, partiendo inicialmente del conjunto formado por las 10 variables PAU. Se utilizará como algoritmo de aprendizaje el de Levenberg-Marquardt. Se dispone de 143 ejemplos (70% para test, 15% para validación y 15% para test%).

Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	143
Variables consideradas	13 (16, 19, 22, 25, 28)
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

b) Comparación de resultados.

Entrenamiento	1	2	3	4	5	6	7
Épocas	14	12	8	9	7	10	9
Tiempo (s)	1	0	0	0	0	1	2
Rendimiento	0.00290	2.00e-31	5.06e-12	3.61e-29	3.14e-20	1.97e-31	9.47e-32
Gradiente	0.0197	6.32e-16	3.77e-06	3.28e-14	9.13e-10	5.34e-16	2.05e-16
Mu	0.0100	1.00e-09	1.00e-07	1.00e-10	1.00e-10	0.0100	1.00e+10
Comprobaciones de validación	6	6	6	6	6	6	6
|Error máximo|	0.474	0.7904	1.123	0.5123	0.5911	0.6968	0.7018
MSE	0.0765	0.1126	0.1354	0.0767	0.1164	0.0841	0.0716

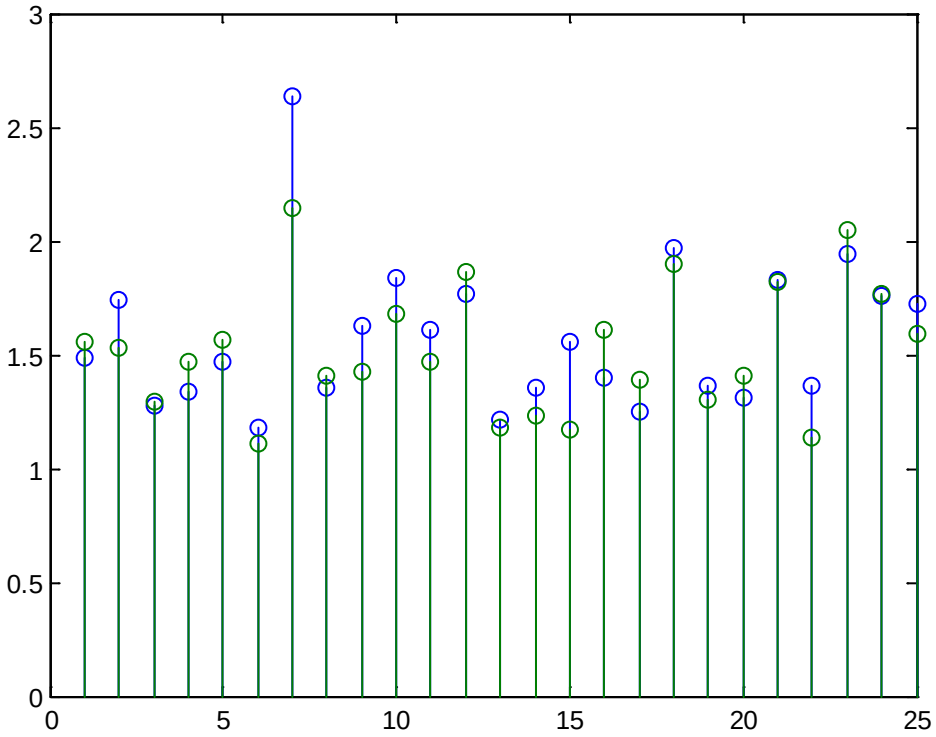
c) Conclusiones.

La introducción de asignaturas como variables que participan en la predicción supone una leve mejoría respecto a considerar únicamente variables PAU. Algo similar ocurre para Ingeniería de Telecomunicaciones.

4.2.2. Conclusiones para ajuste de funciones.

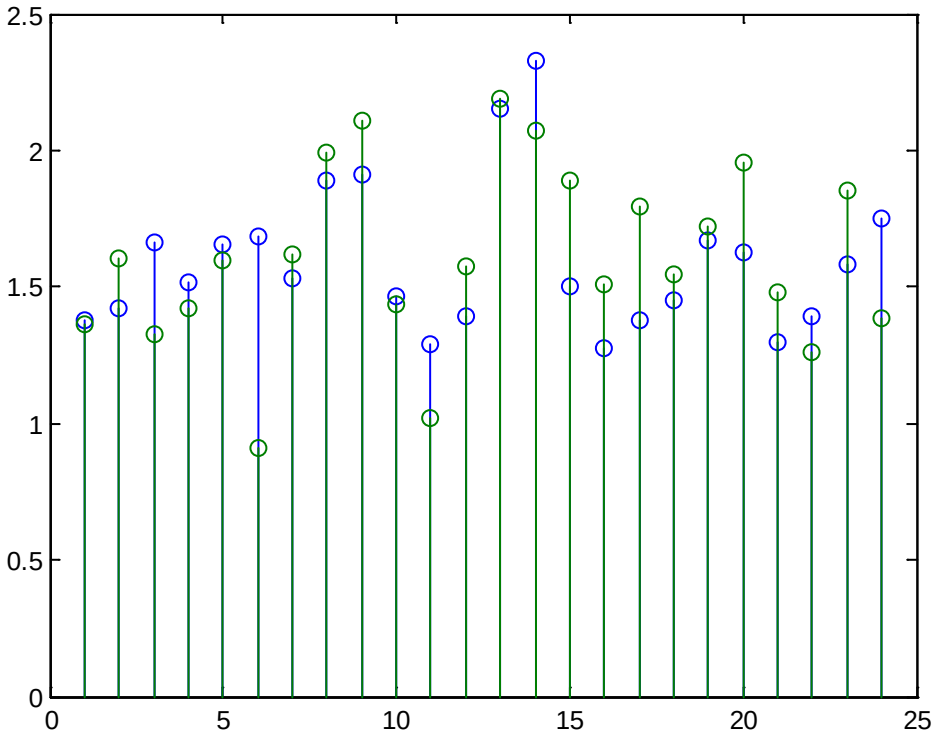
Como hemos podido ver, las RNA para ajuste de funciones son mucho más acordes a la naturaleza del problema abordado que las RNA para clasificación. A continuación se muestra una comparación entre los Targets y los Outputs de las mejores redes (considerando el MSE como indicador), tanto para telecomunicaciones como para industriales.

La red con menor MSE para telecomunicaciones es la que incluye todas las variables PAU más todas las asignaturas hasta Redes de ordenadores. Esta es la comparación:



Es interesante ver la gran capacidad predictiva de la red al añadir estas asignaturas: en 12 de los 25 casos de test, el error de ajuste es inferior a 0.1 puntos en la escala 1-4; en 18 de los 25 casos el error es inferior a 0.15 puntos; y en 23 de los 25 casos el error es inferior a 0.25 puntos.

La red con menor MSE para industriales es la que incluye todas las asignaturas difíciles más las variables PAU (0.0716). Sin embargo, en una de las iteraciones considerando sólo las variables PAU, se consiguió un MSE muy parecido (0.0729). Por tanto, mostraremos el mapa comparativo en el que sólo se consideraron variables PAU, porque las mejoras al introducir las asignaturas de la carrera no son significativas.



Vemos que en 8 de los 24 ejemplos empleados el error que comente la red es menor de 0.1 puntos; en 15 de los 25 ejemplos el error es inferior a 0.25 puntos y en 22 de los 25 ejemplos el error es inferior a 0.4 puntos, aunque un error de esta magnitud es bastante malo por la naturaleza del problema y el rango de valores tan pequeño en que se encuentran los targets.

4.3. Regresión lineal múltiple.

4.3.1. Estudio para alumnos egresados de Ingeniería Industrial.

4.3.1.1. Estudio a partir de datos PAU.

Se llevará a cabo una Regresión Lineal Múltiple con la que se pretende estimar la nota media de egreso de alumnos de Ingeniería Industrial en Ingeniería de Telecomunicaciones [Ver anexo] a partir de las 10 variables PAU descritas en el capítulo III:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 + \hat{\beta}_3 \cdot x_3 + \hat{\beta}_4 \cdot x_4 + \hat{\beta}_5 \cdot x_5 + \hat{\beta}_6 \cdot x_6 + \hat{\beta}_7 \cdot x_7 + \hat{\beta}_8 \cdot x_8 + \hat{\beta}_9 \cdot x_9 + \hat{\beta}_{10} \cdot x_{10}$$

Donde:

Variable	Significado
$\hat{y}$	Nota media de egreso estimada del alumno
x_1	Nota PAU en la parte 1
x_2	Ranking parte 1/Total presentados
x_3	Nota PAU en la parte 2
x_4	Ranking parte 2/Total presentados
x_5	Calificación PAU
x_6	Ranking calificación PAU/Total presentados
x_7	Calificación física PAU
x_8	Ranking calificación física/Total presentados
x_9	Calificación Matemáticas PAU
x_{10}	Ranking calificación matemáticas/Total presentados

Se dispone de datos acerca de 159 alumnos. Se usará el 85% de los datos para el cálculo de los estimadores y el 15% de los datos para test. Así estaremos en las mismas condiciones que en el apartado anterior, en el que usábamos Redes Neuronales Artificiales para ajustar una función que calculase la nota media de egreso. Es importante recordar que en RNA usábamos un 70% para entrenamiento, un 15% para validación y otro 15% para test. Al fin y al cabo, las matrices de pesos eran función de un 85% de los datos ya que era el conjunto de validación el que determinaba cuándo se detenía el entrenamiento. En cambio, en Regresión Lineal Múltiple no es necesario un conjunto de validación pues el cálculo es analítico. Tendremos, por tanto 135 ejemplos para el cálculo de los estimadores, y 24 ejemplos para testear el modelo propuesto.

Para el cálculo de los estimadores se ha empleado la fórmula:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Donde X es un vector que contiene la información de las variables PAU para los distintos ejemplos, de la forma:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$$

Siendo n el número de ejemplos empleados (144 en este caso) y k es el número de variables utilizadas (10 en este caso). La columna de unos al principio permite la aproximación del término independiente.

Y es el vector que contiene las notas medias de egreso para el conjunto de entrenamiento, tiene la siguiente forma:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

$\hat{\beta}$ es el vector columna que contiene los estimadores. Posee la siguiente forma:

$$\hat{\beta} = \begin{bmatrix} \widehat{\beta_0} \\ \hat{\beta}_1 \\ \dots \\ \hat{\beta}_k \end{bmatrix}$$

Por tanto, a partir de X e Y somos capaces de calcular un modelo mediante la obtención de los estimadores $\hat{\beta}$. Empleamos estos estimadores así como los correspondientes valores X_{test} de las variables PAU del conjunto de ejemplos reservados para testeo, para calcular las $\widehat{Y_{test}}$ (solución para la nota media de egreso que proporciona el modelo creado), y compararlas con las Y_{test}, que son las notas medias reales de egreso de los alumnos. En resumen:

Paso 1: Creamos el modelo: $\hat{Y} = \hat{\beta} \cdot X$ mediante la aplicación de la expresión:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Siendo X e Y las matrices correspondientes al conjunto de entrenamiento.

Paso 2: Calculamos las estimaciones que hace el modelo para las Y del conjunto de test:

$$\widehat{Y_{test}} = \hat{\beta} \cdot X_{test}$$

Paso 3: Utilizamos la técnica del error cuadrático medio para ver la validez de los resultados, comparando Y_{test} con $\widehat{Y_{test}}$.

De acuerdo con lo expuesto hasta ahora el entrenamiento se desarrolla en las siguientes condiciones:

Ejemplos de entrenamiento	135
Ejemplos de test	24
Variables consideradas	10

Los resultados se pueden ver en la tabla comparativa del apartado 4.3.1.2.

4.3.1.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

Se desarrollan siete modelos de regresión y en cada uno de ellos se añadirán las variables (N, CP, CT) de una asignatura (mecánica [A], mecánica de fluidos general [B], mecánica de fluidos aplicada [C], teoría de sistemas [D], tecnología de fabricación y tecnología de máquinas [E], teoría de estructuras y construcciones industriales [F], ingeniería térmica y de fluidos [G]) al conjunto de entrada existente previamente, partiendo del conjunto formado por las 10 variables PAU.

Ejemplos de entrenamiento	121
Ejemplos de test	22
Variables consideradas	13 (16, 19, 22, 25, 28, 31)

- Evolución de los estimadores.

	PAU	A	B	C	D	E	F	G
$\hat{\beta}_0$	-1.0157	-1.1575	-0.8678	-1.1661	-1.3798	-1.4295	-1.6143	-1.4806
$\hat{\beta}_1$	-0.0356	-0.0197	-0.0169	-0.0029	0.0054	0.0341	0.0489	0.0198
$\hat{\beta}_2$	-0.1826	-0.1521	-0.0665	-0.0195	-0.0068	0.1165	0.1789	0.0641
$\hat{\beta}_3$	0.1760	0.1445	0.0863	0.0603	0.0623	0.0518	0.0417	0.0228
$\hat{\beta}_4$	0.7838	0.6774	0.7981	0.8418	0.8058	0.7577	0.7392	0.5955
$\hat{\beta}_5$	0.3223	0.2854	0.1827	0.1591	0.1622	0.1504	0.1357	0.1642
$\hat{\beta}_6$	0.9554	0.8114	0.2349	0.1084	0.2380	0.1954	0.1357	0.3140
$\hat{\beta}_7$	-0.0729	-0.0475	-0.0060	0.0111	0.0042	0.0129	0.0243	0.0310
$\hat{\beta}_8$	-0.4509	-0.2561	-0.0176	0.0017	0.0061	0.0294	0.0900	0.1548
$\hat{\beta}_9$	-0.0720	-0.0608	-0.0439	-0.0171	-0.0066	-0.0045	0.0064	0.0015
$\hat{\beta}_{10}$	-0.2700	-0.2606	-0.3438	-0.2407	-0.1799	-0.1505	-0.0928	-0.1215
$\hat{\beta}_{11}$	-	0.0692	0.0467	0.0438	0.0401	0.0360	0.0317	0.0314
$\hat{\beta}_{12}$	-	-0.0493	-0.0175	-0.0124	-0.0123	-0.0050	-0.0034	-0.0028

$\hat{\beta}_{13}$	-	-0.0210	-0.0036	-0.0015	-0.0006	0.0022	0.0027	0.0085
$\hat{\beta}_{14}$	-	-	0.1184	0.1151	0.0934	0.0939	0.0930	0.0871
$\hat{\beta}_{15}$	-	-	-0.0220	-0.0185	-0.0143	-0.0112	-0.0109	-0.0137
$\hat{\beta}_{16}$	-	-	-0.0151	-0.0130	-0.0065	-0.0072	-0.0109	-0.0042
$\hat{\beta}_{17}$	-	-	-	0.0404	0.0412	0.0375	0.0328	0.0367
$\hat{\beta}_{18}$	-	-	-	-0.0049	-0.0094	-0.0102	-0.0088	-0.0072
$\hat{\beta}_{19}$	-	-	-	-0.0009	0.0082	0.0123	0.0109	0.0122
$\hat{\beta}_{20}$	-	-	-	-	0.0410	0.0404	0.0375	0.0337
$\hat{\beta}_{21}$	-	-	-	-	-0.0328	-0.0312	-0.0313	-0.0169
$\hat{\beta}_{22}$	-	-	-	-	-0.0243	-0.0247	-0.0241	-0.0238
$\hat{\beta}_{23}$	-	-	-	-	-	0.0021	0.0033	0.0029
$\hat{\beta}_{24}$	-	-	-	-	-	-0.0384	-0.0410	-0.0372
$\hat{\beta}_{25}$	-	-	-	-	-	-0.0057	-0.0049	-0.0055
$\hat{\beta}_{26}$	-	-	-	-	-	-	0.0221	0.0143
$\hat{\beta}_{27}$	-	-	-	-	-	-	0.0056	0.0062
$\hat{\beta}_{28}$	-	-	-	-	-	-	0.0043	0.0009
$\hat{\beta}_{29}$	-	-	-	-	-	-	-	0.0189
$\hat{\beta}_{30}$	-	-	-	-	-	-	-	-0.0245
$\hat{\beta}_{31}$	-	-	-	-	-	-	-	-0.0156

- Características comparativas.

	PAU	A	B	C	D	E	F	G
|Error máximo|	0.8944	0.6152	0.3455	0.2932	0.2593	0.2749	0.3144	0.3313
MSE	0.0678	0.0435	0.0435	0.0235	0.0156	0.0164	0.0166	0.0214

- Evolución de las estimaciones respecto al Target para el conjunto de test.

Target	PAU	A	B	C	D	E	F	G
1.6511	1.7587	1.8494	1.7479	1.6774	1.7006	1.6989	1.7050	1.6500
1.8670	1.6496	1.7821	1.8707	1.8432	1.8028	1.8177	1.8382	1.8080
1.5365	1.8155	1.8274	1.7384	1.6953	1.6244	1.6628	1.6377	1.6258
1.5837	1.6053	1.6326	1.7623	1.7101	1.5622	1.4970	1.4795	1.5006
2.1728	2.0777	2.0922	2.0804	2.1242	2.1245	2.1620	2.1277	2.1219
1.5841	1.5642	1.7002	1.7033	1.6830	1.6829	1.7225	1.7345	1.6809
2.6553	1.7609	2.0401	2.3197	2.4529	2.5420	2.5420	2.5263	2.5305
1.9915	1.6831	1.7464	1.9039	1.9621	1.9900	2.0265	2.0690	2.0220
1.7532	1.6083	1.5710	1.6847	1.6028	1.4971	1.4829	1.4388	1.4891
1.3975	1.4816	1.3103	1.6023	1.5985	1.4612	1.4194	1.3937	1.5554
1.9087	1.7266	1.8193	1.8756	1.8113	1.8037	1.8316	1.8533	1.9052
1.4402	1.5161	1.6193	1.6736	1.6075	1.4945	1.5286	1.5360	1.5545
1.4647	1.5307	1.5563	1.5614	1.5523	1.5415	1.5852	1.6014	1.6670

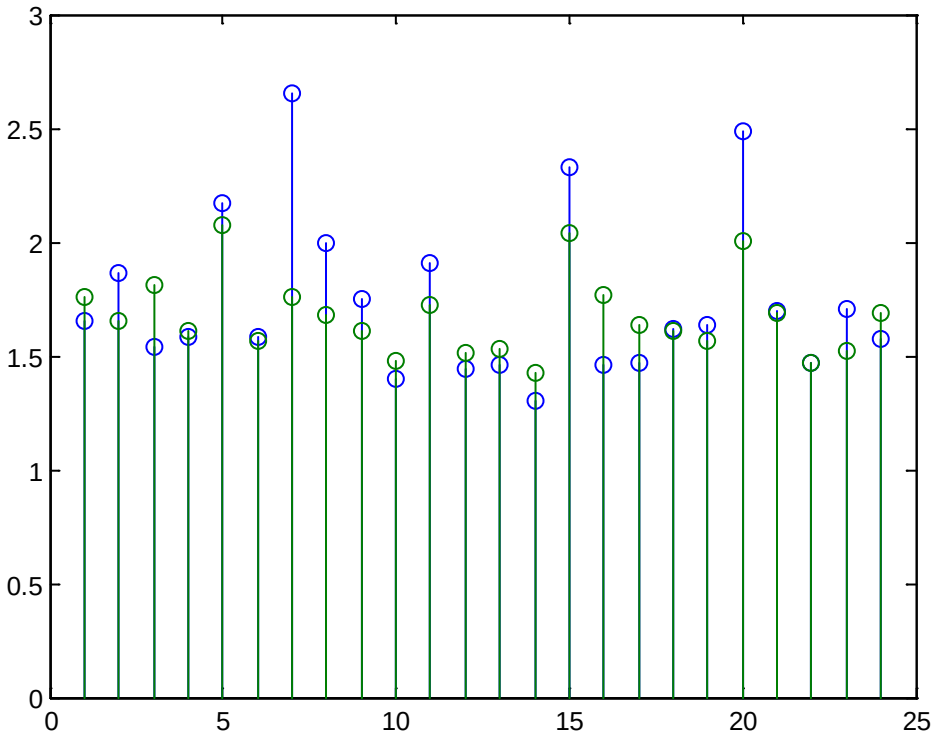
1.3022	1.4272	1.3831	1.4067	1.3556	1.2706	1.2532	1.2534	1.2315
2.3319	2.0377	2.1544	2.2015	2.2283	2.1681	2.1938	2.2099	2.1860
1.4569	1.7698	1.7814	1.6958	1.6293	1.7162	1.7318	1.7179	1.7882
1.4698	1.6359	1.5167	1.7941	1.7171	1.6473	1.5444	1.4944	1.5408
1.6173	1.6126	1.5428	1.8122	1.7452	1.7100	1.7461	1.7261	1.8160
1.6396	1.5671	-	-	-	-	-	-	-
2.4854	2.0019	2.2000	2.3392	2.3486	2.3312	2.3681	2.3760	2.3400
1.6996	1.6928	1.6629	1.6271	1.5718	1.6281	1.6401	1.6569	1.6943
1.4681	1.4680	1.4186	1.7819	1.7613	1.6046	1.5964	1.5651	1.6250
1.7019	1.5198	-	-	-	-	-	-	-
1.5726	1.6868	1.7724	1.9181	1.8401	1.7488	1.7846	1.7809	1.7962

- Evolución de los errores.

Target	PAU	A	B	C	D	E	F	G
1.6511	-0.1076	-0.1983	-0.0968	-0.0263	-0.0495	-0.0478	-0.0539	0.0011
1.8670	0.2174	0.0849	-0.0037	0.0238	0.0642	0.0493	0.0288	0.0590
1.5365	-0.2791	-0.2909	-0.2019	-0.1588	-0.0879	-0.1263	-0.1012	-0.0893
1.5837	-0.0216	-0.0489	-0.1786	-0.1264	0.0215	0.0867	0.1042	0.0831
2.1728	0.0952	0.0806	0.0924	0.0486	0.0483	0.0108	0.0451	0.0509
1.5841	0.0199	-0.1161	-0.1192	-0.0989	-0.0988	-0.1384	-0.1504	-0.0968
2.6553	0.8944	0.6152	0.3356	0.2024	0.1133	0.1133	0.1290	0.1248
1.9915	0.3083	0.2451	0.0876	0.0294	0.0015	-0.0350	-0.0775	-0.0305
1.7532	0.1449	0.1822	0.0685	0.1504	0.2561	0.2703	0.3144	0.2641
1.3975	-0.0842	0.0872	-0.2048	-0.2010	-0.0637	-0.0219	0.0038	-0.1579
1.9087	0.1821	0.0894	0.0331	0.0974	0.1050	0.0771	0.0554	0.0035
1.4402	-0.0759	-0.1791	-0.2334	-0.1673	-0.0543	-0.0884	-0.0958	-0.1143
1.4647	-0.0660	-0.0916	-0.0967	-0.0876	-0.0768	-0.1205	-0.1367	-0.2023
1.3022	-0.1250	-0.0809	-0.1045	-0.0534	0.0316	0.0490	0.0488	0.0707
2.3319	0.2943	0.1775	0.1304	0.1036	0.1638	0.1381	0.1220	0.1459
1.4569	-0.3129	-0.3245	-0.2389	-0.1724	-0.2593	-0.2749	-0.2610	-0.3313
1.4698	-0.1660	-0.0469	-0.3243	-0.2473	-0.1775	-0.0746	-0.0246	-0.0710
1.6173	0.0046	0.0745	-0.1949	-0.1279	-0.0927	-0.1288	-0.1088	-0.1987
1.6396	0.0725	-	-	-	-	-	-	-
2.4854	0.4835	0.2854	0.1462	0.1368	0.1542	0.1173	0.1094	0.1454
1.6996	0.0067	0.0367	0.0725	0.1278	0.0715	0.0595	0.0427	0.0053
1.4681	0.0001	0.0495	-0.3138	-0.2932	-0.1365	-0.1283	-0.0970	-0.1569
1.7019	0.1821	-	-	-	-	-	-	-
1.5726	-0.1142	-0.1998	-0.3455	-0.2675	-0.1762	-0.2120	-0.2083	-0.2236

4.3.1.3. Conclusiones.

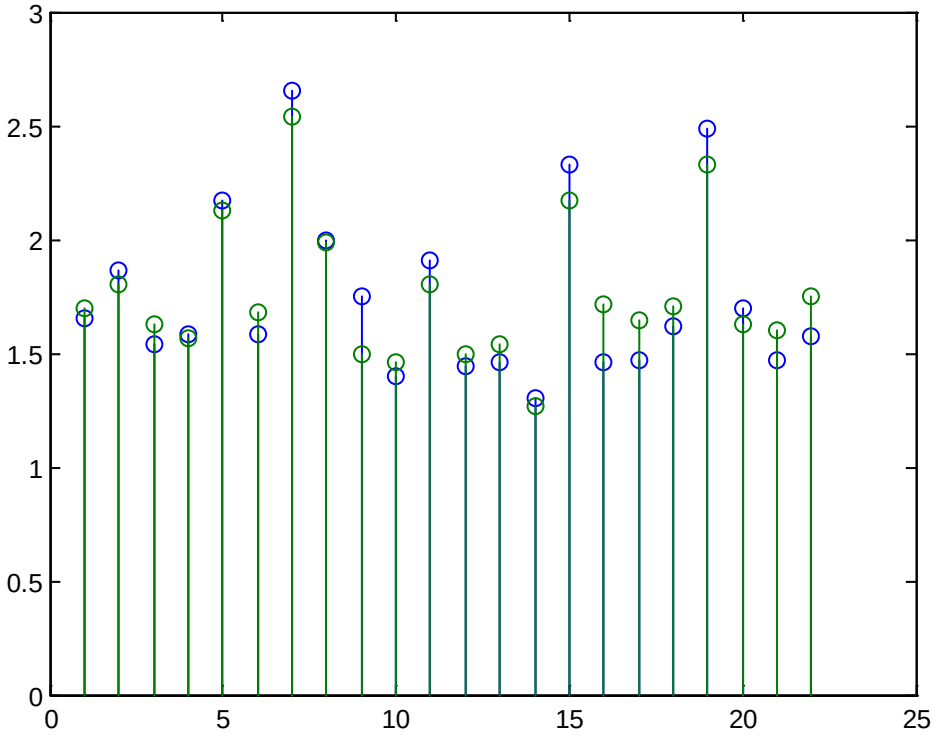
En primer lugar, destacan los buenos resultados obtenidos para la predicción de nota media de egreso de la carrera únicamente a partir de variables PAU, con un MSE de 0.0678. Veamos la comparación de los Targets con las predicciones:



En 10 de los 24 alumnos del conjunto Test, el error cometido es inferior a 0.1 puntos y en 17 de los 24 ejemplos, el error es inferior a 0.2 puntos. Los errores más grandes se dan generalmente cuanto mayor es la nota media de egreso del alumno debido a que la mayoría de los ejemplos de entrenamiento tienen notas entre 1 y 1.8. Destaca el error cometido en el ejemplo 7, que es de casi 9 décimas. Se predice un valor 9 décimas inferior a su nota media real a partir de variables PAU. El error disminuye significativamente conforme se van introduciendo variables de las asignaturas de la titulación. Este alumno tiene unas calificaciones PAU bastante similares al resto, lo que provoca que la regresión lo incluya entre la mayoría, con una nota media de egreso predicha de 1.76 (la real es 2.65). Sin embargo, en las asignaturas complicadas de la carrera obtiene muy buenos resultados por la influencia de factores no considerados en este estudio (motivación, …) y la regresión es capaz a través de las variables de estas asignaturas de predecir su nota con un error mínimo de 0.1133.

Además, los resultados obtenidos al ir introduciendo las asignaturas también mejoran los obtenidos para ingeniería de telecomunicaciones con un MSE mínimo de 0.0156 obtenido al introducir teoría de sistemas. Es significativo que al introducir esta asignatura en RNA también se haya obtenido una de las mejores iteraciones, con un MSE

de 0.0767. Parece que la forma de abordar esta asignatura por parte de los alumnos está relacionada con la forma en general con que abordan la carrera, lo que ayuda a explicar su gran influencia en los resultados. Veamos el mapa comparativo:



En 13 de los 22 ejemplos de Test, el error cometido es inferior a 0.1 puntos y en 20 de los 22 ejemplos el error es inferior a 0.2 puntos.

Veamos la comparación entre Regresión Lineal Múltiple (RLM) y RNA:

		PAU	A	B	C	D	E	F	G
RLM	|Error máximo|	0.8944	0.6152	0.3455	0.2932	0.2593	0.2749	0.3144	0.3313
	MSE	0.0678	0.0435	0.0435	0.0235	0.0156	0.0164	0.0166	0.0214
RNA	|Error máximo|	0.7938	0.474	0.7904	1.123	0.5123	0.5911	0.6968	0.7018
	MSE	0.0729	0.0765	0.1126	0.1354	0.0767	0.1164	0.0841	0.0716

Vemos que los resultados obtenidos mediante RLM son mejores que los obtenidos mediante RNA en general. Además se observa más o menos la disminución progresiva del error al introducir las asignaturas, cosa que no ocurre con las RNA.

5. Capítulo V: Estudio y predicción de tasa de abandono.

5.1. Objetivo.

En el contexto económico en el que nos hallamos inmersos, donde la competitividad es un factor clave, se hace preciso que la preparación de las generaciones venideras sea lo más óptima posible en todos los ámbitos, y con los menores gastos posibles. De aquí surge la importancia de este capítulo. Se pretende crear una red neuronal de clasificación con capacidad para predecir potenciales abandonos, alumnos que tienen grandes posibilidades de abandonar la carrera universitaria que iniciaron. Ello podría ayudar a la universidad para tomar medidas de refuerzo sobre tales alumnos y así, evitar este desenlace. Además, también podría llegar a ser útil para llevar a cabo estimaciones del conjunto de alumnos que acabarán la carrera de cada promoción de entrada, o de cada grupo de alumnos tras su primer año de matrícula.

Trataremos de predecir dos casos: por un lado, el caso de aquel alumno que tras su primer año matriculado, decide abandonar la carrera y no se matricula el segundo año y, por otro lado, la de aquel alumno que abandona en los cursos sucesivos y, por tanto, no está matriculado en su año teórico de egreso ni al siguiente (definición oficial de abandono). Para ello, y según el caso, trataremos de hacerlo inicialmente a partir de variables PAU, a continuación introduciremos variables que explican cómo ha ido su primer año de matrícula y, finalmente, introduciremos variables que explican su segundo año como alumno matriculado. Esta introducción cronológica se hace porque, tal y como se explicó en el capítulo III, lo que se pretende es poder predecir el abandono de la manera más fiable posible, pero también de la manera más prematura posible. Por tanto, lo óptimo sería poder clasificar un potencial abandono como tal, de manera fiable, únicamente a partir de variables PAU. Los conjuntos de datos contienen, así pues, tanto alumnos que abandonan (en el primer curso, o en los sucesivos) como alumnos que consiguen terminar la titulación (egresados) para que la red pueda aprender ambos patrones de comportamiento.

5.2. Clasificación a partir de variables PAU.

Se trata de predecir si un alumno abandonará o no la carrera en cuestión (Ingeniería Industrial o Telecomunicaciones), en función de los conocimientos demostrados en la PAU. Para ello, emplearemos las variables comentadas en capítulos anteriores, a saber:

Variable
Nota PAU en la parte 1
Ranking parte 1/Total presentados
Nota PAU en la parte 2
Ranking parte 2/Total presentados
Calificación PAU
Ranking calificación PAU/Total presentados
Calificación física PAU
Ranking calificación física/Total presentados
Calificación Matemáticas PAU
Ranking calificación matemáticas/Total presentados

5.2.1. Alumnos de ingeniería industrial.

En primer lugar hagamos un análisis previo de los datos, para ver las características del conjunto empleado para entrenar la red.

Variable	Media	Desviación típica
Nota PAU en la parte 1	6.8106	1.2310
Ranking parte 1/Total presentados	0.3092	0.2500
Nota PAU en la parte 2	6.6645	1.6576
Ranking parte 2/Total presentados	0.2931	0.2648
Calificación PAU	7.3804	1.1279
Ranking calificación PAU/Total presentados	0.3040	0.2588
Calificación física PAU	6.2415	2.1619
Ranking calificación física/Total presentados	0.3593	0.2584
Calificación Matemáticas PAU	7.0403	2.1242
Ranking calificación matemáticas/Total presentados	0.3570	0.2752

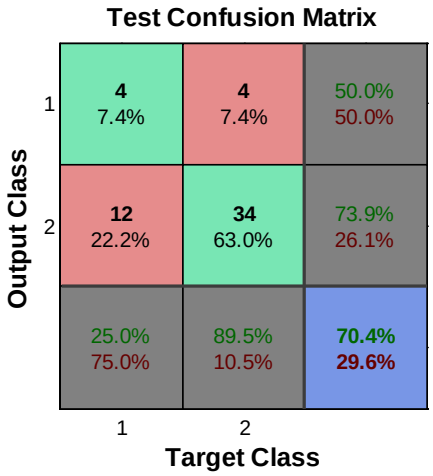
Vemos que el valor medio de las variables describe a un alumno con notas entre 6 y 7 sobre 10 y situado en una posición buena en los rankings. De cada 100 alumnos que se presenten a selectividad, aquel que ingresa a ingeniería industrial se halla en general entre los 30 que mejores calificaciones PAU obtienen.

5.2.1.1. No matriculados el segundo año.

En primer lugar trataremos de predecir a partir de variables PAU, si un alumno matriculado en el primer curso de Ingeniería industrial abandonará la carrera y no se matriculará el segundo año.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	357
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/12-2/17

Resultados:



Con 1 aparecen marcados los alumnos que abandonan y con 2 los que no abandonan. Tenemos un conjunto de test formado por 54 alumnos, de los cuales, 16 abandonaron la carrera y 38, no. Tras probar distintas arquitecturas, la red únicamente ha sido capaz de predecir 4 de esos 16 abandonos. De los 38 alumnos que no abandonaron, ha predicho 34 como tal.

Veamos tres ejemplos en los que la red predice un “no abandono” y finalmente el alumno abandona, para entender mejor la complejidad el problema:

Alumno	PAU1	R_PAU1	PAU2	R_PAU2	PAUT	R_PAUT	FIS	R_FIS	MAT	R_MAT
1	7,6	0,106	7,1	0,133	8,0	0,122	7,8	0,133	5,4	0,579
2	5,6	0,570	6,9	0,154	6,5	0,514	6,9	0,257	5,8	0,521
3	7,6	0,126	7,0	0,174	8,0	0,138	4,6	0,446	8,0	0,235

Son alumnos con notas bastante buenas en general y con ranking también bastante buenos por lo que la red no es capaz de predecir su abandono. Analicemos la salida de la red para estos tres alumnos:

Alumno	Abandona	No abandona
1	0.1289	0.9128
2	0.2843	0.7046
3	0.1353	0.8541

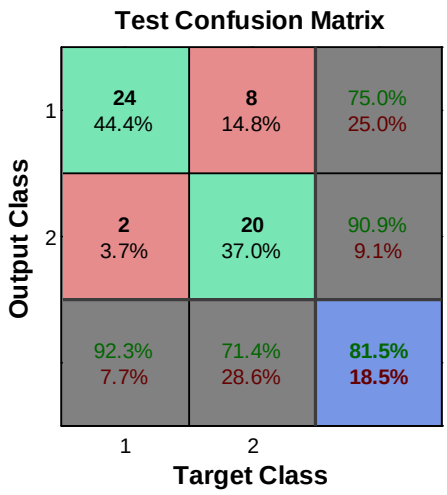
Como vemos, de acuerdo con el entrenamiento recibido, la red los clasifica casi inequívocamente como “no abandonos” y al final resulta que abandonan.

5.2.1.2. Abandono oficial.

Ahora se trata de predecir a partir de variables PAU, aquellos alumnos que no estarán matriculados en el año teórico de egreso o al siguiente.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	357
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/15-2/20

Resultados:



Vemos que de los 52 alumnos que reservamos para test, 26 abandonan y 28 no. De esos 26, la red es capaz de predecir 24 abandonos correctamente. En cambio, de los 28 que no abandonan, se equivoca en 8, asumiendo que abandonan.

Alumno	PAU1	R_PAU1	PAU2	R_PAU2	PAUT	R_PAUT	FIS	R_FIS	MAT	R_MAT
1	8,70	0,02	6,90	0,18	8,00	0,13	6,80	0,19	6,80	0,43
2	7,40	0,14	7,20	0,17	6,90	0,37	7,00	0,33	7,00	0,47
3	7,90	0,06	6,90	0,22	7,20	0,27	6,40	0,48	6,90	0,43
4	5,50	0,56	5,30	0,55	6,60	0,48	6,80	0,17	5,00	0,78
5	7,00	0,17	6,80	0,24	7,60	0,18	6,90	0,16	5,90	0,67
6	6,30	0,45	6,30	0,31	7,00	0,38	5,10	0,50	7,20	0,27
7	7,90	0,11	7,00	0,20	8,30	0,10	7,10	0,21	5,50	0,28
8	7,90	0,06	7,50	0,09	8,10	0,09	8,60	0,14	5,50	0,22

Alumno	Abandona	No abandona
1	0.7262	0.3562
2	0.7315	0.3025
3	0.6926	0.3273
4	0.7829	0.2815
5	0.6113	0.4629
6	0.6494	0.4003
7	0.5642	0.4994
8	0.5994	0.4814

Vemos por un lado que de acuerdo con sus calificaciones PAU se trata en general de buenos alumnos, por lo que no se entiende muy bien que la red los clasifique como potenciales abandonos. En cambio, por otro lado, vemos que la salida de la red indica, en general, para estos alumnos, duda, en la medida en que las dos variables de salida tienen valores próximos entre sí y a 0.5 para la mayoría de los casos. Para el caso que menos duda presenta es para alumno 4: la red dice que abandona y finalmente el alumno no abandona pero parece lógico que sea el que tiene mayor posibilidad de abandono si se atiende a sus notas PAU pues de los 8 alumnos es el que peor calificación tiene en general.

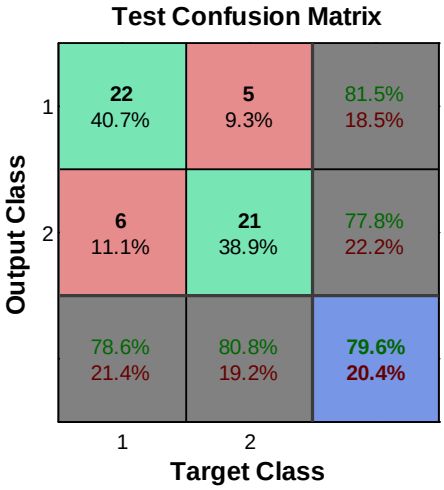
Alumno	PAU1	R_PAU1	PAU2	R_PAU2	PAUT	R_PAUT	FIS	R_FIS	MAT	R_MAT
1	5.1000	0.6469	7.8000	0.0986	7.4000	0.2401	8.8000	0.1053	6.4000	0.5665
2	6.9000	0.2179	9.2000	0.0117	7.6000	0.2000	9.0000	0.0835	9.3000	0.0963

Alumno	Abandona	No abandona
1	0.4539	0.6987
2	0.3442	0.6361

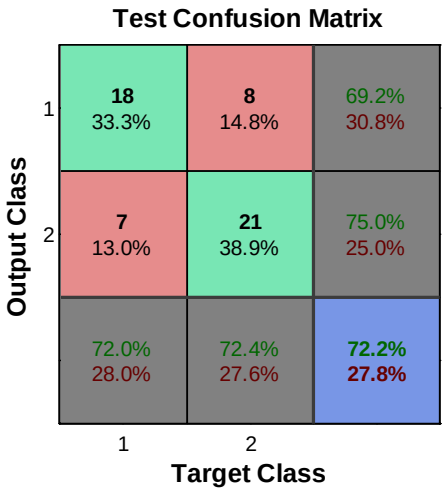
Por sus calificaciones PAU vemos que se trata en general de buenos alumnos. En un grupo de 100 alumnos se encontrarían clasificados entre los 25 mejores atendiendo a su calificación PAU total y, además, tienen buenas notas en las asignaturas específicas. Por ello la red predice que no se producirá el abandono. En cambio, éste se produce.

Otras arquitecturas permiten obtener otros resultados que, en cualquier caso, son bastante buenos:

Arquitectura: Capa oculta i/neuronas capa oculta i	1/5-2/5



Arquitectura: Capa oculta i/neuronas capa oculta i	1/15



Es preciso destacar que la variabilidad de los resultados a igualdad de arquitecturas es consecuencia de dos factores: la inicialización aleatoria de los pesos en la fase de

aprendizaje de la red, y la selección aleatoria de los conjuntos de test, validación y entrenamiento, que también cambia de un entrenamiento a otro.

5.3. Clasificación con variables de primer curso.

5.3.1. Alumnos de Ingeniería Industrial.

En una primera aproximación a los datos, calculemos algunos indicadores que nos den una idea de la situación media de los alumnos considerados tras su primer año matriculado:

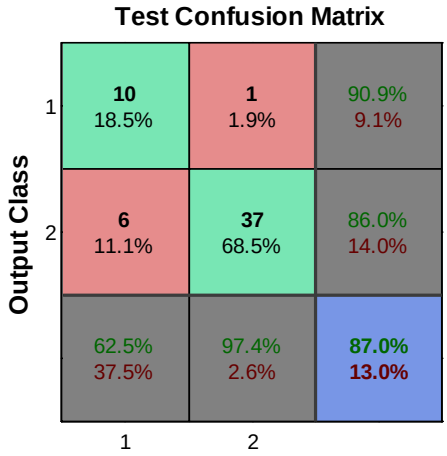
Variable	Media	Desviación típica
Créditos matriculados	70.5210	2.7134
Tasa evaluados	0.7296	0.2538
Tasa presentados	0.9597	0.3444
Tasa rendimiento	0.5184	0.3402
Tasa éxito	0.6350	0.3275
Número de convocatorias	18.3501	4.1817
Tasa NP	0.4272	0.2684
Tasa 0	0.2340	0.1486
Tasa 1	0.1790	0.1441
Tasa 2	0.1065	0.1377
Tasa 3	0.0334	0.0865
Tasa 4	0.0198	0.0701

Vemos que los alumnos son evaluados en su primera matrícula de tres de cada cuatro asignaturas aproximadamente. Se suelen presentar casi a tantos créditos como están matriculados aunque no hemos de olvidar que en el indicador “Tasa presentados” si un alumno se presenta dos veces a la misma asignatura cuenta por el doble de créditos presentados. En general, en la primera matrícula, los alumnos suelen aprobar la mitad de los créditos de los que están matriculados, y un 65% aproximadamente de los créditos a los que se presentan.

5.3.1.1. Alumnos no matriculados el segundo año.

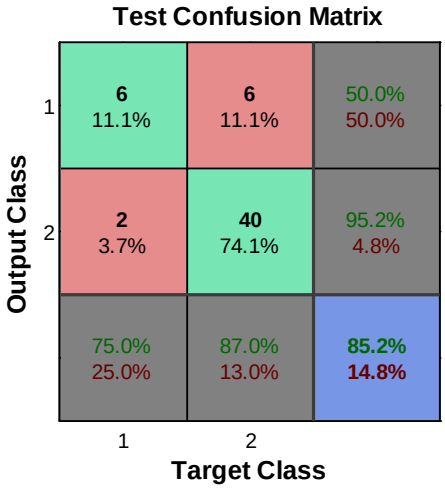
En primer lugar hacemos pruebas usando en el conjunto de inputs tanto variables PAU como las variables de primer curso, obteniendo resultados con las siguientes características:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	357
Variables consideradas	22
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25



Los resultados han mejorado bastante respecto a los obtenidos utilizando únicamente variables PAU. Por ello, decidimos quitar las variables PAU y tratar de predecir el abandono en segundo curso a partir, únicamente, de variables de primer curso. Los resultados se muestran a continuación:

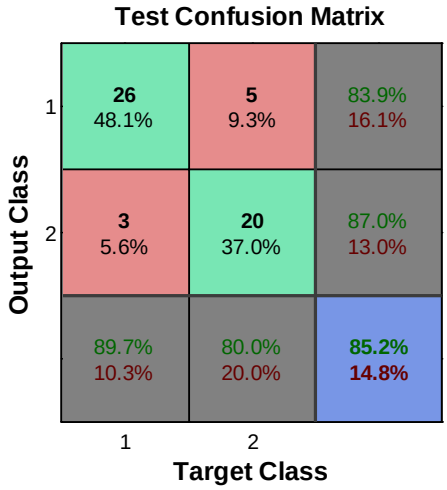
Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	357
Variables consideradas	12
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25



Vemos que los resultados son de similares características a los obtenidos utilizando tanto variables PAU como variables del primer año de matrícula. En este sentido podemos afirmar que las características PAU no influyen a la hora de predecir un potencial abandono en el segundo año de matrícula y que las variables más influyentes son las relacionadas con el primer año de matrícula.

5.3.1.2. Abandono oficial.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	357
Variables consideradas	22
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25



Alumnos que abandonan y que se predicen como que no abandonan:

Alumno	1	2	3
Cred_matr	69.0000	69.0000	69.0000
Tasaevaluados	1.0000	0.9348	1.0000
Tasapresentados	1.0000	1.2391	1.2391
Tasarendimiento	1.0000	0.8696	0.9130
Tasaexito	1.0000	0.9302	0.9130
NumConv	10.0000	16.0000	15.0000
TasaNP	0	0.2500	0.2000
Tasa0	0	0.2500	0.2000
Tasa1	0.6000	0.3125	0.3333
Tasa2	0.4000	0.1875	0.2667
Tasa3	0	0	0
Tasa4	0	0	0
PAU_1	5.9000	6.9000	6.4000
R_PAU_1	0.5648	0.2918	0.4131
PAU_2	7.1000	7.9000	7.6000
R_PAU_2	0.1948	0.1222	0.1757
PAU_T	7.6000	8.2000	7.9000
R_PAU_T	0.2286	0.1266	0.1722
FISICA	6.1000	5.7000	5.1000
R_FISICA	0.3190	0.3286	0.4203
MAT	8.1500	9.5000	9.8000
R_MAT	0.2918	0.0820	0.0556

Alumno	1	2	3
Abandona	0.1819	0.2748	0.2291
No abandona	0.8681	0.8818	0.8910

Se trata de tres alumnos bastante buenos, en general. Un vistazo a su rendimiento en el primer año de matrícula y a sus calificaciones PAU nos hace pensar que se trata de alumnos con un rendimiento normal-bueno. No tienen notas muy altas pero su tasa de éxito está próxima a 1. La red no es capaz de predecir su abandono porque en problemas de la naturaleza que aquí se aborda participan variables aleatorias cuya consideración no es sencilla y que pueden ser determinantes: enfermedad, descubrimiento por parte del alumno de que le gusta otra carrera, etc.

Alumnos que no abandonan y que la red predice abandono:

Alumno	1	2	3	4	5
Cred_matr	69.0000	72.0000	81.0000	69.0000	73.5000
Tasaevaluados	0.4130	1.0000	0.5741	0.9130	1.0000
Tasapresentados	0.6522	1.4167	0.6111	1.4783	1.4694
Tasarendimiento	0.1304	0.5417	0.2037	0.7826	0.7755
Tasaexito	0.3158	0.5417	0.3548	0.8571	0.7755
NumConv	20.0000	18.0000	24.0000	21.0000	16.0000
TasaNP	0.7500	0.1667	0.7500	0.3333	0.0625
Tasa0	0.2000	0.5556	0.2083	0.3333	0.5000
Tasa1	0.0500	0.2222	0	0.2381	0.4375
Tasa2	0	0.0556	0.0417	0.0476	0
Tasa3	0	0	0	0.0476	0
Tasa4	0	0	0	0	0
PAU_1	5.8000	6.3000	8.2000	7.0000	6.8000
R_PAU_1	0.5101	0.3351	0.0392	0.2727	0.3188
PAU_2	5.6000	8.2000	7.1000	5.5000	4.5000
R_PAU_2	0.3969	0.0592	0.1775	0.4873	0.7001
PAU_T	6.5000	8.3000	8.7000	7.1000	6.7000
R_PAU_T	0.5073	0.0934	0.0473	0.3555	0.4706
FISICA	4.7000	7.7000	6.7000	4.6000	5.8000
R_FISICA	0.5893	0.2239	0.3697	0.5866	0.3985
MAT	5.0000	8.7000	8.3000	4.8000	5.1000
R_MAT	0.6398	0.1820	0.2328	0.7053	0.6548

Alumno	1	2	3	4	5
Abandono	0.9647	0.6102	0.6387	0.6300	0.5796
No abandono	0.0453	0.3101	0.2093	0.5494	0.3229

Son en general alumnos cuyas características de primera matrícula son bastante deficientes, especialmente las de los números 1, 2 y 3. El primer alumno aprueba únicamente 1 de 20 convocatorias de asignaturas, El segundo, aprueba una de cada cinco convocatorias y en una de veinte obtiene notable. Y el tercero obtiene notable en una de veinte convocatorias y el resto o no se presenta o suspende. Para estos tres alumnos son para los que la red presenta más certeza, especialmente para el primero. Los alumnos cuatro y cinco son ligeramente mejores que los anteriores y por ello, la red presenta un poco más de incertidumbre.

En este sentido hay que recordar lo que estamos tratando de predecir: abandono oficial a partir de características PAU y características del primer año de matrícula. Puede suceder y, de hecho, los casos en los que la red se equivoca suelen estar asociados a

ellos, que hay casos en los que el alumno no hace una buena selectividad, consigue entrar a la carrera porque la nota de corte no es muy alta y en primer curso no obtiene buenos resultados. Ello no le impide continuar y conseguir acabar la carrera, lo que supone, a la postre, no abandonar.

5.4. Clasificación con variables de segundo curso.

A los conjuntos de variables ya expuestos anteriormente, se añaden ahora las mismas variables explicadas e introducidas para la primera matrícula del alumno, pero para su segunda matrícula. En este caso, lógicamente, únicamente estamos en disposición de predecir el abandono oficial y no el de segundo curso.

5.4.1. Abandono en Ingeniería Industrial.

Veamos un primer análisis de las variables:

Variable	Media	Desviación típica
Créditos matriculados	70.3391	11.0646
Tasa evaluados	0.7544	0.2215
Tasa presentados	1.0120	0.3267
Tasa rendimiento	0.5651	0.2991
Tasa éxito	0.7047	0.2762
Número de convocatorias	20.7586	5.4426
Tasa NP	0.4296	0.2482
Tasa 0	0.2230	0.1378
Tasa 1	0.2110	0.1414
Tasa 2	0.1058	0.1487
Tasa 3	0.0222	0.0701
Tasa 4	0.0083	0.0346

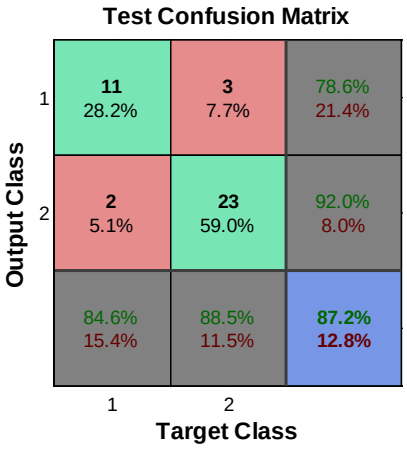
Comparación de las variables de primero con las variables de segundo:

Variable	Media primero	Media segundo
Créditos matriculados	70.3391	70.5210
Tasa evaluados	0.7544	0.7296
Tasa presentados	1.0120	0.9597
Tasa rendimiento	0.5651	0.5184
Tasa éxito	0.7047	0.6350
Número de convocatorias	20.7586	18.3501
Tasa NP	0.4296	0.4272
Tasa 0	0.2230	0.2340
Tasa 1	0.2110	0.1790
Tasa 2	0.1058	0.1065
Tasa 3	0.0222	0.0334
Tasa 4	0.0083	0.0198

No se aprecian cambios significativos entre las características de los alumnos en su primer año de matrícula y las características de los alumnos en su segundo año de matrícula.

Entrenamos la red:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	261
Variables consideradas	34
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/40



Se viene explicando a lo largo de todo este apartado de tasa de abandono que existen ciertos casos especiales o particulares, alumnos que, aun teniendo unas notas muy buenas, finalmente abandonan y otros que, aun obteniendo calificaciones bastante

malas los primeros años deciden continuar y terminar sus estudios. Dependiendo del conjunto en el que caigan estos casos particulares (test, validación o entrenamiento) en la distribución aleatoria de ejemplos los resultados pueden variar bastante.

Si caen en el conjunto de entrenamiento sucede que falsean los pesos en el sentido de que generan variaciones de ellos que no contribuyen a la generalización, sino a adaptarse a casos particulares cuya predicción no depende de las variables consideradas en el conjunto de entrada.

El conjunto de validación era el que se encargaba de asegurar la generalización de la red mediante lo que llamábamos "early stopping", que consistía en detener el entrenamiento si durante un número determinado de iteraciones comenzaba a incrementarse el error cometido. Sucede entonces que los casos especiales que se incluyen en este conjunto también provocan un deterioro de la generalización ya que provocan la parada prematura del entrenamiento de la red: dan errores cuadráticos de la red mayores que los que se obtendrían en el caso de que en el lugar del caso particular, tuviéramos un caso habitual o general.

En el caso de caer estos casos particulares en el conjunto de test, se falsean los resultados pues a pesar de que se haya conseguido una buena generalidad con los conjuntos de entrenamiento y validación, la red se equivocará con las excepciones y nos mostrará un error que estaba destinado a cometerse desde el mismo momento en que se seleccionaron las variables involucradas en la clasificación.

Veamos alumnos que abandonan y la red predice como no abandonos:

	Alumno	1	2
Primero	Cred_matr	69.0000	76.5000
	Tasaevaluados	1.0000	0.9412
	Tasapresentados	1.4783	1.4902
	Tasarendimiento	0.9348	0.6667
	Tasaexito	0.9348	0.7083
	NumConv	15.0000	22.0000
	TasaNP	0.0667	0.3182
	Tasa0	0.3333	0.4091
	Tasa1	0.3333	0.2273
	Tasa2	0.2000	0.0455
	Tasa3	0	0
	Tasa4	0.0667	0
Segundo	Cred_matr	76.5000	97.5000
	Tasaevaluados	0.6863	0.8769
	Tasapresentados	0.9020	1.0000
	Tasarendimiento	0.6078	0.4462
	Tasaexito	0.8857	0.5088
	NumConv	25.0000	32.0000

	TasaNP	0.5600	0.5000
	Tasa0	0.1600	0.2813
	Tasa1	0.1600	0.2188
	Tasa2	0.0800	0
	Tasa3	0	0
	Tasa4	0.0400	0
PAU	PAU_1	7.3000	7.6000
	R_PAU_1	0.2120	0.1592
	PAU_2	7.4000	6.7000
	R_PAU_2	0.2017	0.3459
	PAU_T	8.1000	7.8000
	R_PAU_T	0.1414	0.2100
	FISICA	6.6000	4.5000
	R_FISICA	0.2101	0.5373
	MAT	7.5000	9.5000
	R_MAT	0.4286	0.1120

Alumno	1	2
Abandona	0.0676	0.1260
No abandona	0.9651	0.6385

En el primer alumno, la red presenta muy poca duda: no abandonará. Se trata de un alumno con calificaciones PAU bastante buenas y con rankings también muy buenos. En su primer año consigue unas tasas de éxito y rendimiento muy próximas a 1 y aprueba o saca notable en más del 50% de las convocatorias. En el segundo curso, desciende el porcentaje de convocatorias aprobadas bastante y no se presenta a más de la mitad. Aun así, la tasa de éxito que consigue es bastante alta. Todo ello hace que la red concluya un no abandono. Sin embargo el alumno, finalmente abandona, probablemente por la influencia de variables no consideradas en este estudio: causas personales, enfermedad, etc.

En el segundo alumno, la red presenta más duda. No es un alumno con las características de un potencial abandono, aunque tampoco está próximo a 1 el valor relacionado con el no abandono. Vemos que las características del alumno invitan a esta incertidumbre: se trata de un alumno con una nota muy buena en matemáticas y suspenso en física en lo que a variables PAU se refiere. Tanto en primero como en segundo únicamente es capaz de aprobar una de cada cuatro convocatorias en que se presenta. La red le predice como un no abandono aunque, finalmente, el alumno abandona.

Alumnos que no abandonan y la red predice como abandonos:

	Alumno	1	2	3
Primero	Cred_matr	69.0000	72.0000	69.0000
	Tasaevaluados	0.5435	0.9375	0.8261
	Tasapresentados	0.5435	1.1250	1.3478
	Tasarendimiento	0.4348	0.5000	0.4565
	Tasaexito	0.8000	0.5333	0.5526
	NumConv	16.0000	19.0000	18.0000
	TasaNP	0.6250	0.4211	0.2222
	Tasa0	0.0625	0.2632	0.5000
	Tasa1	0.1875	0.1579	0.2222
	Tasa2	0.1250	0.1053	0.0556
	Tasa3	0	0.0526	0
	Tasa4	0	0	0
Segundo	Cred_matr	67.5000	69.0000	67.5000
	Tasaevaluados	0.7111	0.8478	0.5333
	Tasapresentados	0.8000	1.1957	0.9111
	Tasarendimiento	0.2000	0.5870	0.4222
	Tasaexito	0.2813	0.6923	0.7917
	NumConv	20.0000	20.0000	24.0000
	TasaNP	0.6500	0.4500	0.6667
	Tasa0	0.2500	0.3000	0.1667
	Tasa1	0.0500	0.2000	0.1667
	Tasa2	0.0500	0.0500	0
	Tasa3	0	0	0
	Tasa4	0	0	0
PAU	PAU_1	4.0000	7.4000	5.3000
	R_PAU_1	0.9016	0.1406	0.6456
	PAU_2	7.6000	7.2000	6.0000
	R_PAU_2	0.0791	0.1735	0.3313
	PAU_T	7.0000	6.9000	6.8000
	R_PAU_T	0.3371	0.3663	0.3509
	FISICA	8.8000	7.0000	5.5000
	R_FISICA	0.1159	0.3339	0.4957
	MAT	5.8000	7.0000	6.1000
	R_MAT	0.1855	0.4685	0.1087

Alumno	1	2	3
Abandona	0.7694	0.5702	0.7922
No abandona	0.1270	0.4416	0.2657

El primer alumno presenta unas calificaciones PAU normales, con rankings bastante buenos en ciertos apartados. En el primer año matriculado en la universidad no se presenta ni a la mitad de las asignaturas matriculadas y en el segundo curso únicamente aprueba en torno a un 10% de las asignaturas a las que se presenta. La red lo predice como abandono, cosa que parece bastante lógica. Finalmente el alumno no abandona y termina obteniendo el título.

El segundo alumno tiene una PAU similar al anterior. En primero se presenta mucho más que el primer alumno, aunque no consigue aprobar muchas más asignaturas. En segundo tampoco consigue tasas especialmente buenas.

El tercer alumno es el peor de los tres a nivel de PAU, pero el mejor en los dos primeros años de matrícula, aunque sus resultados tampoco son especialmente buenos. A pesar de que la red lo clasifica como abandono, el alumno sigue matriculado en su año teórico de egreso y al siguiente.

5.5. Análisis de la influencia de las variables.

A lo largo de todas las pruebas que aquí se desarrollan han sido empleados en los conjuntos de entrenamiento, validación y test los mismos alumnos para que la posible variabilidad por la selección aleatoria de los conjuntos no afecte a las conclusiones.

El enfoque que se adoptó inicialmente con el fin de ver la influencia de las diferentes variables fue ir sacando del conjunto de entrenamiento cada una de ellas para ver cómo afectaba este hecho a la bondad de los resultados. En vista de que en líneas generales, sacar una variable del conjunto de entrenamiento no afectaba significativamente a los resultados, se decidió realizar el análisis desde otro enfoque: comenzamos introduciendo cada una de las variables para ver qué capacidad tenía para realizar la clasificación por sí sola. Los resultados fueron claros: hay variables que representan muy bien los patrones de abandono/no abandono, tan bien -o mejor, incluso- como todas las variables en conjunto. En la tabla que se muestra en el siguiente apartado se recogen los resultados tanto de los entrenamientos del primer enfoque como de los entrenamientos del segundo enfoque.

5.5.1. Influencia de las variables PAU para Ingeniería Industrial.

Cada fila de esta tabla se corresponde con un entrenamiento realizado con un total de 357 ejemplos, donde se han utilizado el 70% para entrenamiento, el 15% para validación, y el 15% para test. El algoritmo de entrenamiento ha sido el gradiente escalado conjugado. Respecto a la nomenclatura, cabe destacar que las filas cuyo nombre es "Todas menos (aspecto)" se corresponden con entrenamientos realizados con todas las variables salvo las relacionadas con el aspecto mencionado. Es decir, el entrenamiento de "Todas menos matemáticas" se ha realizado con todas las variables salvo las dos correspondientes a matemáticas (notas y ranking). Por otro lado, las filas cuyo nombre es "Sólo (aspecto)" se corresponden con entrenamientos en los que las variables de entrada son las relacionadas (nota y ranking) con el aspecto citado.

La columna de "0" hace referencia al porcentaje de aciertos en "No abandonos" y la columna de "1" hace referencia al porcentaje de aciertos en "Abandonos".

INDUSTRIAL PAU			
	0	1	Total
Todas las variables	88.0	82.8	85.2
Todas menos matemáticas	92.0	89.7	90.7
Todas menos física	76.0	89.7	83.3
Todas menos PAU	80.0	86.2	83.3
Todas menos PAU1	88.0	89.7	88.9
Todas menos PAU2	80.0	86.2	83.3
Todas menos notas	84.0	86.2	85.2
Todas menos clasificación	88.0	89.7	88.9
Sólo matemáticas	56.0	79.3	68.5
Sólo física	76.0	79.3	77.8
Sólo PAU	72.0	86.2	79.6
Sólo PAU1	68.0	65.5	66.7
Sólo PAU2	76.0	89.7	83.3

Respecto a las conclusiones, vemos que al dejar PAU1 como única variable de entrada, los resultados empeoran considerablemente (en torno al 20%) tanto en la predicción de abandonos como de no abandonos respecto a considerar todas las variables de entrada. Emplear como única variable de entrada matemáticas supone empeorar solo un 2% las predicciones de abandono, aunque empeoran cerca de un 30% las predicciones de no abandonos. Es más, quitar la variable matemáticas del conjunto de entrada mejora los resultados significativamente, por lo que podemos concluir que esta variable es la que peor representa los patrones de abandono y no abandono.

Con física ocurre que empleada como única variable de entrada es capaz de predecir el 76% de los no abandonos y el 79.3% de los abandonos. Las variables PAU y PAU 2 andan en la misma línea, siendo capaces de predecir por sí solas en torno al 75% de los no abandonos y el 86.2% de los abandonos. Por ello, podemos concluir que estos aspectos relacionados con las calificaciones PAU poseen patrones vinculados con los patrones de abandono/no abandono y que la red es capaz de representar con un tanto por ciento de acierto relativamente bueno.

¿Existe relación entre Ingeniería Industrial e Ingeniería de Telecomunicaciones?

Observando los resultados de Ingeniería Industrial y los de Ingeniería de Telecomunicaciones [en el Anexo] se aprecia a simple vista la dificultad para encontrar similitudes entre ambas tablas. Por ejemplo, para alumnos de telecomunicaciones, la asignatura de matemáticas de la PAU lleva a clasificar la mayor parte de los alumnos como no abandonos, mientras que en el caso de industriales, la distribución se reparte más, siendo incluso superior el número de alumnos clasificados como abandonos (34 de 54) al número de alumnos clasificados como no abandonos (20 de 54). Sin embargo, tanto para telecomunicaciones como para industriales, vemos que, por si sola, esta variable no representa correctamente los patrones de abandono y no abandono y en industriales, además, confunde a la red, pues quitarla del conjunto de entrada supone mejorar los resultados.

PAU y PAU 2 no son especialmente representativas para telecomunicaciones pero sí para industriales, al igual que física.

En telecomunicaciones vemos que los ranking de los alumnos no influyen especialmente, pero sí las notas, ya que si quitamos estas últimas el porcentaje de acierto de los abandonos cae hasta el 41%. En industriales, ambas influyen de manera similar.

Se trata por tanto de patrones distintos en la relación variables de PAU con la tasa de abandono. Esto significa que el perfil PAU de un alumno que abandona telecomunicaciones es en general distinto al perfil PAU de un alumno que abandona industriales.

5.5.2. Influencia de las variables de primer curso para Ingeniería Industrial.

La siguiente tabla presenta el mismo aspecto que la tabla del apartado anterior y ha sido desarrollada mediante una metodología similar. Se ha empleado un conjunto de datos con 357 ejemplos (70% para entrenamiento, 15% para validación y 15% para test). Se ha realizado un entrenamiento con las 12 variables de primer curso (sin variables PAU) y se han ido sacando del conjunto Input cada una de ellas. Luego, en un segundo enfoque, se ha empleado cada una de las variables como única variable del conjunto Input.

INDUSTRIAL primero			
	0	1	Total
Todas variables	64.0	89.7	77.8
Todas menos creditosmatriculados	64.0	89.7	77.8
Todas menos numconv	64.0	93.1	79.6
Todas menos tasa0	72.0	89.7	81.5
Todas menos tasa1	60.0	93.1	77.8
Todas menos tasa2	72.0	86.2	79.6
Todas menos tasa3	76.0	93.1	85.2
Todas menos tasa4	68.0	93.1	81.5
Todas menos tasaevaluados	64.0	93.1	79.6
Todas menos tasaexito	68.0	89.7	79.6
Todas menos tasaNP	68.0	86.2	77.8
Todas menos tasapresentados	64.0	89.7	77.8
Todas menos tasarendimiento	60.0	89.7	75.9
Sólo creditosmatriculados	52.0	93.1	74.1
Sólo numconv	76.0	82.8	79.6
Sólo tasa0	64.0	51.7	57.4
Sólo tasa1	40.0	96.6	70.4
Sólo tasa2	64.0	89.7	77.8
Sólo tasa3	40.0	93.1	68.5
Sólo tasa4	16.0	96.6	59.3
Sólo tasaevaluados	72.0	89.7	81.5
Sólo tasaexito	64.0	82.8	74.1
Sólo tasaNP	44.0	96.6	72.2
Sólo tasapresentados	92.0	48.3	68.5
Sólo tasarendimiento	64.0	93.1	79.6

Vemos que tasa de presentados tiene poca influencia para la predicción de abandonos ya que quitarla del conjunto inputs no varía el porcentaje de éxito y dejarla como única variable de entrada ayuda a predecir muy bien los que no abandonan pero a costa de una caída muy grande del porcentaje de acierto asociados a los abandonos. Ocurre lo mismo con la tasa de convocatorias que el alumno suspende: por sí sola, es incapaz de obtener buenas tasas de abandono porque hay muchos alumnos que el primer año suspenden muchas asignaturas porque les cuestas adaptarse y no por ello abandonan.

El número de convocatorias a las que el alumno podría presentarse es un buen indicador de la tasa de abandono ya que por sí sola es capaz de predecir el 76% de los no abandonos y el 82.8% de los abandonos, mejorando los registros obtenidos con todas las variables. Esto hemos de tenerlo en cuenta ya que es lo que explica que al quitar otras variables, no se alteren en exceso los resultados. Mientras esta variable permanezca en el conjunto de entrada, los resultados seguirán siendo buenos. Al quitarla del conjunto inputs no se producen cambios para la predicción de los no abandonos y mejora ligeramente la predicción de los abandonos. Ello nos hace pensar que hay otras variables como esta, que son capaces por si mismas de alcanzar grandes tasas de acierto. Un ejemplo de ello es la variable de tasa de éxito, que por sí sola, acierta el 64% de los no abandonos y el 82.8% de los abandonos.

La tasa de convocatorias en las que el alumno obtiene sobresaliente, por sí sola, ayuda a clasificar al 93.1% de los abandonos pero sólo clasifica bien el 40% de los no abandonos. Ello puede ser así porque la red está entrenada de manera que ve muy claro que los alumnos que abandonan nunca obtienen sobresaliente, pero no es capaz de asimilar que muchos alumnos sin sobresaliente no abandonan. Entonces a la mayoría de los alumnos sin sobresaliente o con poca tasa de sobresaliente los clasifica como abandonos (42 de 54) mientras que sólo clasifica como no abandono a los que tienen una tasa alta de sobresaliente (12 de 54). Por ello, quitar la tasa de abandono mejora la predicción de los alumnos que no abandonan y no influye decisivamente en la predicción de los alumnos que abandonan (incluso mejora su predicción al quitarla como variable de entrada). Algo similar o incluso un poco más radical sucede con la tasa de convocatorias con matrículas de honor. Utilizada como única variable de entrada clasifica como abandonos a 49 de 54 alumnos y solo a 5 de 54 alumnos los clasifica como no abandono. Otro ejemplo de variable con estas características es la de tasa de convocatorias a las que el alumno no se presenta. La red clasifica a 42 de 54 casos como abandono y únicamente a 12 de 54 como no abandono. Las variables de número de créditos matriculados y tasa de convocatorias que el alumno aprueba poseen las mismas características que éstas.

La tasa de convocatorias en que el alumno tiene calificación de notable tiene por sí sola, como input, la misma capacidad de predecir abandonos y no abandonos que la red que usa todas las variables de primero como inputs. Ello explica que al quitar otras variables, la red siga teniendo tasas de acierto elevadas. La presencia de esta variable supone que la red tenga esas tasas de acierto. Las mismas características que esta variable presenta la variable tasa de evaluados. También sigue esta tendencia la variable de tasa de

rendimiento que, como única variable de entrada, mejora incluso los resultados obtenidos con todas las variables, con una tasa de acierto para no abandonos del 64% y una tasa de acierto para abandonos del 93.1%. Aun así, en estos casos también se observa la tendencia de la red a clasificar la gran parte de los ejemplos de test como abandonos. En el caso de la tasa de rendimiento clasifica como no abandono a 18 de 54 casos y como abandono a 36 de 54 casos, cuando la realidad está más repartida: de los 54 ejemplos, 25 no abandonan y 29 sí.

En general, considerando que es más interesante una red con buena tasa de acierto en abandonos, que una red con buena tasa de acierto en no abandonos podemos clasificar las variables dependiendo de su influencia como malas, buenas y muy buenas según si explican, mal, bien o muy bien los abandonos/no abandonos:

Malas: Créditos matriculados, tasaNP, tasa1, tasa 3, tasa 4, tasa0 y tasa presentados.

Buenas: Numconv, tasa2, tasa éxito, tasa rendimiento.

Muy buenas: tasa evaluados.

¿Existe relación entre Ingeniería Industrial e Ingeniería de Telecomunicaciones [Anexo]?

La primera conclusión que extraemos es que las tasas de acierto en telecomunicaciones son mayores que las tasas de acierto en industriales. Por tanto, las similitudes entre las dos titulaciones que obtenemos aquí son relativas y cuando se indica que una variable se adecúa bien (regular, mal,…) a los patrones de abanono/no abandono, en realidad lo que indicamos es que se adecúa bien (regular, mal) respecto a las demás variables de esa misma titulación.

Vemos que tanto para industriales como para telecomunicaciones variables con poca capacidad de predecir abandonos (con poca influencia sobre la predicción) son: créditos matriculados, tasa0, tasa3 y tasa4.

Variables que en ambas titulaciones son buenas o muy buenas son las siguientes: numconv, tasa_éxito, tasa_rendimiento.

Las variables tasa_presentados y tasaNP predicen mal para industriales y regular para telecomunicaciones.

Cabe destacar los casos de las variables tasa1 y tasa2: Mientras que para industriales tasa1 predice mal y tasa2 predice bien, para telecomunicaciones tasa 1 predice bien y tasa2 predice mal.

Finalmente encontramos la variable tasa_evaluados que para telecomunicaciones no es especialmente buena pero que en industriales ayuda a predecir muy bien los casos de abandono.

6. Capítulo VI: Conclusiones generales.

En este trabajo se han presentado diversos análisis ante distintas problemáticas que en este capítulo trataremos de sintetizar.

Se ha pretendido, por un lado, ver la capacidad de las Redes Neuronales Artificiales para predecir la nota de egreso de los alumnos de Ingeniería Industrial y de Ingeniería de Telecomunicaciones a partir de una serie de variables relacionadas con la PAU y con las asignaturas más complejas de la titulación. Se ha propuesto como alternativa a las Redes Neuronales, una regresión lineal múltiple, con muy buenos resultados. Por otro lado, se ha estudiado la capacidad de las Redes Neuronales Artificiales para clasificar a un alumno como potencial abandono (o no abandono) en función de variables PAU y de variables que explican el rendimiento académico del alumno durante el primer y el segundo año de matrícula.

6.1. Predicción de la nota de egreso.

Respecto a la predicción de la nota de egreso se han presentado tres metodologías: dos basadas en redes neuronales artificiales (clasificación y ajuste de funciones) y otra basada en regresión lineal múltiple.

Los resultados obtenidos mediante Redes Neuronales para clasificación han sido en general malos. Aun incluyendo notas de las asignaturas más complejas de la titulación la red tenía grandes dificultades para establecer correctamente la frontera de decisión. Ello parece lógico si retomamos el análisis realizado en el capítulo correspondiente: dos alumnos con notas 1.51 y 1.99 habrán de ser clasificados por la red dentro del mismo grupo para que exista acierto en ambos casos, mientras que dos alumnos con notas 1.99 y 2.01 habrán de ser clasificados en grupos distintos. En el primer caso la diferencia de nota media es casi de medio punto, mientras que en el segundo caso la diferencia es de apenas dos centésimas. Especialmente malos han sido los resultados de clasificación para alumnos con notas entre 2 y 4 puntos, pues la mayoría del conjunto de entrenamiento tiene notas medias que se hallan entre 1 y 2 puntos.

También parece importante destacar la enorme variabilidad que introduce la elección aleatoria de los tres conjuntos (entrenamiento, validación y test). Los conjuntos de datos que se han manejados han sido relativamente pequeños, en el sentido de que para testear la red entrenada poseíamos para Telecomunicaciones unos 25 alumnos y para industriales aún menos (un 15% del total). Sabida es la existencia de casos anómalos o cambiantes en estos entornos: alumnos que habiendo hecho una PAU muy mala, entran a una titulación donde se imparten conocimientos que les motivan y sus notas son buenas, y el caso inverso, representado por alumnos brillantes hasta la PAU y que luego tienen dificultades en la titulación. Ocurre entonces que si estos alumnos caen en el conjunto de entrenamiento se generarán variaciones en los pesos que harán perder generalidad a la red. Algo parecido ocurre si caen en el conjunto de validación: puede que el entrenamiento se detenga sin que la generalidad de la red sea todo lo buena que

pudiese. Si estos ejemplos caen en el conjunto de test puede ocurrir que aunque la generalidad de la red sea buena, los resultados para test sean malos porque la red no sea capaz de clasificar estos casos anómalos.

Decidir abordar el problema desde una perspectiva de predicción en lugar de clasificación es un acierto por varios motivos. Por un lado, no alteramos la naturaleza del "Target" haciendo discreta una variable continua, cosa que si hacemos para clasificación. La interpretación, por tanto, es más directa en el sentido de que podemos ver directamente en cuantas décimas se equivoca la red para cada ejemplo.

Los resultados obtenidos son, en general, bastante buenos, observándose mejoras al introducir ciertas asignaturas de la titulación, tanto para Ingeniería Industrial como para Ingeniería de Telecomunicaciones.

El empleo de regresión lineal múltiple ha resultado bastante útil pues los resultados obtenidos han sido muy buenos, tanto como los obtenidos con redes neuronales. Ello corrobora en cierta medida algo que inicialmente sólo era una intuición: existe una relación lineal entre las variables PAU y la nota de egreso de la carrera y entre las asignaturas complejas de la titulación y la nota de egreso, en algunos casos positiva y en otros, sorprendentemente, negativa, lo que dificulta la interpretación del modelo.

Por otro lado, un aspecto reseñable es que en regresión lineal múltiple los resultados iban mejorando conforme se introducían variables de asignaturas de la carrera mientras que con redes neuronales no era tan fácil observar esta tendencia. Esto apoya una de las características que mencionamos al explicar los métodos para ver la influencia de las variables de entrada en las variables de salida de la red: la gran dificultad existente para interpretar los resultados que arroja una red neuronal. Muchas veces se ha visto esta herramienta como una especie de "caja negra" que da buenos resultados, pero en la que es difícil analizar, por ejemplo, qué significado tienen los pesos entre neuronas.

6.2. Clasificación de abandonos.

Respecto a la clasificación de alumnos como potenciales abandonos (o no abandonos) es preciso destacar que los resultados obtenidos para clasificar los abandonos de segundo año han sido bastante malos, mientras que los resultados obtenidos para clasificar el abandono oficial han sido, en líneas generales, bastante buenos.

La principal explicación que encontramos para los malos resultados obtenidos en el abandono de segundo año es la influencia de variables no contempladas en el conjunto de entrada. Este tipo de abandonos suele estar relacionado con alumnos que descubren que no les gusta la titulación que han elegido más que con las calificaciones que han obtenido en la PAU. Pueden haber obtenido notas brillantes en selectividad y equivocarse en la elección de la carrera, o no. Ello implica la necesidad de emplear variables de las que no disponemos, relacionadas sobre todo con las motivaciones de

los alumnos y que se podrían obtener mediante encuestas. El perfil de este tipo de alumnos también puede basarse en lo mal que ha ido durante el primer año matriculado; y al segundo año, decide no matricularse. Ello explica la mejora que experimentan los resultados al emplear variables del primer año de matrícula. Sin embargo, los resultados tampoco son excesivamente buenos, por la misma razón argumentada antes: la influencia de variables no contempladas en los conjuntos. Hay alumnos con notas muy malas el primer año de matrícula que deciden continuar y se matriculan el segundo año y otros que deciden abandonar. También pueden afectar factores económicos de manera que entre dos alumnos con las mismas asignaturas suspensas el primer año, uno pueda permitirse matricularse un segundo año y el otro no.

Respecto al abandono oficial, vemos que los resultados obtenidos son bastante buenos. Mejoran significativamente al incluir variables de primer año. Especialmente reseñable es que hay variables del primer año de matrícula que, por sí solas (utilizadas como única variable de entrada), son capaces de clasificar muy bien los abandonos, tanto para el caso de Ingeniero Industrial como para el caso de Ingeniero en Telecomunicaciones, tal y como se recoge en el correspondiente apartado del capítulo V. Son variables como número de convocatorias, tasa de éxito y tasa de rendimiento.

6.3. Líneas de investigación. Actualidad.

Este proyecto constituye una primera aproximación a una problemática de actualidad en la sociedad en que vivimos: el abandono universitario. Se afronta principalmente desde la perspectiva de las Redes Neuronales Artificiales pero hay otros métodos que también han sido empleados al respecto en otros estudios.

Cabe destacar el trabajo recogido en (Oficina de Cooperación Universitaria) [6], de la Oficina de Cooperación Universitaria, que alberga dos ejemplos de análisis que, conceptualmente, guardan relación con este proyecto y, además, sintetizan bastante bien las alternativas a la hora de afrontar este tipo de problemas.

En el primero tratan de predecir la nota de un alumno en el examen de una determinada asignatura de la titulación a partir de distintas variables relacionadas con conocimientos previos, experiencia laboral en el sector y trabajo hasta la realización del examen. Concluyen que la técnica más apropiada es la denominada M5’ (la comparan con redes neuronales artificiales, regresión lineal, regresión lineal localmente ponderada y máquinas de vectores soporte), que construye un árbol de regresión a partir de las variables de entrada. Se destaca en esta referencia que los resultados de este método son “más interpretables” que los obtenidos mediante otros métodos. En este sentido puede resultar interesante la aplicación de esta técnica para la predicción de la nota de egreso de los alumnos (Capítulo IV). Se podría ver si es capaz de mejorar los resultados

obtenidos mediante redes neuronales y qué conclusiones se pueden desprender de ellos.

En el segundo análisis pretenden clasificar a un alumno como abandono o no abandono. Recogen variables de distintos ámbitos: calificaciones (7 variables), socioeconómico (25 variables), cuestionarios (45 variables). Mediante el empleo de algoritmos de selección de atributos reducen el número de variables a 15. Emplean distintos métodos de clasificación y los comparan, destacando, por la calidad de los resultados y la posibilidad de interpretación, distintas variantes de los métodos ICRM. Estos métodos podrían también emplearse para comparar los resultados con los que se han obtenido en el capítulo V del presente proyecto.

Por otro lado, es importante destacar que los algoritmos de *Machine Learning*, entre los que se encuentran las redes neuronales artificiales, han experimentado un gran auge en el contexto de un concepto muy de moda en la actualidad: el Big Data. Este concepto se explica mediante otros tres:

- Volumen: En la actualidad los volúmenes de información manejados (redes sociales, Apps, sensores, información generada por máquinas, …) son muy grandes. Estamos pasando de hablar de Gigabytes o Terabytes a Petabytes, Exabytes o Zettabytes.
- Variedad: Ahora la información no sólo se encuentra en bases de datos, sino que hay grandes cantidades de información desestructurada, cuya ordenación y estructuración requiere de nuevas técnicas.
- Velocidad: La velocidad a la que se genera la información hace imposible gestionarla con sistemas de base de datos convencionales. Se requieren sistemas que procesen más información a mayor velocidad.

Los desarrollos relacionados con el concepto del Big Data están especialmente orientados a la toma de decisiones en base a datos empíricos y experiencias, que era precisamente de lo que hablábamos en la introducción de este trabajo. En este sentido es importante tener presentes los avances que surgen porque para predicción de rendimiento académico o clasificación de abandono se puede llegar a trabajar con conjuntos de datos muy grandes y que requieran de una estricta ordenación y pre procesado para poder sacar conclusiones mediante las técnicas mencionadas: regresión, máquinas de vectores soporte, redes neuronales artificiales, entre otras.

7. Capítulo VII: Bibliografía.

[1] Matich, D. J. (2001). *Tema: Redes Neuronales: Conceptos Básicos y Aplicaciones.* Rosario: Universidad Tecnológica Nacional. Enlace: http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r101038.PDF

[2] Hudson Beale, M., T. Hagan, M., & B. Demuth, H. (2010). *Neural Network Toolbox 7. User's Guide.*

[3] Yu, H., & M. Wilamowski, B. (2010). *Levenberg-Marquardt Training.* Auburn University. Enlace: http://www.eng.auburn.edu/~wilambm/pap/2011/K10149_C012.pdf

[4] Fodslette Moller, M. (1993). *A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning.* Artículo en *Neural Networks,* Vol 6, pp. 525-533. Enlace: http://sci2s.ugr.es/keel/pdf/algorithm/articulo/moller1990.pdf

[5] Montaño Moreno, J. J. (2002). *Redes Neuronales Artificiales aplicadas al Análisis de Datos.* Tesis Doctoral. Universitat de Les Illes Balears. Facultad de Psicología. Enlace: http://www.tesisenred.net/bitstream/handle/10803/9441/tjjmm1de1.pdf?sequence=1

[6] Oficina de Cooperación Universitaria. (2013). Libro blanco. Inteligencia Institucional en Universidades. Enlace: http://www.ocu.es/portal/page/portal/inicio/documentos/OCU_LB_I2_013_Digital.pdf

[7] Curso online. *"Introducción a los algoritmos de regresión logística y redes neuronales"*. Coursera, Machine Learning, Stanford. Enlace: https://www.coursera.org/course/ml

[8] UCI. Machine Learning Repository. *Center for Machine Learning and Intelligent Systems*. Enlace: http://archive.ics.uci.edu/ml/index.html

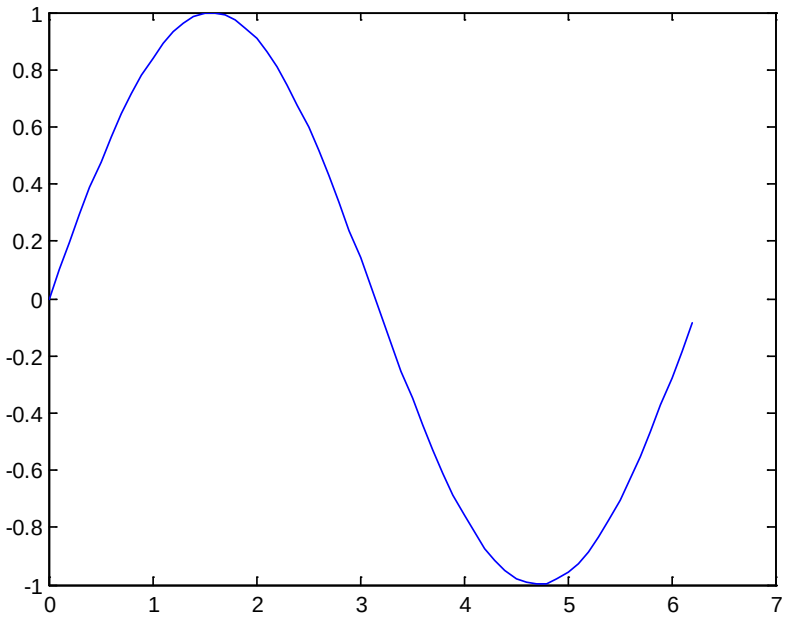
8. Capítulo VIII: Anexos.

8.1. Ejemplos de aplicación en Matlab.

8.1.1. Ejemplos para ajuste de funciones.

8.1.1.1. Ajuste de la función seno.

Para obtener los datos que ajusten esta función se ha creado un vector X que recoge valores entre 0 y 2π separados por una distancia de 0.1 (en total, 63 valores). A continuación se ha generado un vector Y que recoge los valores asociados a $\sin(X)$. Representando gráficamente Y frente a X se obtiene la función seno:



Nuestro objetivo es construir una red neuronal que sea capaz de ajustar la función seno, es decir, que para uno valor de cualquiera nos diga el valor de su seno pero sin ejecutar la función seno, sino realizando la propagación hacia delante de la red neuronal.

El entrenamiento se llevará a cabo en las siguientes condiciones:

Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	63
Variables consideradas	1
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	2/4

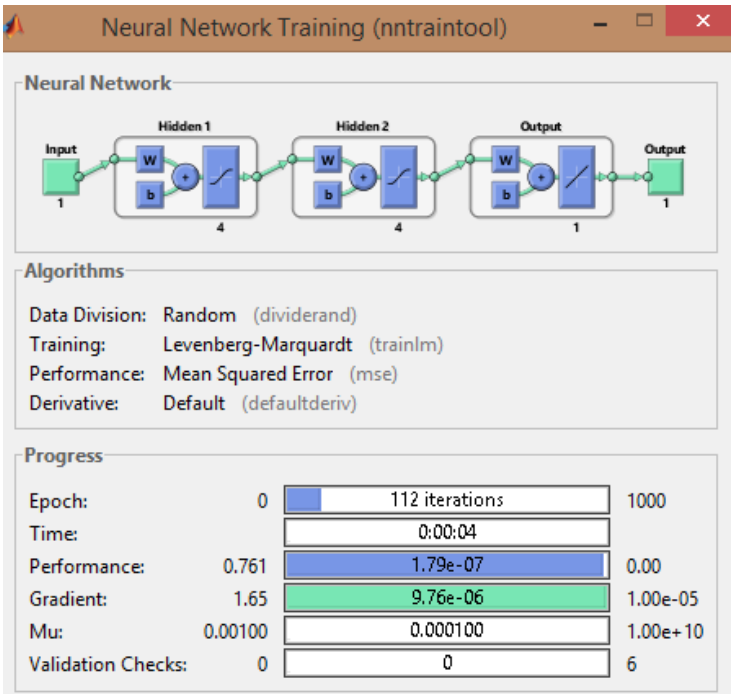
La función de entrenamiento que emplearemos es la que implementa el algoritmo de Levenberg-Marquardt. Habrá 63 ejemplos de entrenamiento y la única variable que

influye en el output de la red es X. Por tanto los vectores X e Y tienen ambos dimensión [1x63].

Los comandos necesarios para crear y entrenar la red, de acuerdo con lo ya explicado son los siguientes:

```
hiddenLayerSize = [4 4];

net = fitnet(hiddenLayerSize);

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

[net,tr] = train(net,X,Y);
```

El proceso de entrenamiento lo podemos ver en una ventana como la siguiente, en la que aparece representado el estado final del entrenamiento:

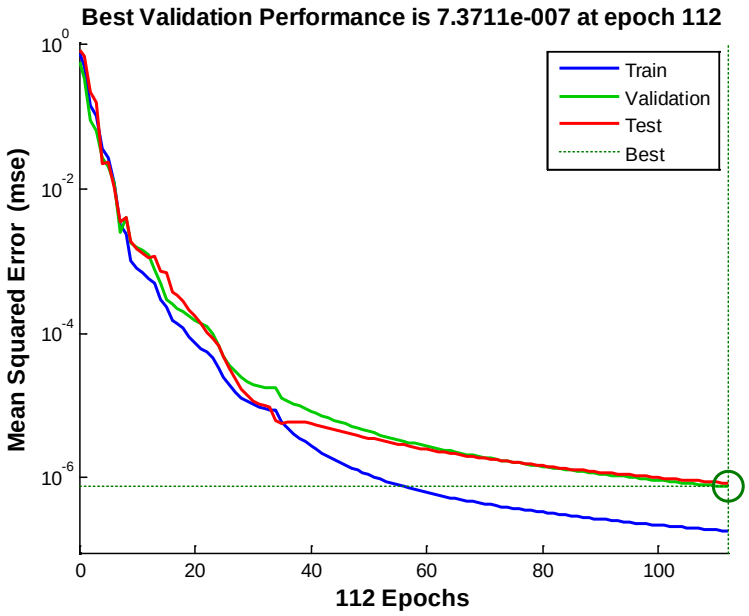


Vemos que la arquitectura de la red está formada por dos capas ocultas de cuatro neuronas cada una y una capa de salida con una función de activación lineal.

El entrenamiento ha requerido 112 iteraciones, que se han llevado a cabo durante 4 segundos. El valor de MSE alcanzado es de 1.79e-07, y el valor del gradiente final ha sido de 9.76e-06. Es preciso destacar que el proceso de entrenamiento se ha detenido cuando el gradiente ha alcanzado un valor inferior al umbral marcado (1.00e-05).

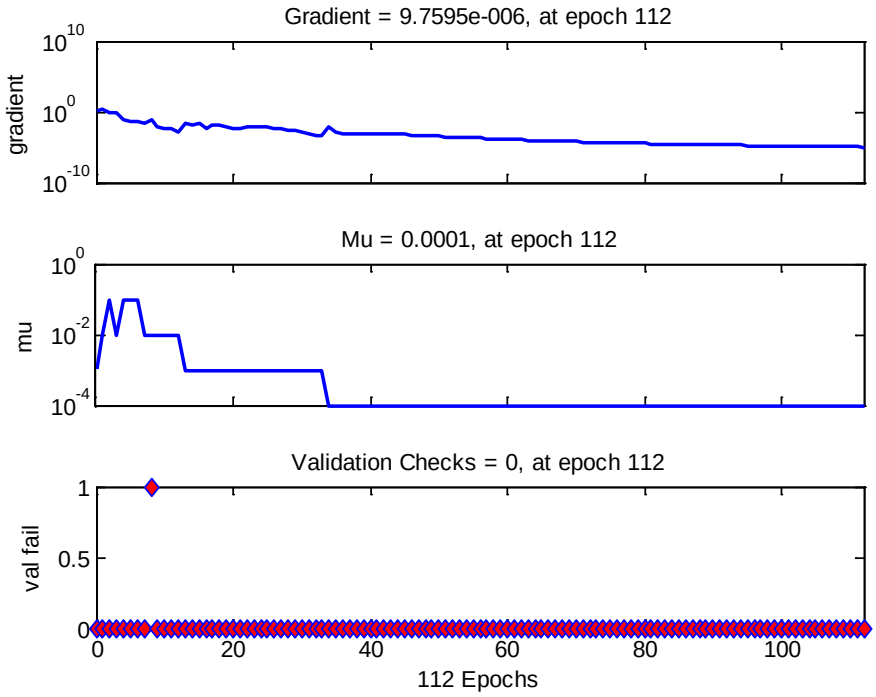
Respecto a las soluciones, encontramos los siguientes resultados:

a) Evolución del MSE:



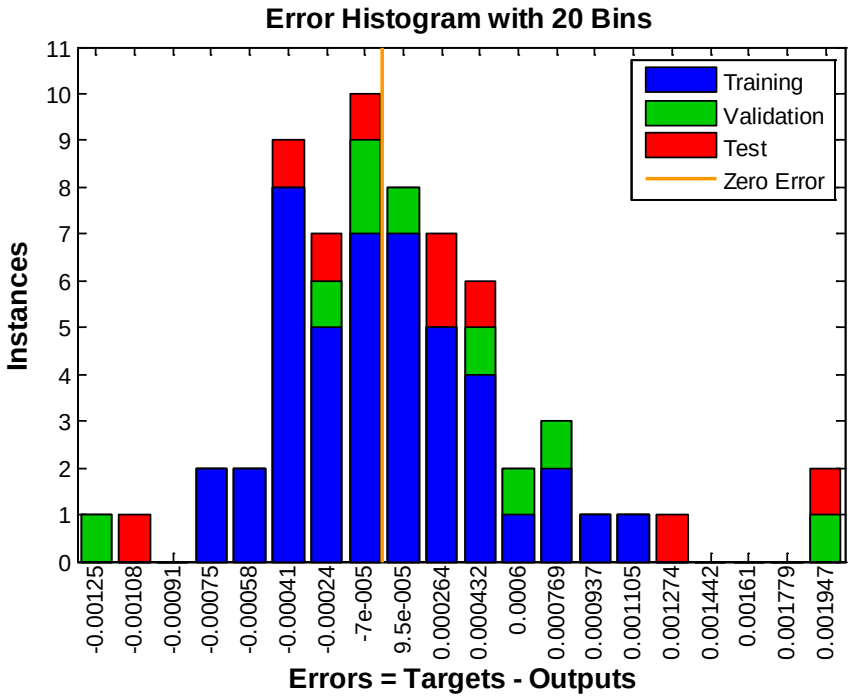
Se aprecia que conforme avanza el número de épocas, el valor del MSE disminuye y parece que aumentado el número de épocas el valor del MSE podría disminuir aún un poco más, aunque la tendencia para los conjuntos de test y validación se halla cercana a la estabilización en un valor mínimo constante.

b) Estado del entrenamiento.



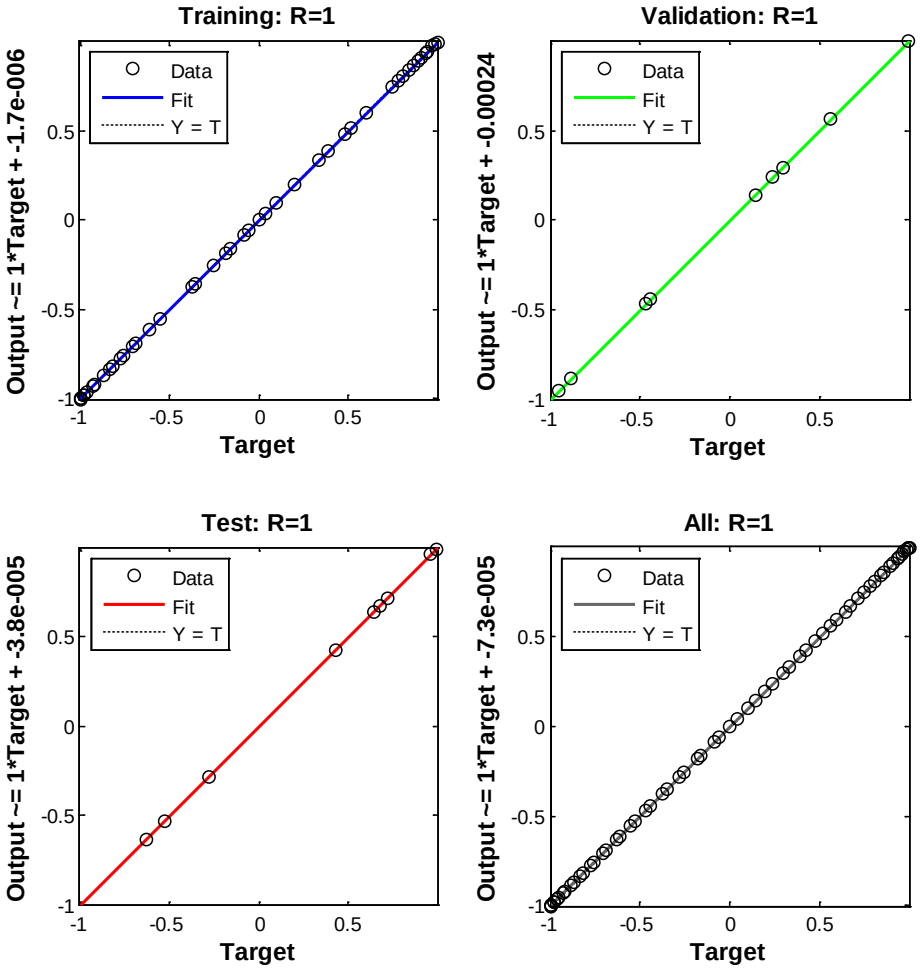
En la primera de las tres gráficas que se muestran en la imagen vemos la evolución del valor del gradiente a lo largo de las distintas épocas de entrenamiento. Disminuye sucesivamente, al igual que el valor mu. En la última gráfica vemos las comprobaciones de validación. Tras cada época el conjunto de validación es implementado hacia adelante en la red propia de la época y el error es calculado. Si el error aumenta respecto a la iteración anterior, se contabiliza una comprobación de validación. Si el error disminuye respecto a la iteración anterior, el contador de comprobación se pone a cero. Cuando se alcanza el valor fijado de comprobaciones de validación (6 en este caso), el entrenamiento se detiene. Como podemos apreciar, durante este proceso de entrenamiento sólo se ha producido una vez un aumento del error respecto al error cometido en la iteración precedente.

c) Histograma del error.

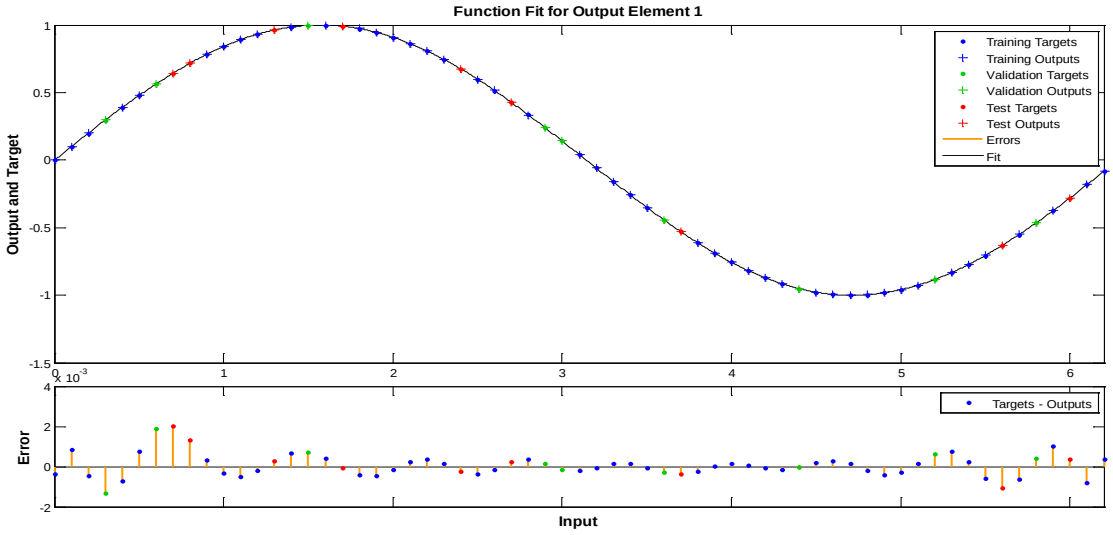


Este histograma permite ver los errores cometidos por la red para cada ejemplo de cada uno de los conjuntos de datos (entrenamiento, validación y test). Para el conjunto que nos interesa, que es el de test, para ningún ejemplo se supera un valor para el error de 0.002, lo que significa que el ajuste de la red es muy bueno.

d) Ajustes lineales de los datos.



e) Representación visual de los errores.



8.1.1.2. Ajuste de la función $z = e^{-(x^2+y^2)}$.

En primer lugar vamos a representar la función para ver su apariencia. Para ello, lo primero que hemos de hacer es generar un mayado:

[x,y]=meshgrid(-2:.25:2);

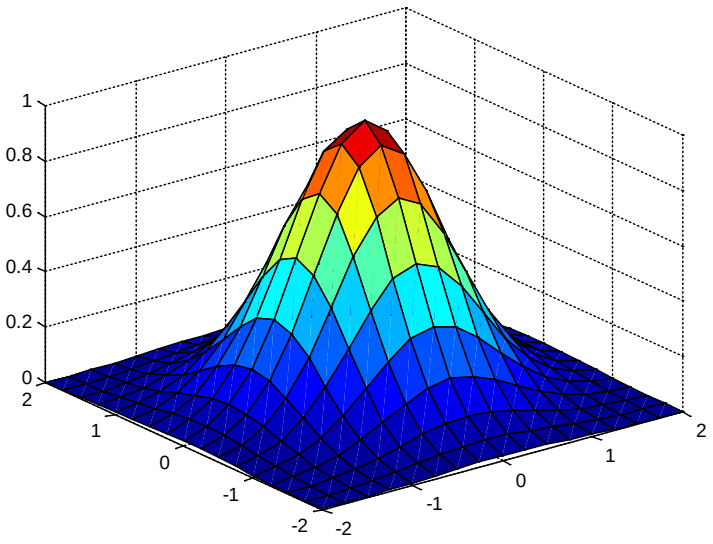
A continuación obtenemos los valores de z asociados a esos puntos:

z=exp(-x.^2-y.^2);

Finalmente, representamos mediante:

surf(x,y,z);

Y obtenemos:



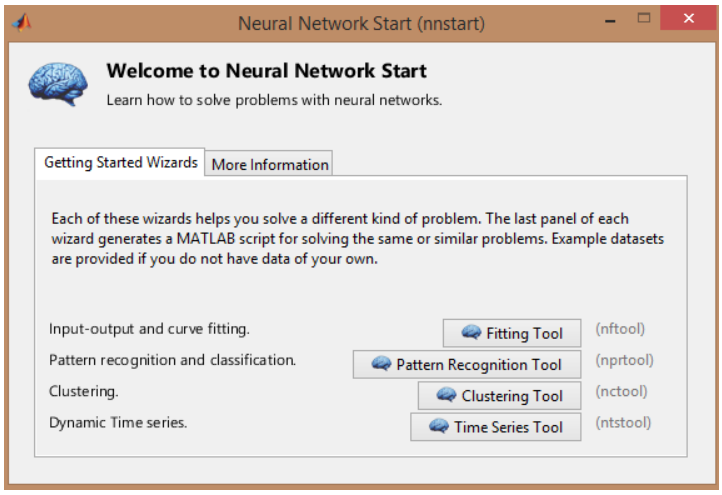
En la gráfica se pueden ver los distintos puntos empleados para hacer la representación gráfica y las líneas rectas que los unen. Dichos puntos nos servirán de entrenamiento, validación y testeo en nuestra red neuronal.

Para crear la red neuronal creamos nuestra matriz de inputs, formada por dos variables x_1, y_1, con 17 ejemplos cada una situados entre -2 y +2 y separados por 0.25. Creamos también la matriz targets z_1. El entrenamiento, por tanto, se llevará a cabo en las siguientes condiciones:

El entrenamiento se llevará a cabo en las siguientes condiciones:

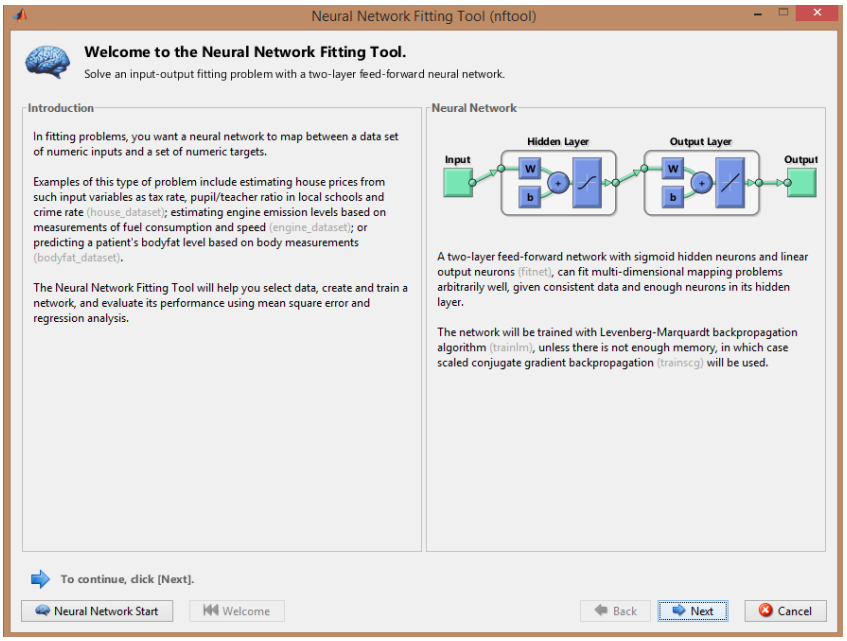
Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	17
Variables consideradas	2
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/5

Para entrenar la red, usaremos para este ejemplo el toolbox de Matlab que se inicia mediante la introducción del comando nnstart.

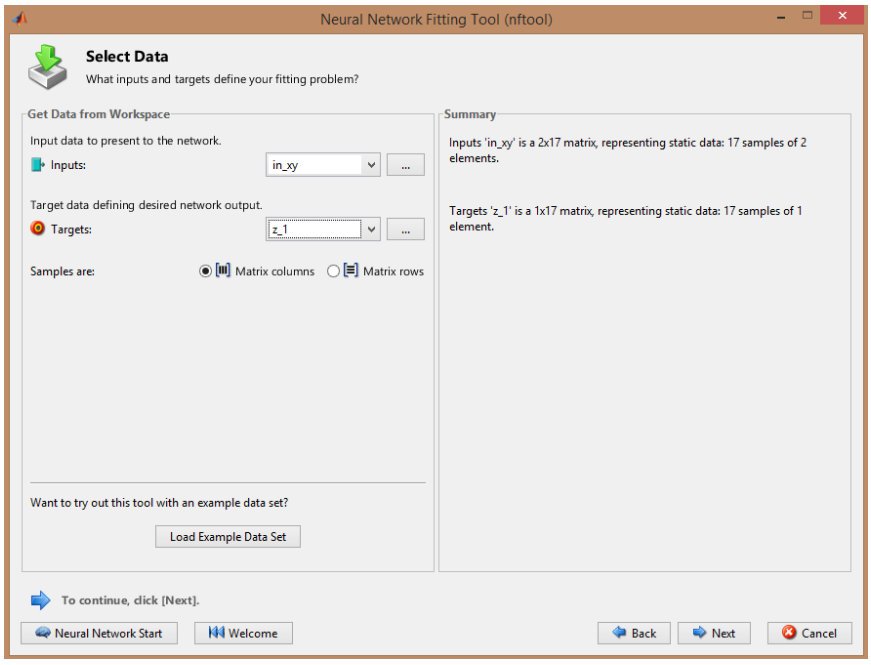


Esta ventana nos presenta la posibilidad de elegir entre cuatro posibles itinerarios. La aplicación que requiere la resolución de este ejemplo es “Fitting Tool”.

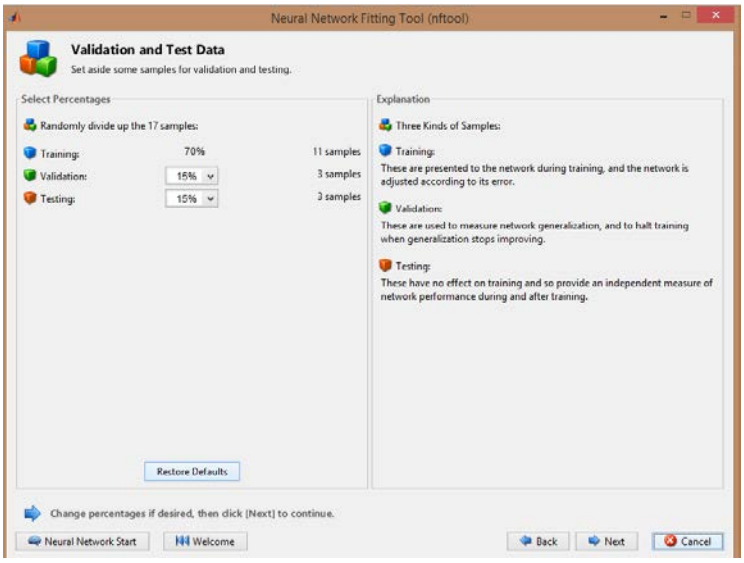
En la ventana que se abre a continuación se nos ofrece información por defecto asociada a esta aplicación como la arquitectura de la red y el algoritmo de entrenamiento que se va a utilizar (Levenberg-Marquardt):



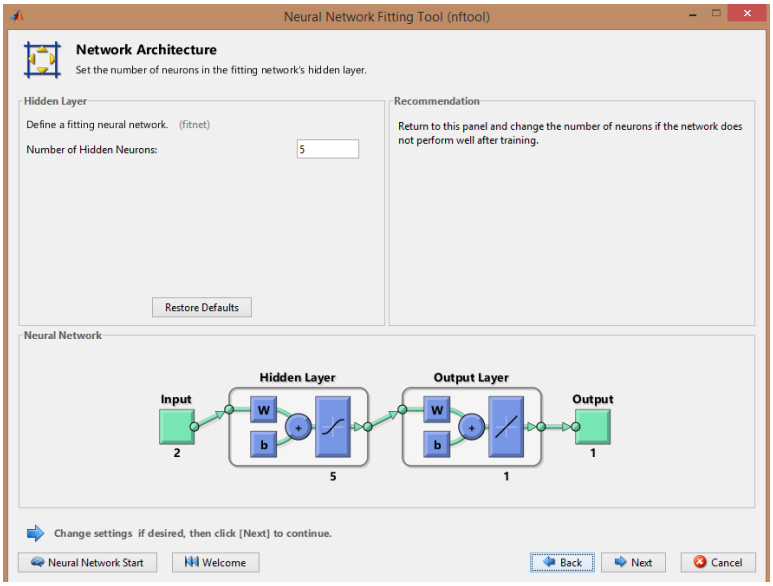
Ahora seleccionamos las matrices necesarias para poder entrenar la red (inputs y targets), que habrán de estar definidas en el espacio de trabajo cargado.



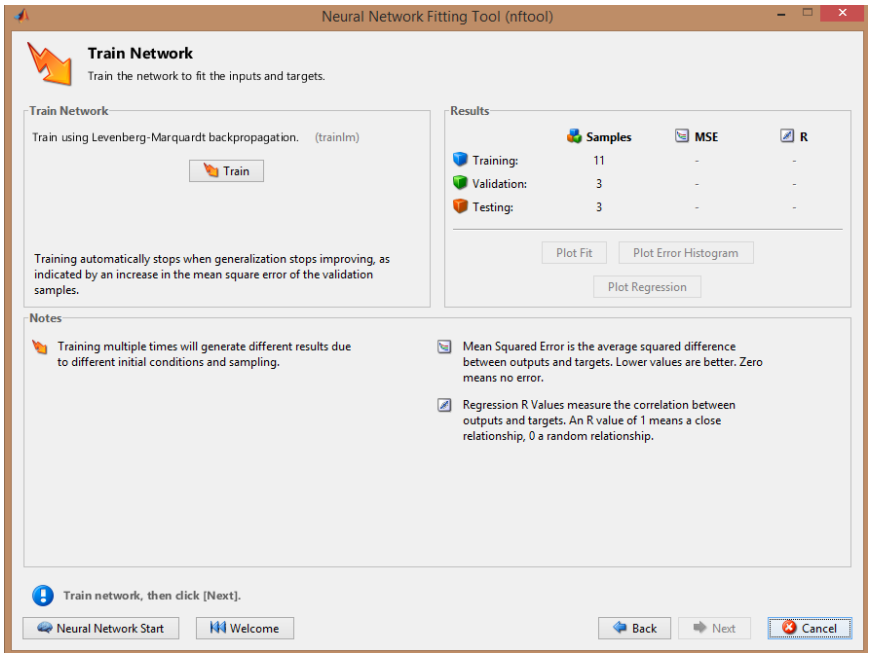
Ahora hemos de elegir la proporción de datos que dedicamos a entrenamiento, la proporción que dedicamos a validación y la proporción que dedicamos a test. Para este ejemplo, se ha decidido usar un 70% de los datos para entrenar la red, un 15% para validación, y un 15% para comprobación final:



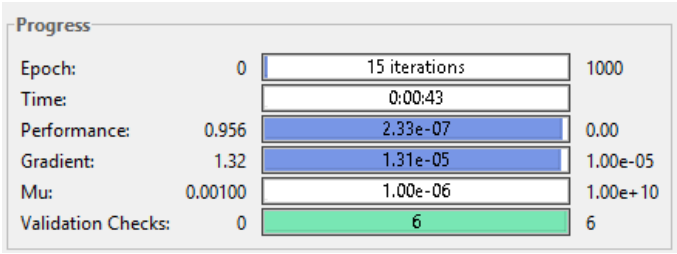
En la siguiente pantalla se nos pide que introduzcamos el número de neuronas de la capa oculta, y nos muestra un esquema de la arquitectura de la red:



Finalmente se nos muestra una ventana de resumen donde ya estamos en disposición de entrenar la red:



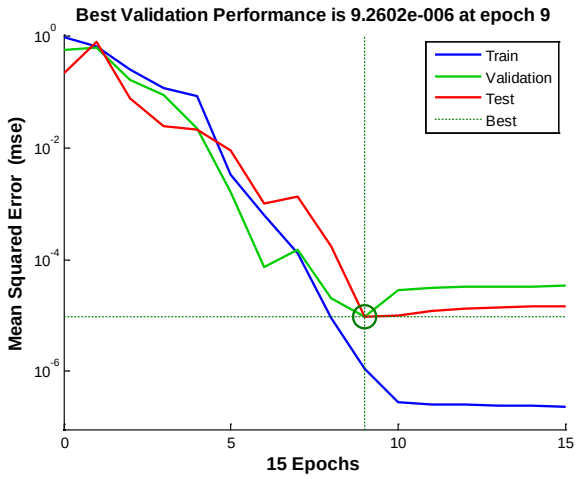
El proceso de entrenamiento lo podemos ver en una ventana como la siguiente, en la que aparece representado el estado final del entrenamiento:



El entrenamiento ha requerido 15 iteraciones, que se han llevado a cabo durante 43 segundos. El valor de MSE alcanzado es de 2.33e-07, y el valor del gradiente final ha sido de 1.31e-05. Es preciso destacar que el proceso de entrenamiento se ha detenido al alcanzarse las seis comprobaciones de validación, para evitar el problema del sobre ajuste y la pérdida de generalidad.

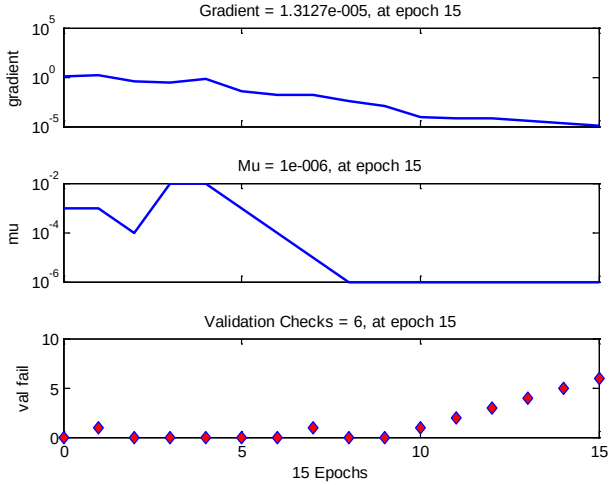
Respecto a las soluciones, encontramos los siguientes resultados:

a) Evolución del MSE:

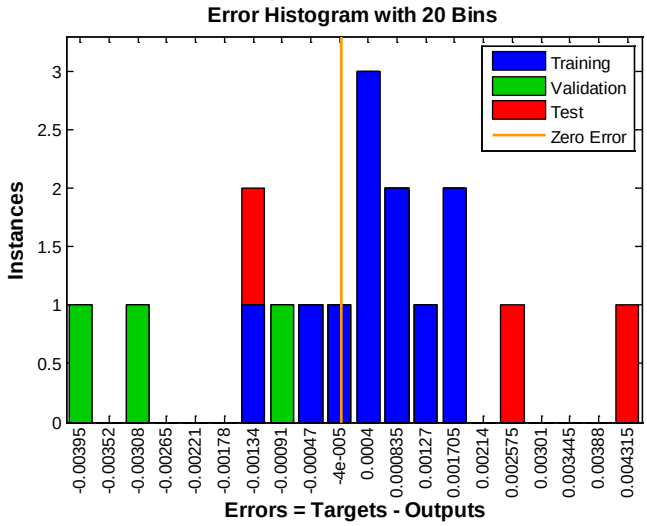


Se aprecia que conforme avanza el número de épocas, el valor del MSE disminuye y, a priori parece que aunque aumentemos el número de épocas, el valor del MSE va a permanecer constante o incluso puede aumentar un poco.

b) Estado del entrenamiento.

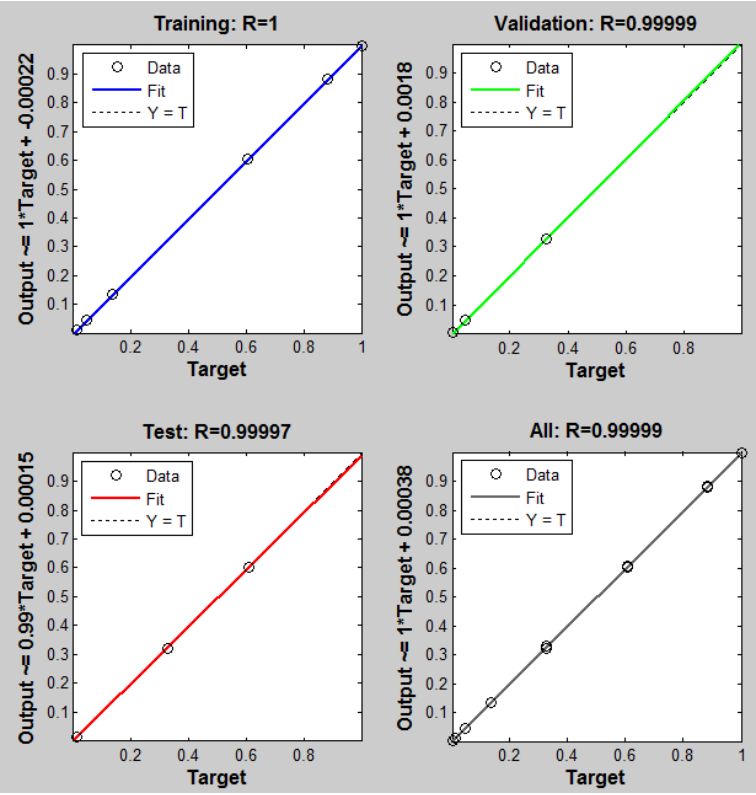


c) Histograma del error.



Este histograma permite ver los errores cometidos por la red para cada ejemplo de cada uno de los conjuntos de datos (entrenamiento, validación y test). Para el conjunto que nos interesa, que es el de test, vemos que se cometen los errores más grandes, aunque en ningún caso superan el valor de 0.005. Una de las principales conclusiones que extraemos es que con un conjunto de entrenamiento relativamente pequeño (sólo 11 ejemplos), hemos conseguido aproximar muy bien la centésima de la función.

d) Ajustes lineales de los datos.



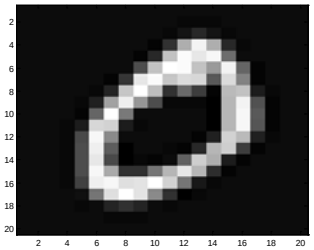
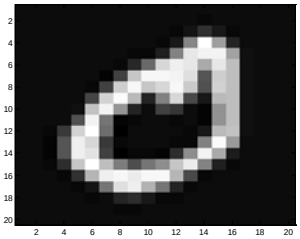
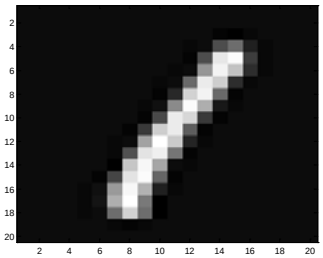
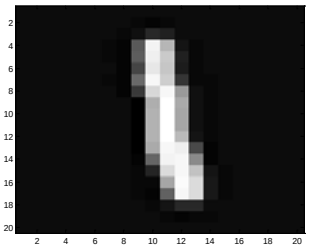
8.1.2. Ejemplos para búsqueda de patrones y clasificación.

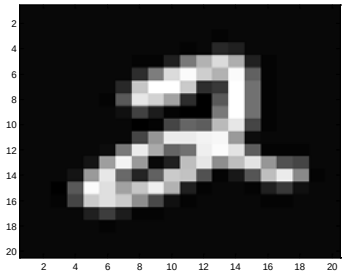
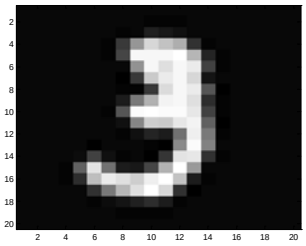
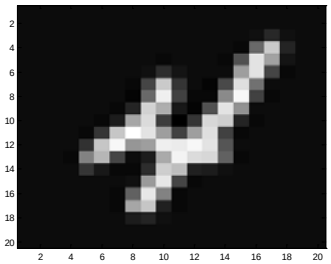
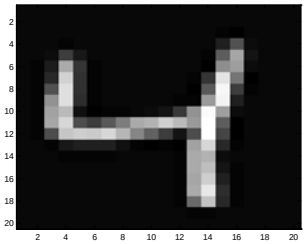
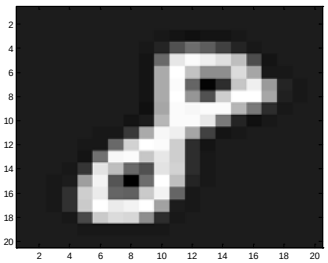
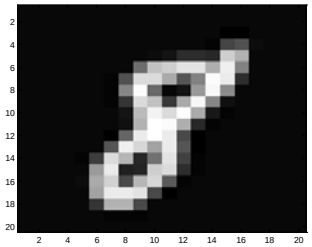
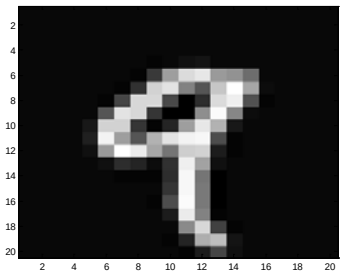
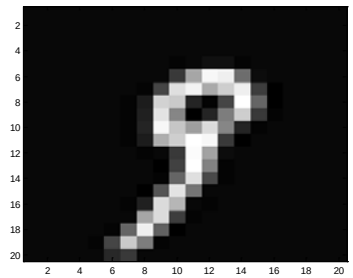
8.1.2.1. Reconocimiento de dígitos

[Fuente: Curso profesor Ng [7]].

El primer ejemplo que se desarrolla a continuación supone una aplicación típica de clasificación o búsqueda de patrones de las redes neuronales: reconocimiento de dígitos. Se crearán redes de distintas arquitecturas o topologías para comparar y se entrenarán con un conjunto de datos que contiene 5000 imágenes cuadradas, cada una de ella dividida en 400 píxeles, de manera que el tono en gris de cada pixel se codifica con un número que se halla dentro de un determinado intervalo. Dentro de una escala de grises, se codifican los distintos tonos de manera que cuanto menor es el número, más oscuro es el píxel, lo que indica menor contacto del dedo o el útil empleado para generar los ejemplos. Cuanto mayor es el número, más claro será el tono de gris, lo que indica mayor contacto, o más proximidad a la zona de contacto total.

La matriz de inputs se llamará X para este ejemplo. Su dimensión es 400x5000: 5000 ejemplos de 400 variables cada uno. Veamos algunas representaciones gráficas en escala de grises de los números:

Nº	Ejemplo 1	Ejemplo 2
10		
1		

2		
3		
4		
...	...	...
8		
9		

Como podemos ver para cada uno de los números hay múltiples ejemplos, todos ellos distintos. En el vector objetivo "y", tienen asociado el número que les corresponde, para poder entrenar la red.

Así, por ejemplo, el ejemplo 1 del dígito 1 se corresponde con las 400 variables de la columna 501 del vector X. En el vector y, 10x5000, aparecerá, en la columna 501, fila 2, un 1 (el resto de filas de esa columna serán 0).

Para dibujar la imagen, es necesario poner el vector columna correspondiente en forma de matriz 20x20:

```
sel_1_1=reshape(X(:,501),20,20);
```

Y representarlo mediante el comando imagesc.

En este ejemplo se ve muy clara la aplicación de las redes neuronales para reconocimiento de patrones y clasificación: se le presentan una gran cantidad y diversidad de ejemplos, todos ellos distintos en la matriz de inputs, pero agrupados por categorías en la matriz de "targets". Gracias a estas dos matrices podemos entrenar la red para que cuando se le presente un ejemplo caracterizado por las mismas variables que los ejemplos de la matriz de inputs pero con valores numéricos distintos, sea capaz de clasificar el nuevo ejemplo dentro de una de las categorías que aparecen en la matriz "targets". Ahora veamos una de las posibles formas que ofrece Matlab para crear una red "con memoria", capaz de clasificar los dígitos.

El vector target que hemos de conseguir tendrá un aspecto similar a esto:

Número	*10*	*10*	*…*	*1*	*…*	*2*	*…*	*9*
	0	0	…	1	…	0	…	0
	0	0	…	0	…	1	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	0
	0	0	…	0	…	0	…	1
	1	1	…	0	…	0	…	0

Para ello, implementamos en Matlab el siguiente código:

```
for i=1:5000,
    for j=1:10,
      if y(1,i)==j target_digit(j,i)=1;
      else target_digit(j,i)=0;
      end;
```

```
    end;
end;
```

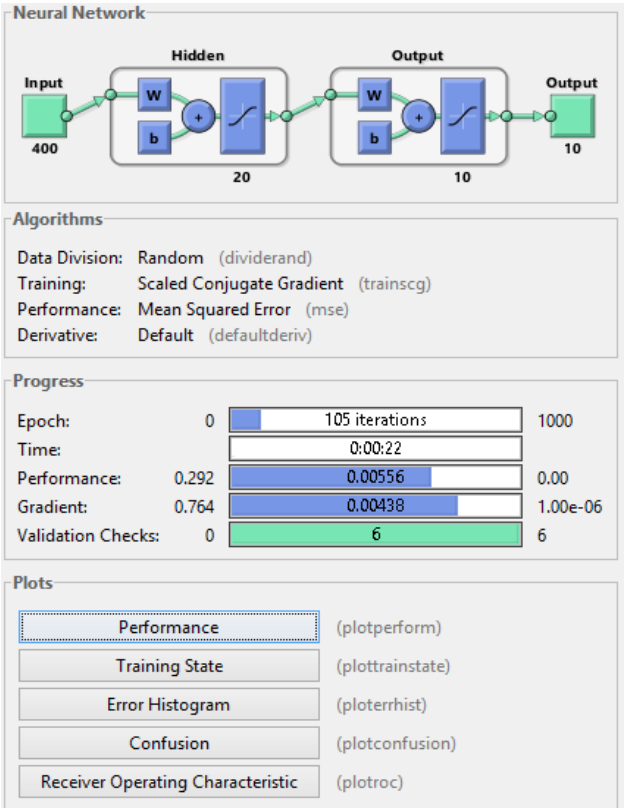
Establecemos la correspondiente división de datos, explicada en este capítulo:

```
net.divideParam.trainRatio = 60/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 20/100;
```

Ahora comenzamos a probar topologías para tratar de optimizar el problema:

- **Comenzamos con 1 capa oculta de 20 neuronas.**

```
hiddenLayerSize=20;
[net_digit_v11,tr_digit_v11]=trainscg(net_digit_v11, X, target_digit);
```

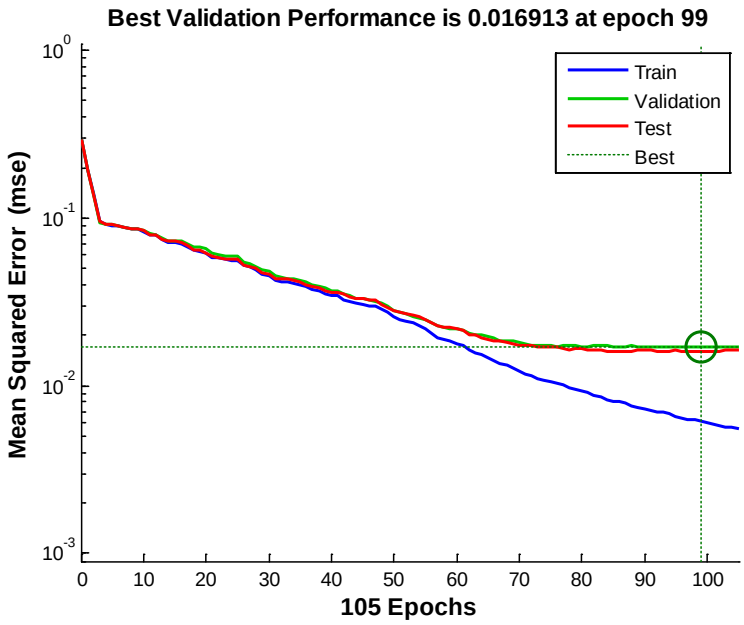


Matlab nos muestra información sobre la red creada y el proceso de entrenamiento que se ha seguido:

- En primer lugar nos muestra la forma topológica de la red, el número de capas, el número de neuronas por capa y la función de activación de cada neurona.
- En segundo lugar nos muestra un apartado donde recoge el algoritmo seguido para crear la red. En este caso, la división de los datos ha sido aleatoria, se ha usado como algoritmo de entrenamiento el explicado en este capítulo (gradiente escalado conjugado), la función a optimizar ha sido el error cuadrático medio, y el método para derivar que se ha utilizado es el implementado por defecto.
- En el siguiente apartado nos muestra el proceso de entrenamiento. En la captura de pantalla realizada anteriormente podemos ver la situación final, tras el entrenamiento. Han sido necesarias 105 épocas, en las que se han invertido 22 segundos, para conseguir un error cuadrático medio de 0.00556 y un gradiente de 0.00438, con seis comprobaciones de validación.
  Como se puede observar, para los criterios de parada se han dejado los valores existentes por defecto: 1000 épocas, sin límite de tiempo, mse=0.0, gradiente=10^{-6}, 6 comprobaciones de validación.
  El color azul claro resalta el criterio que ha motivado la parada del entrenamiento: se han alcanzado las 6 comprobaciones de validación y, para evitar el sobreajuste, se ha parado el entrenamiento.

- El último apartado es el de representaciones gráficas. Veamos algunos ejemplos:

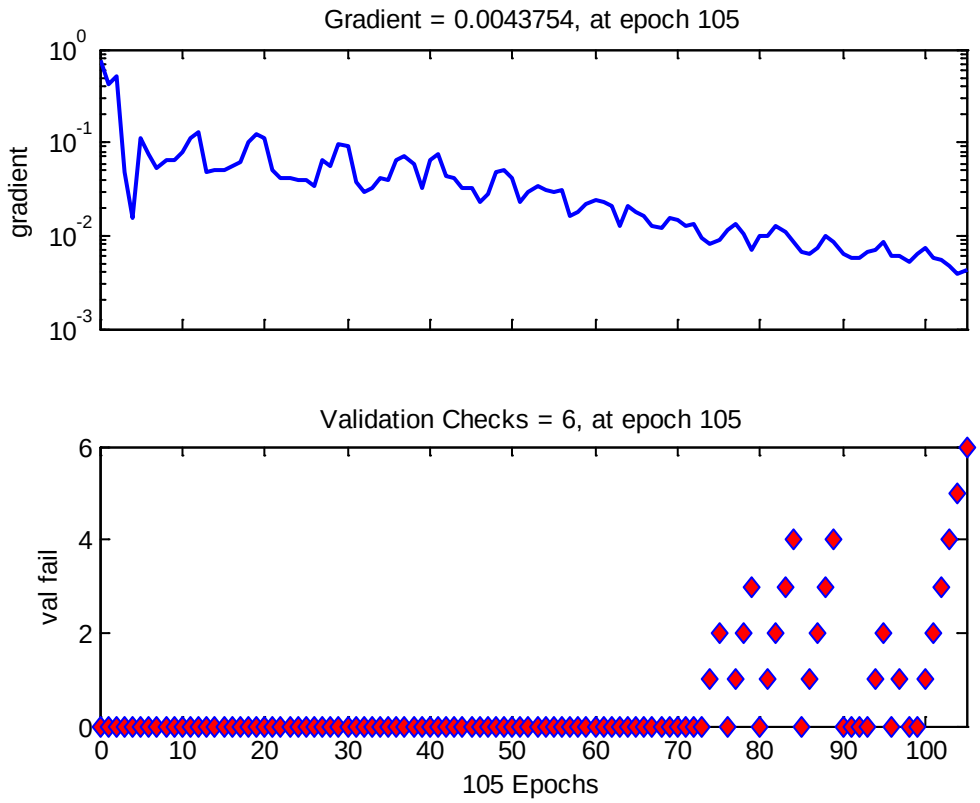
a) MSE:



Conforme va aumentando el número de épocas, el error cuadrático medio de la red va disminuyendo, ya que aumenta su entrenamiento, su capacidad para clasificar. Hay una

zona en la gráfica en la que se observa que los errores para los conjuntos de validación y test se estabilizan, mientras que el error del conjunto de entrenamiento sigue disminuyendo. Esto ocurre así porque cuantas más épocas se den, mejor se ajustarán los pesos de la red a los datos de entrenamiento. Sin embargo, este buen ajuste lleva asociada una indeseable pérdida de generalidad, que controlamos mediante las comprobaciones de validación.

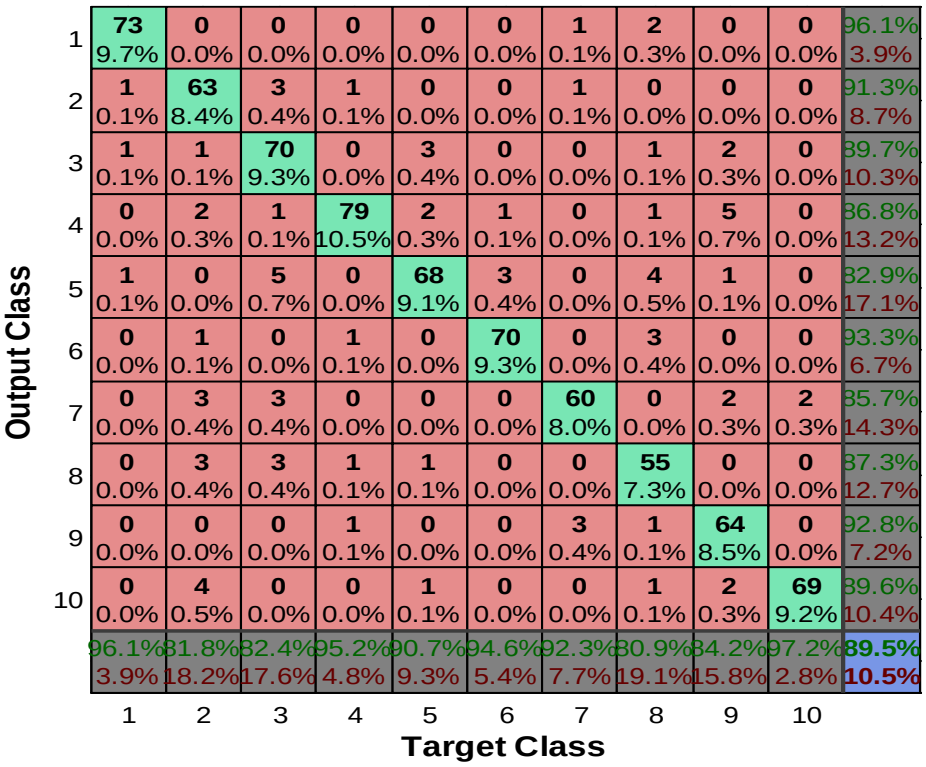
b) Estado del entrenamiento.



En la gráfica superior vemos que el valor del gradiente va disminuyendo conforme va aumentando el número de épocas.

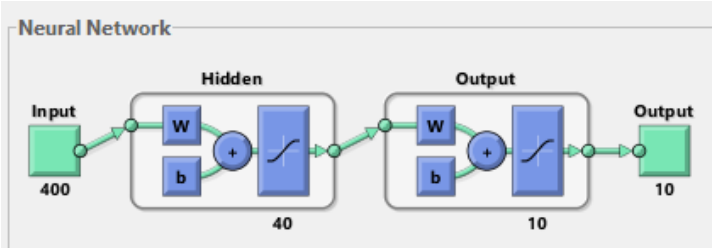
En la gráfica inferior vemos que las comprobaciones para validación se dan especialmente al final, donde la red comienza a ajustarse bastante bien a los datos de entrenamiento. Alcanzadas las seis comprobaciones, el entrenamiento se detiene, para evitar el aumento del error en el entrenamiento de la red.

c) Ahora vemos el mapa que nos muestra el número de aciertos.

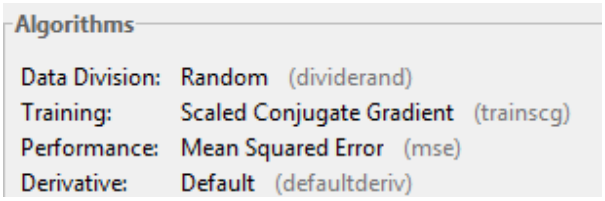


Nos fijaremos en el conjunto Test (aunque también se testea con los conjuntos de entrenamiento y validación), puesto que es el único que no se ha utilizado para nada durante el entrenamiento de la red. Vemos que el porcentaje final de acierto es del 89.5%. También podemos extraer algunas conclusiones y relaciones interesantes, como que el número 9 con el que más confusión presenta es con el 4, o la equivocación de interpretar un cinco, cuando en realidad lo que se ha dibujado es un tres.

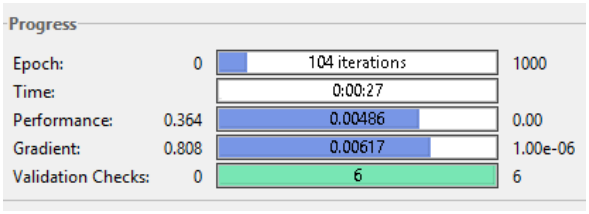
- **Una capa oculta de 40 neuronas.**



- Funciones de entrenamiento:

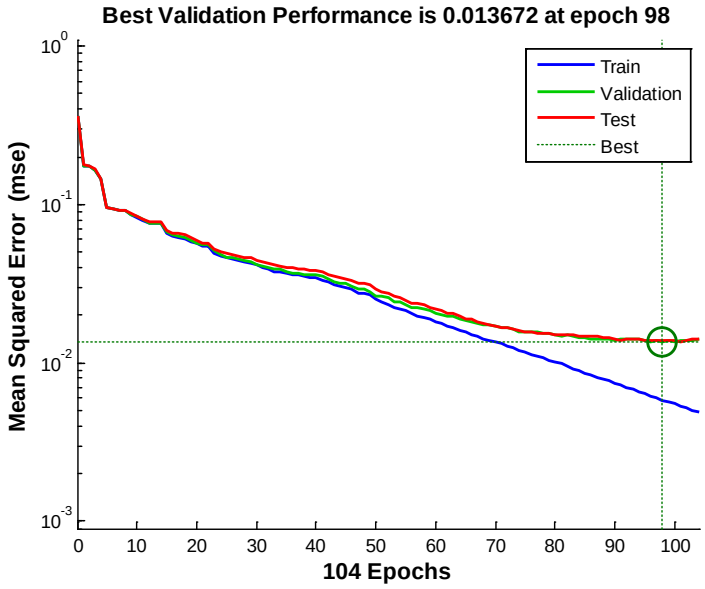


- Proceso de entrenamiento:

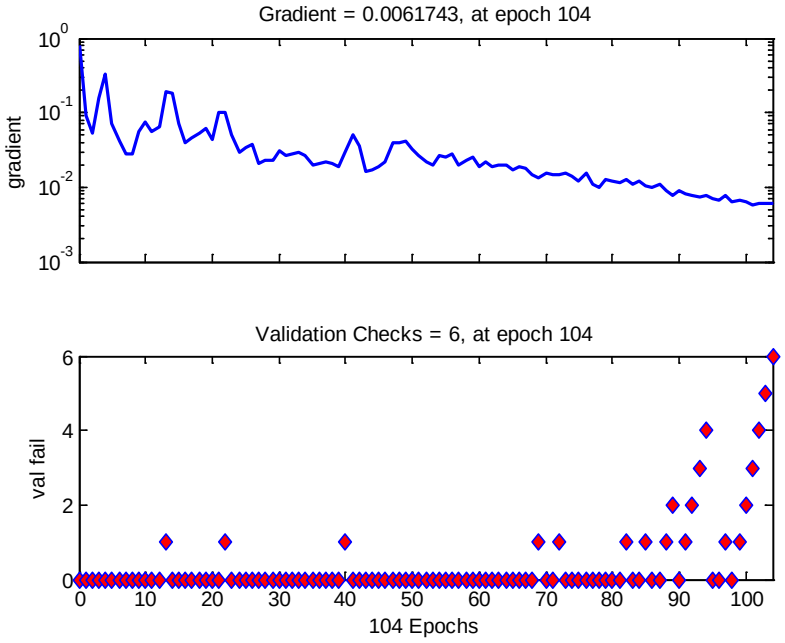


- Representaciones gráficas:

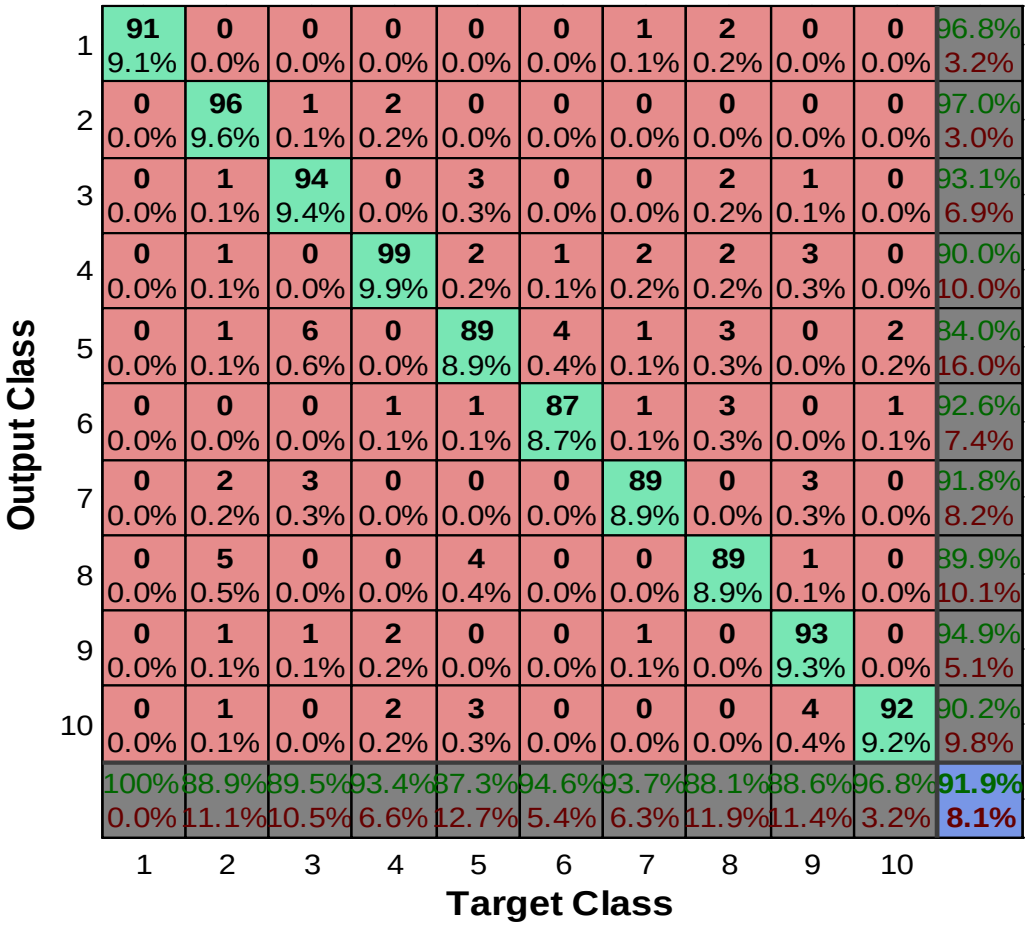
a) MSE:



b) Evolución del entrenamiento:

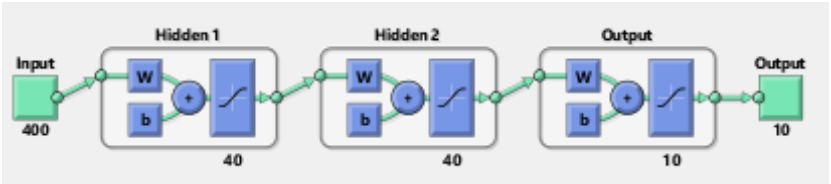


c) Mapa de aciertos:

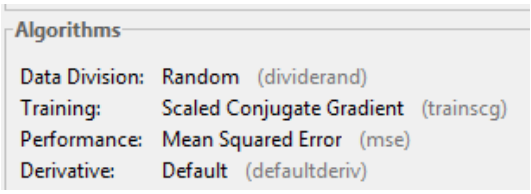


Se observan ligeras mejoras respecto a la arquitectura de 20 neuronas con poca penalización de tiempo (apenas 5 segundos extra), existiendo en el conjunto de test un 91.9 % de aciertos.

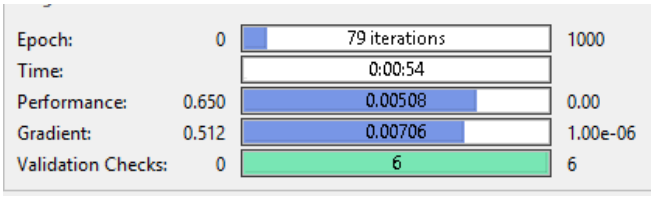
- **Dos capas de cuarenta neuronas:**



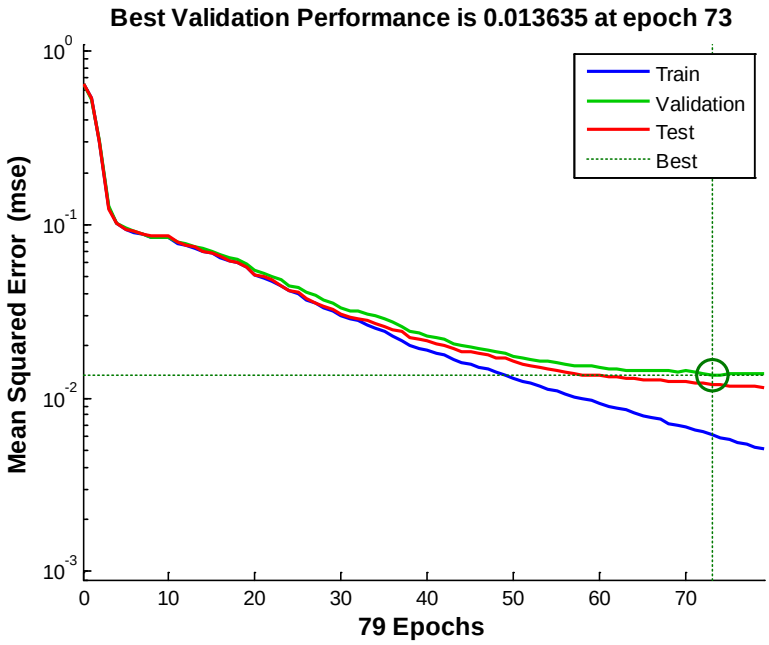
- Funciones de entrenamiento:



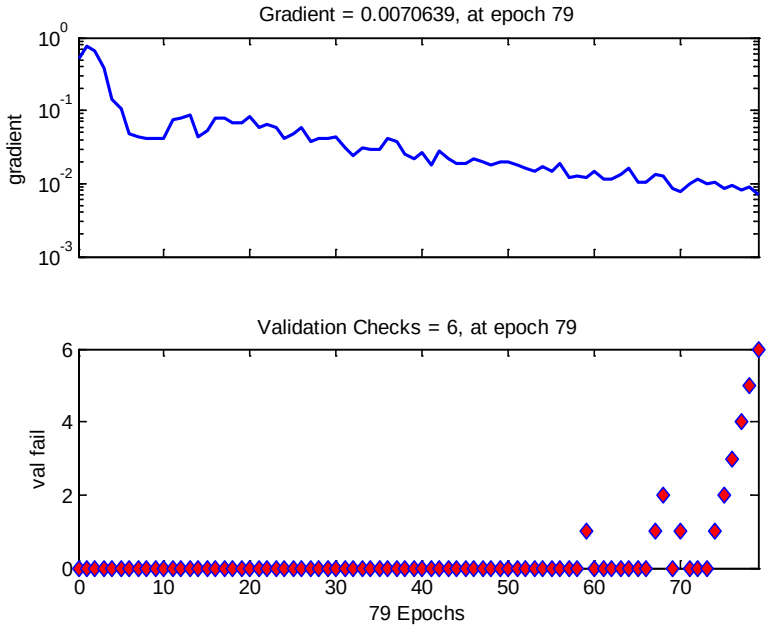
- Proceso de entrenamiento.



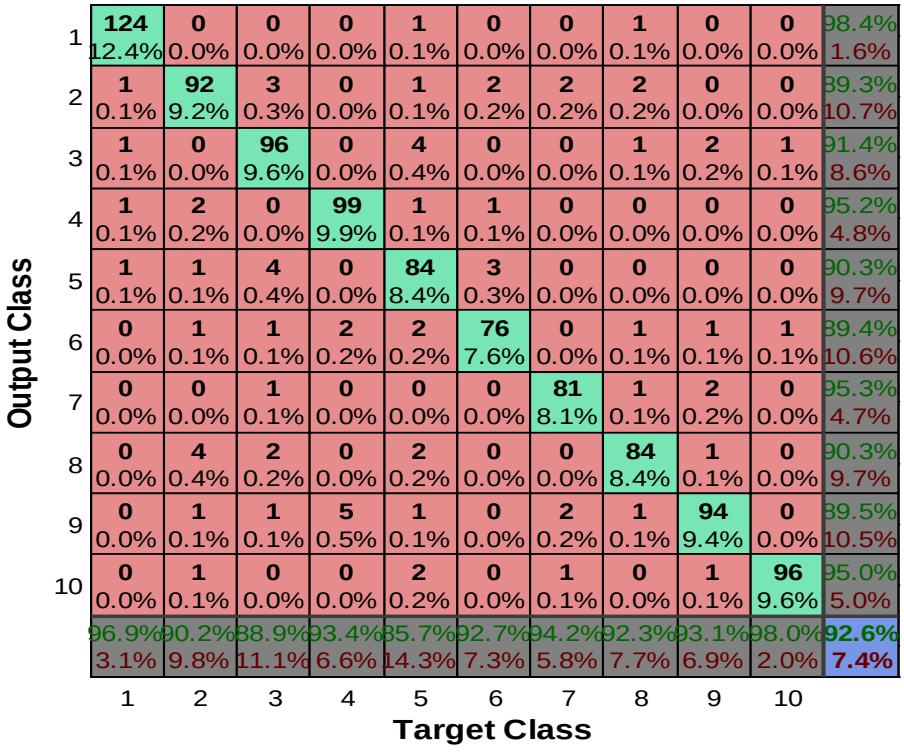
- Representaciones gráficas:
  a) MSE:



b) Entrenamiento:



c) Mapa de aciertos:



	1	2	3	4	5	6	7	8	9	10	
1	124 12.4%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	98.4% 1.6%
2	1 0.1%	92 9.2%	3 0.3%	0 0.0%	1 0.1%	2 0.2%	2 0.2%	2 0.2%	0 0.0%	0 0.0%	89.3% 10.7%
3	1 0.1%	0 0.0%	96 9.6%	0 0.0%	4 0.4%	0 0.0%	0 0.0%	1 0.1%	2 0.2%	1 0.1%	91.4% 8.6%
4	1 0.1%	2 0.2%	0 0.0%	99 9.9%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.2% 4.8%
5	1 0.1%	1 0.1%	4 0.4%	0 0.0%	84 8.4%	3 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.3% 9.7%
6	0 0.0%	1 0.1%	1 0.1%	2 0.2%	2 0.2%	76 7.6%	0 0.0%	1 0.1%	1 0.1%	1 0.1%	89.4% 10.6%
7	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	81 8.1%	1 0.1%	2 0.2%	0 0.0%	95.3% 4.7%
8	0 0.0%	4 0.4%	2 0.2%	0 0.0%	2 0.2%	0 0.0%	0 0.0%	84 8.4%	1 0.1%	0 0.0%	90.3% 9.7%
9	0 0.0%	1 0.1%	1 0.1%	5 0.5%	1 0.1%	0 0.0%	2 0.2%	1 0.1%	94 9.4%	0 0.0%	89.5% 10.5%
10	0 0.0%	1 0.1%	0 0.0%	0 0.0%	2 0.2%	0 0.0%	1 0.1%	0 0.0%	1 0.1%	96 9.6%	95.0% 5.0%
	96.9% 3.1%	90.2% 9.8%	88.9% 11.1%	93.4% 6.6%	85.7% 14.3%	92.7% 7.3%	94.2% 5.8%	92.3% 7.7%	93.1% 6.9%	98.0% 2.0%	**92.6%** **7.4%**

El tiempo se duplica y no hay mejoras significativas en lo que a fiabilidad se refiere.

8.1.2.2. Ejemplo: Diagnóstico del cáncer de mama.

[Fuente de datos: [8]].

En el enlace, se ofrece un conjunto de 569 ejemplos, de 32 variables cada uno. Cada ejemplo se corresponde con la imagen digitalizada de una FNA de una masa de mama. Las variables de la imagen permiten codificar características de los núcleos de las células.

En líneas generales, el archivo tiene la siguiente estructura:

2) Número de identificación.
3) Diagnóstico (M: Maligno, B: Benigno).

Diez características con valores reales se calculan para cada núcleo de la célula:

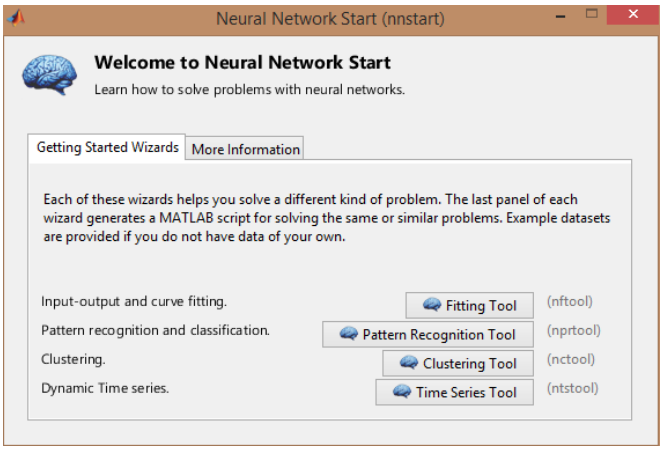
a) radio (media de las distancias de centro a puntos en el perímetro);
b) la textura (desviación estándar de los valores de la escala de grises);
c) perímetro;
d) área;
e) la suavidad (variación local en longitudes de radio);
f) compacidad (perímetro ^ 2 / zona - 1,0);
g) concavidad (severidad de las porciones cóncavas del contorno);
h) puntos cóncavos (número de porciones cóncavas del contorno);
i) la simetría;
j) la dimensión fractal ("aproximación costa" - 1);

El objetivo es construir una red neuronal que sea capaz de predecir en base a las características de los núcleos de las células, si el cáncer es maligno o benigno.

Para ello, en primer lugar, preparamos el conjunto de datos que inicialmente tenemos (32x569). Eliminamos la columna del identificador, y separamos la columna de las “Targets”, quedándonos únicamente con una matriz de variables 30x569.

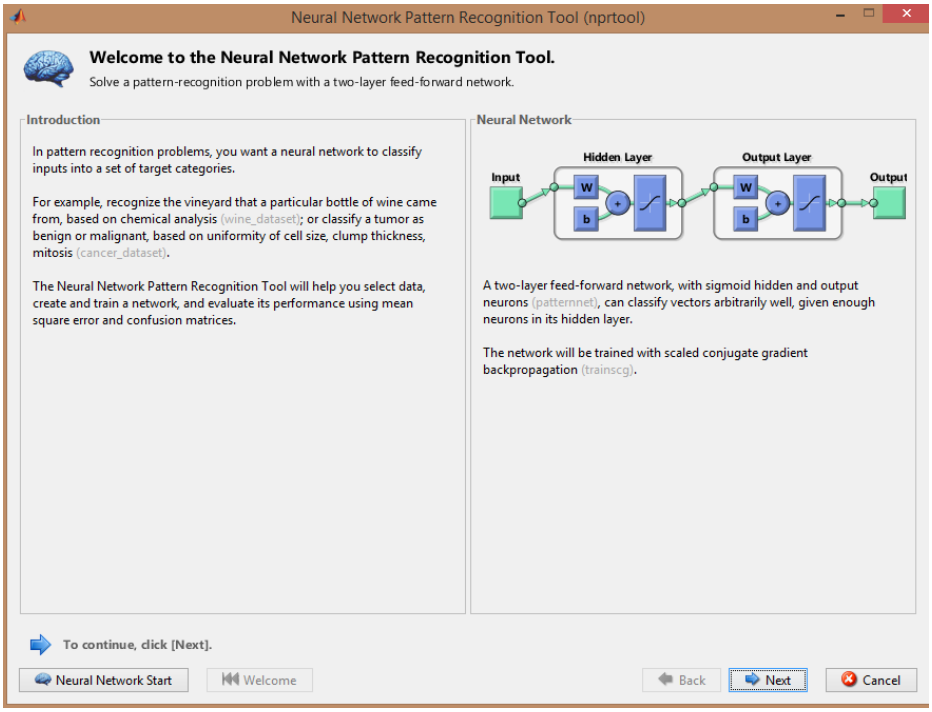
A continuación preparamos el vector “Target”, como un vector 2x569.

Ya estamos en disposición de entrenar la red. Como alternativa a la metodología de resolución presentada para el ejemplo anterior, se propone para la resolución de este ejemplo utilizar una interfaz gráfica de Matlab, que se inicia introduciendo en la ventana de comandos “nnstart”. Se abre la siguiente ventana:

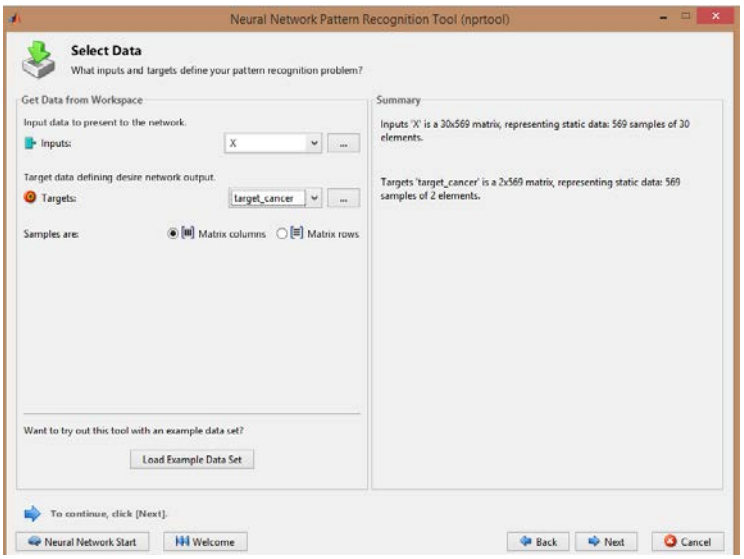


Esta ventana nos presenta la posibilidad de elegir entre cuatro posibles itinerarios. La aplicación que requiere la resolución de este ejemplo es “Pattern Recognition Tool”.

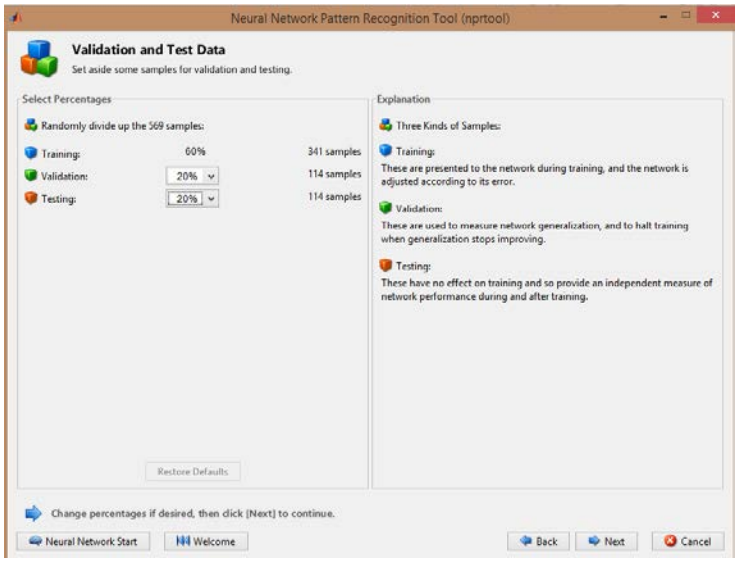
En la ventana que se abre a continuación se nos ofrece información por defecto asociada a esta aplicación como la arquitectura de la red y el algoritmo de entrenamiento que se va a utilizar (retropropagación con gradiente escalado conjugado):



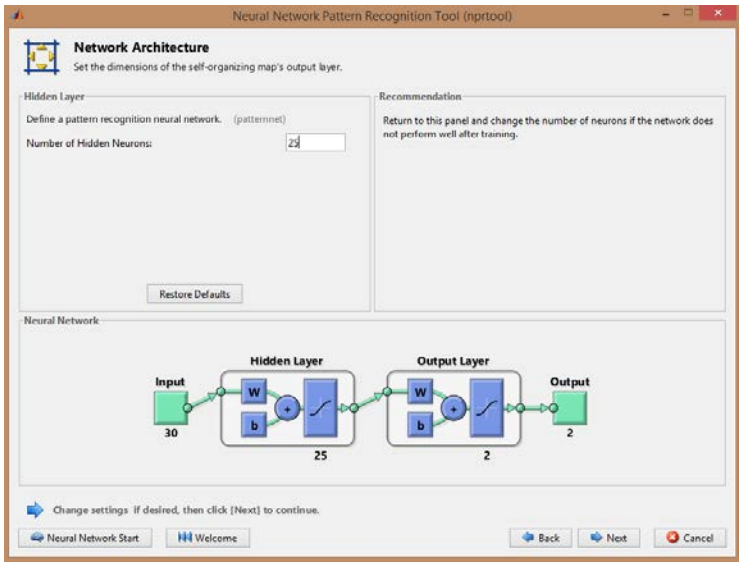
Ahora seleccionamos las matrices necesarias para poder entrenar la red (inputs y targets), que habrán de estar definidas en el espacio de trabajo cargado.



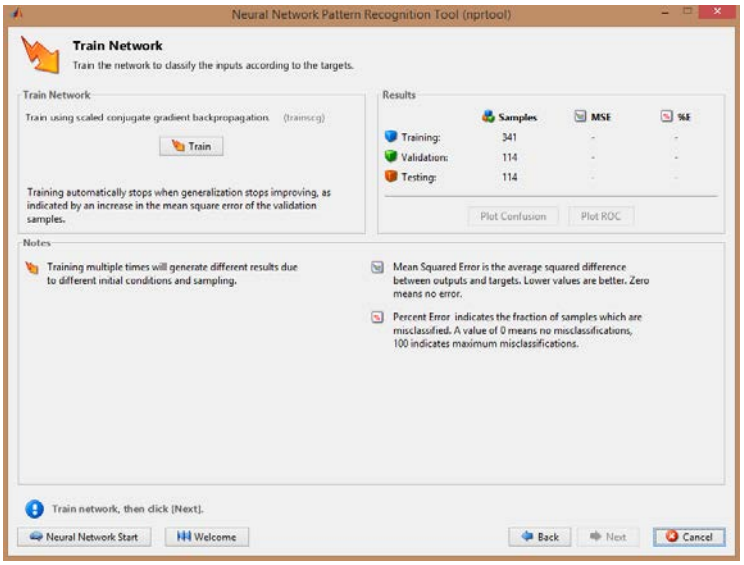
Ahora hemos de elegir la proporción de datos que dedicamos a entrenamiento, la proporción que dedicamos a validación y la proporción que dedicamos a test. Para este ejemplo, se ha decidido usar un 60% de los datos para entrenar la red, un 20% para validación, y un 20% para comprobación final:



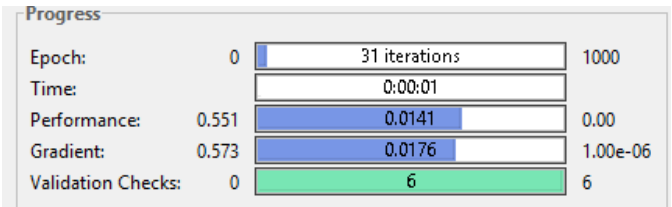
En la siguiente pantalla se nos pide que introduzcamos el número de neuronas de la capa oculta, y nos muestra un esquema de la arquitectura de la red:



Finalmente se nos muestra una ventana de resumen donde ya estamos en disposición de entrenar la red:

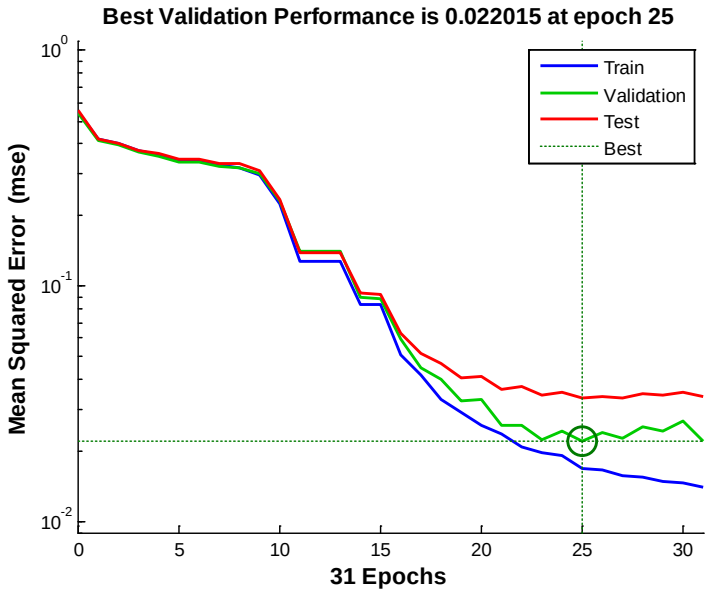


Entrenamos la red obteniendo el siguiente cuadro de resultados:

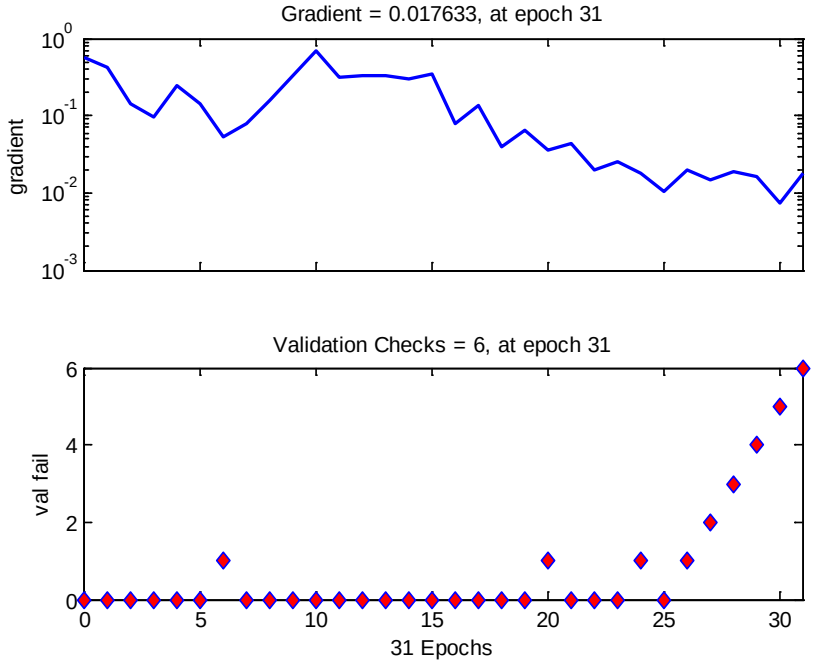


Veamos ahora la evolución del entrenamiento y la calidad de los resultados:

a) Evolución del MSE:



b) Evolución del gradiente y comprobaciones con el conjunto de validación:



c) Mapas de aciertos.

Para el conjunto de datos de testeo vemos que el acierto es considerablemente alto: 96.5%. Con un entrenamiento de apenas un segundo y 31 iteraciones hemos conseguido unos valores para los pesos que ajustan muy bien los datos. Esto es así porque siguen patrones muy marcados.

8.1.2.3. Ejemplo: Vinos.

[Fuente de datos: [8]]

Estos datos son el resultado de un análisis químico de los vinos cultivados en la misma región en Italia, pero se derivan de tres cultivos diferentes. El análisis determina las cantidades de 13 componentes que se encuentran en cada uno de los tres tipos de vinos.

Los atributos son:

1) El alcohol
2) El ácido málico
3) Ceniza
4) Alcalinidad de cenizas
5) Magnesio
6) fenoles totales
7) Los flavonoides
8) fenoles Nonflavanoid
9) proantocianinas
10) La intensidad de color
11) Hue
12) OD280/OD315 de vinos diluidos
13) Prolina

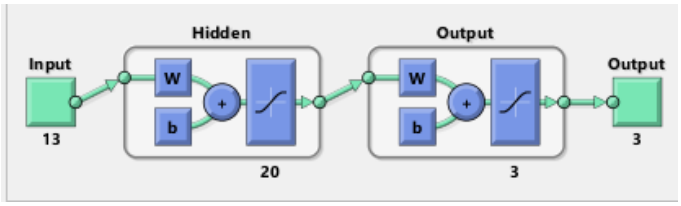
El archivo que se nos proporciona en el enlace consta de 178 ejemplos con las 13 variables y el cultivo al que pertenecen. Para la preparación de los datos se procede de manera análoga a los ejemplos anteriores:

Se obtiene una matriz de inputs llamada X de dimensiones 13x178 y una matriz de 3x178 de “targets” llamada target_vino, donde el 3 indica el número de grupos para la clasificación.

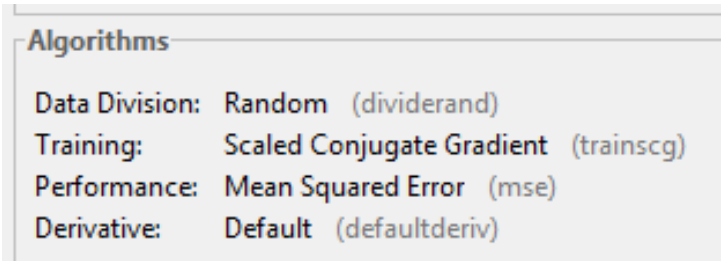
Elegimos una arquitectura compuesta por una capa oculta de 20 neuronas y la implementamos mediante el siguiente código:

```
hiddenLayerSize=20;

net_vino_v2=patternnet(hiddenLayerSize,'trainscg');

net_vino_v2.divideParam.trainRatio = 60/100;

net_vino_v2.divideParam.valRatio = 20/100;

net_vino_v2.divideParam.testRatio = 20/100;

[net_vino_v2,tr_vino_v2]=trainscg(net_vino_v2, X, target_vino);
```

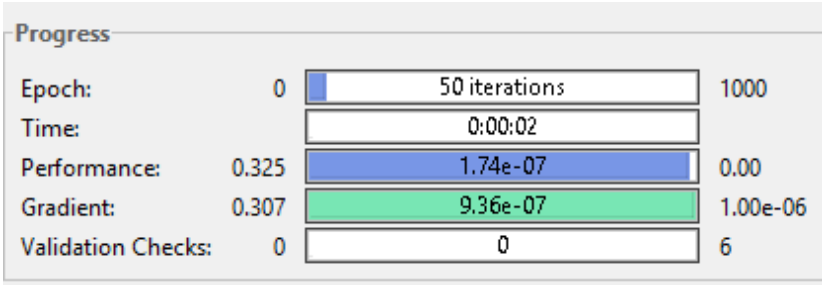
- Obtenemos la siguiente arquitectura de red:



- Utiliza los siguientes algoritmos:

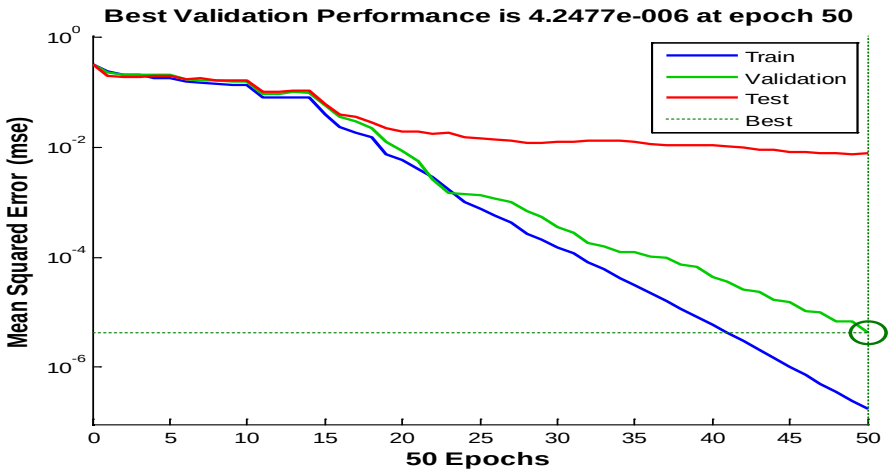


- El estado final del proceso de entrenamiento es el siguiente:



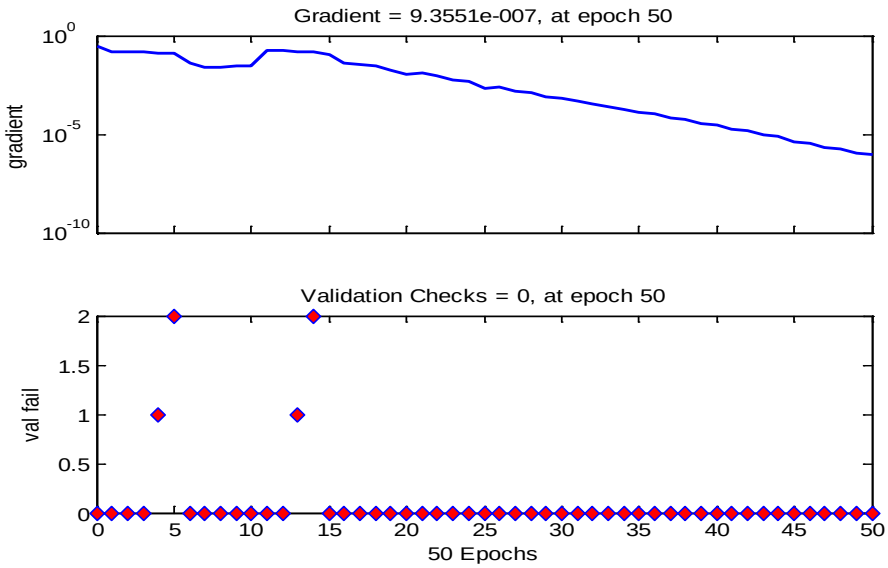
- Análisis de la solución:

a) MSE:



Vemos que si aumentamos el número de épocas, el error cuadrático medio de los conjuntos de entrenamiento y validación sigue disminuyendo. En cambio el error del conjunto test se ha estabilizado.

b) Gradiente y validación:



c) Gráficas de errores:

Vemos que en el análisis realizado para cada uno de los conjuntos, únicamente se ha registrado un fallo, situado en el conjunto de testeo, en el que un vino del cultivo 1 ha sido confundido con un vino del cultivo 2.

Es reseñable que en apenas dos segundos, el error alcanzado es prácticamente cero, aunque la parada ha estado motivada por el valor tan bajo del gradiente, que provocaba que tras cada iteración la variación de pesos fuera insignificante.

8.2. Análisis para Ingeniería de Telecomunicaciones.

8.2.1. Estudio de nota media de alumnos egresados para Ingeniería de Telecomunicaciones.

8.2.1.1. Clasificación.

El objetivo de esta sección es el estudio de la predictibilidad del rendimiento académico de alumnos de Ingeniería de Telecomunicaciones mediante Redes Neuronales Artificiales para clasificación.

8.2.1.1.1. Estudio a partir de datos PAU.

a) Estudio previo de los datos.

La Red Neuronal clasificará, en función de las variables de entrada, a los alumnos en uno de los grupos establecidos en el texto principal. En primer lugar, veamos algunos parámetros característicos de las variables de entrada:

Variable	Media	Desviación típica
Nota PAU en la parte 1	7.2647	1.0117
Ranking parte 1/Total presentados	0.2002	0.1876
Nota PAU en la parte 2	7.2729	1.4050
Ranking parte 2/Total presentados	0.1889	0.1899
Calificación PAU	8.0047	0.9251
Ranking calificación PAU/Total presentados	0.1596	0.1623
Calificación física PAU	7.0365	2.0311
Ranking calificación física/Total presentados	0.2672	0.2289
Calificación Matemáticas PAU	7.5347	2.0108
Ranking calificación matemáticas/Total presentados	0.2784	0.2293

Vemos que los alumnos DURM que ingresan en Ingeniería de telecomunicaciones presentan una media en la PAU bastante alta (8 puntos sobre 10). Además su posición respecto al resto de alumnos DURM es bastante buena pues la media de los apartados de ranking es siempre baja: de cada cien alumnos DURM, la posición media de un alumno de nuevo ingreso en ingeniería de telecomunicaciones para las variables consideradas nunca estaría más allá de la 30, siendo especialmente significativo que de cada cien alumnos presentados a la PAU, la posición media considerando la nota global de la PAU es aproximadamente la 15.

Indicador de la variable de salida:

	Valor medio	Desviación típica
Nota egresado	1.6370	0.3669

Vemos que el valor medio para la calificación media de los egresados de la titulación de Ingeniería de Telecomunicaciones es de aprobado alto, con una desviación típica pequeña (0.3669) respecto al intervalo considerado (1-4). Está desviación típica tan pequeña influirá decisivamente en los resultados obtenidos, tal y como se explicará tras desarrollar los respectivos entrenamientos.

b) Entrenamiento:

El proceso de entrenamiento se llevará a cabo con un conjunto de datos correspondiente a 170 alumnos de la Escuela de Telecomunicaciones, en las condiciones establecidas en el capítulo III.

Utilizaremos para el entrenamiento el algoritmo del Gradiente Escalado Conjugado, descrito en el capítulo 2. El conjunto de entrenamiento estará formado por el 70% de los datos disponibles, el conjunto de validación estará compuesto por el 15% de los datos, y el conjunto de test estará compuesto por el restante 15%.

Se llevarán a cabo cuatro entrenamientos, teniendo lugar el primero y el segundo a igualdad de condiciones entre sí, y el tercero y el cuarto también a igualdad de condiciones entre sí, tal y como recogen las tablas siguientes. El objetivo de realizar este emparejamiento es ver la variabilidad de los resultados de la red por la aleatoriedad en la inicialización de los pesos.

b.1) Entrenamientos 1 y 2:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	170
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

b.2) Entrenamientos 3 y 4:

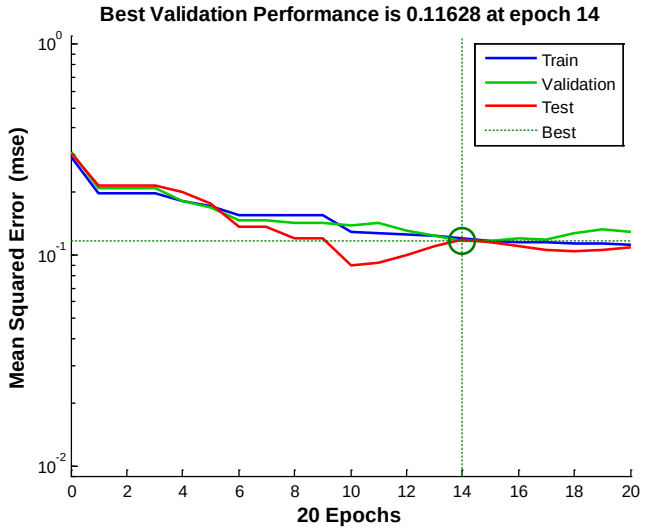
Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	170
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25-2/25

c) Comparación de resultados:

Entrenamiento	1	2	3	4
Épocas	20	11	15	15
Tiempo (s)	1	3	1	0
Rendimiento	0.113	0.111	0.105	0.104
Gradiente	0.0331	0.0361	0.0236	0.0248
Comprobaciones de validación	6	6	6	6
% aciertos (1.0-1.5)	88.9	55.6	100	85.7
% aciertos (1.5-2.0)	71.4	83.3	0	33.3
% aciertos (2.0-2.5)	0	0	0	0
% aciertos (2.5-3.0)	NaN	NaN	NaN	NaN
% aciertos (3.0-4.0)	NaN	0	NaN	NaN
% aciertos General	68.4	52.56	47.4	47.4

d) Conclusiones:

El número de épocas no es un parámetro relevante pues tal y como vemos en la evolución del error cuadrático medio a lo largo de las épocas, si aumentamos el número de estas no conseguimos la reducción de aquel. De tiempo, rendimiento y gradiente tampoco obtenemos información importante.



Cabe destacar que al aumentar el número de capas ocultas se ha producido un empeoramiento del tanto por ciento de aciertos de la red aunque ello, por las conclusiones que se explican a continuación, no resultará decisivo.

La aleatoriedad en la inicialización de los pesos resulta determinante pues la variación en el tanto por ciento de aciertos para el caso de una sola capa oculta ronda el 20%.

Otro aspecto reseñable de las soluciones obtenidas, es la inexistencia de datos o el nulo porcentaje de aciertos para los grupos 2.0-2.5, 2.5-3.0, 3.0-4.0. Y es en este punto donde recordamos los parámetros "media" y "desviación típica" calculados para la nota media de egreso. Sucede que 147 de los 170 alumnos de la base de datos empleada obtuvieron una calificación media menor que 2. Ello incapacita a la red para predecir el rendimiento de alumnos que obtengan notable o sobresaliente. Este aspecto, unido a la gran variabilidad que introduce la inicialización aleatoria de los pesos para las dos categorías en que se divide a los aprobados, hace que los resultados obtenidos no sean satisfactorios.

Por tanto, concluimos que para los alumnos de Ingeniería de Telecomunicaciones no es posible predecir mediante Redes Neuronales Artificiales de clasificación la nota media de la carrera a partir de las variables de acceso a la universidad consideradas.

8.2.1.1.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

a) Estudio previo de los datos.

En este apartado se realizarán distintos entrenamientos con el objetivo de ver hasta qué punto se puede predecir la nota de egreso de un alumno de Telecomunicaciones a partir de las variables PAU y las asignaturas más complicadas de la carrera. Introduciremos las asignaturas en orden cronológico sobre la matriz de entrada del apartado anterior para ver si existe un punto de inflexión que genere buenos resultados.

Asignatura	Variable	Media	Desviación típica
Electrónica analógica	N	6.5683	1.2542
	CP	1.5329	0.8628
	CT	1.7605	1.5218
Campos electromagnéticos	N	6.1042	0.8863
	CP	2.1138	1.4329
	CT	2.8922	3.2966
Diseño de circuitos y sistemas electrónicos	N	6.8677	1.5023
	CP	1.3114	0.7273
	CT	1.7784	1.8641

Redes de ordenadores	N	6.3162	1.1213
	CP	2.2814	1.2068
	CT	2.0778	1.7801
Tratamiento digital de señales	N	6.4234	1.2370
	CP	1.4431	0.8958
	CT	1.8263	1.5981

Vemos que las notas medias obtenidas en estas asignaturas se hallan en torno a los 6.5 puntos. En cuanto a convocatorias presentadas y convocatorias transcurridas hasta presentarse por primera vez, destaca campos electromagnéticos: los alumnos dejan pasar de media casi tres convocatorias hasta presentarse por primera vez y necesitan dos convocatorias para aprobar. Redes de ordenadores presenta datos parecidos.

b) Entrenamiento:

Se llevarán a cabo cinco entrenamientos introduciendo cronológicamente las asignaturas, en cada uno de los cuales se considerarán como variables de entrada todas las variables PAU ya descritas, más las tres variables por asignatura (N, CP, CT) de la nueva asignatura que se introduce en el conjunto, más las tres de las que se han introducido previamente. Por tanto, se usarán 13, 16, 19, 22 y 25 variables respectivamente. Se usarán los mismos alumnos que para el entrenamiento sólo con variables PAU, habiéndose eliminado tres que no tenían todos los campos completos (167 ejemplos: 70% para train, 15% para validación y 15% para test). Se empleará el gradiente escaldo conjugado y como arquitectura se empleará una red con una única capa oculta de 25 neuronas.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	167
Variables consideradas	13 (16,19,22,25)
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

c) Comparación de resultados.

Entrenamiento	1	2	3	4	5
Épocas	28	29	26	21	32
Tiempo (s)	2	2	2	2	3
Rendimiento	0.0830	0.0773	0.0478	0.0575	0.0404
Gradiente	0.0120	0.0371	0.0574	0.0511	0.0296
Comprobaciones de validación	6	6	6	6	6
% aciertos (1.0-1.5)	77.8	88.9	72.7	100	62.5
% aciertos (1.5-2.0)	16.7	50.0	83.3	33.3	85.7
% aciertos (2.0-2.5)	0.0	50.0	100	NaN	66.7
% aciertos (2.5-3.0)	0.0	NaN	NaN	0.0	0.0
% aciertos (3.0-4.0)	0.0	NaN	0.0	NaN	NaN
% aciertos General	42.1	68.4	73.7	63.2	68.4

d) Conclusiones.

Tal y como se puede observar los resultados no son satisfactorios pues la clasificación es bastante mala. Ello está relacionado con que la mayoría de los ejemplos de entrenamiento están en un único grupo y eso, además, depende del reparto aleatorio de los ejemplos en los tres conjuntos (training, validation y test). También podemos observar, como para el caso en el que sólo considerábamos variables PAU, la gran variabilidad introducida por la inicialización aleatoria de los pesos.

8.2.1.2. Ajuste de funciones.

8.2.1.2.1. Estudio a partir de datos PAU.

a) Entrenamiento.

Se desarrollan cuatro entrenamientos, siendo 1 y 2, y 3 y 4 iguales entre sí. Se utilizará como algoritmo de aprendizaje el de Levenberg-Marquardt. Se dispone de 170 ejemplos (70% para test, 15% para validación y 15% para test).

a.1.) Entrenamientos 1 y 2.

Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	170
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

a.2.) Entrenamientos 3 y 4.

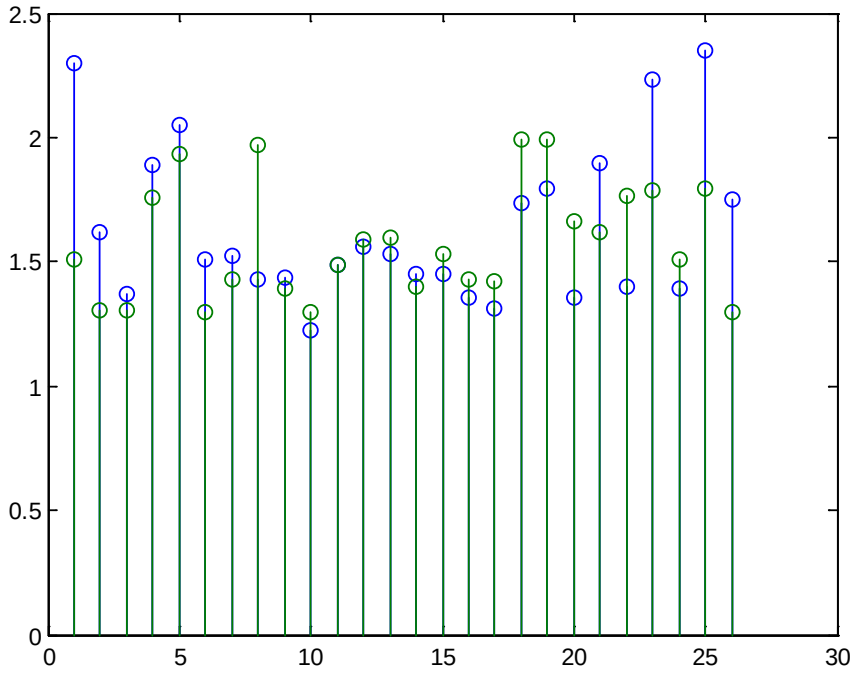
Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	170
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/10-1/10

b) Resultados.

Entrenamiento	1	2	3	4
Épocas	10	11	13	13
Tiempo (s)	5	7	0	8
Rendimiento	0.0213	0.0373	0.0473	0.0238
Gradiente	0.168	0.0121	0.120	0.0464
Mu	0.0100	0.100	0.0100	0.100
Comprobaciones de validación	6	6	6	6
|Error máximo|	1.026	0.7108	0.7771	0.8391
MSE	0.2168	0.0946	0.0874	0.1347

c) Conclusiones.

En la mejor iteración (la tercera), la comparación de los “Targets” con los “Outputs” de la red presenta el siguiente aspecto:



Vemos que hay alumnos para los cuales las predicciones son bastante malas como 1, 8, 22, 23, 25 o 26, en los que el error es mayor de medio punto. Hay otros en los que las predicciones son realmente buenas, como 3, 9, 10, 11, 12, 13, 14, 15, 16, 17, en los que el error de predicción apenas llega a 0.1 puntos y otros alumnos donde el error es intermedio.

Además, es importante destacar que no se han observado cambios significativos al variar la arquitectura de la red.

8.2.1.2.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

a) Entrenamiento.

Se desarrollan cinco entrenamientos y en cada uno de ellos se añadirán las variables (N, CP, CT) de una asignatura (electrónica analógica, campos electromagnéticos, diseño de circuitos y sistemas electrónicos, redes de ordenadores y tratamiento digital de señales) al conjunto de entrada existente previamente, partiendo del conjunto formado por las 10 variables PAU. Se utilizará como algoritmo de aprendizaje el de Levenberg-Marquardt. Se dispone de 167 ejemplos (70% para test, 15% para validación y 15% para test%).

Función de entrenamiento.	Trainlm
Ejemplos de entrenamiento	167
Variables consideradas	13 (16, 19, 22, 25)
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25

b) Comparación de resultados.

Entrenamiento	1	2	3	4	5
Épocas	10	10	9	9	9
Tiempo (s)	0	0	0	0	0
Rendimiento	0.00615	0.00265	1.81e-12	2.95e-24	3.59e-31
Gradiente	0.160	0.0331	9.65e-06	3.79e-12	1.07e-15
Mu	0.0100	0.0100	1.00e-09	1.00e-07	1.00e-12
Comprobaciones de validación	6	6	4	6	6
Error máximo	0.9087	1.396	0.6523	0.5106	0.7607
MSE	0.1370	0.2185	0.1076	0.0298	0.1067

c) Conclusiones.

Si bien es cierto que los resultados obtenidos nos son mucho mejores que los obtenidos en el apartado anterior, en el que sólo utilizábamos datos PAU, es preciso destacar el resultado obtenido en la iteración cuatro, donde se ha añadido la asignatura Redes de ordenadores. El error cuadrático medio es aproximadamente tres veces inferior al segundo MSE más pequeño y el error máximo cometido también es considerablemente pequeño.

8.2.1.3. Regresión lineal múltiple.

8.2.1.3.1. Estudio a partir de datos PAU.

Se ha realizado una Regresión Lineal Múltiple, al igual que para Ingeniería Industrial:

Ejemplos de entrenamiento	144
Ejemplos de test	26
Variables consideradas	10

Los resultados que se han obtenido vienen recogidos en la tabla comparativa del punto siguiente.

8.2.1.3.2. Estudio a partir de datos PAU y asignaturas más complejas de la carrera.

Se desarrollan cinco modelos de regresión y en cada uno de ellos se añadirán las variables (N, CP, CT) de una asignatura (electrónica analógica [A], campos electromagnéticos [B], diseño de circuitos y sistemas electrónicos [C], redes de ordenadores [D] y tratamiento digital de señales [E]) al conjunto de entrada existente previamente, partiendo del conjunto formado por las 10 variables PAU.

Ejemplos de entrenamiento	142
Ejemplos de test	25
Variables consideradas	13 (16, 19, 22, 25)

- Evolución de los estimadores.

	PAU	A	B	C	D	E
$\hat{\beta}_0$	-0.9965	-1.1727	-1.0038	-1.0467	-1.1334	-1.2211
$\hat{\beta}_1$	-0.0926	0.0102	0.0546	0.0764	0.0928	0.0941
$\hat{\beta}_2$	-0.5057	-0.0300	0.1612	0.2406	0.3095	0.3357
$\hat{\beta}_3$	0.1542	0.0481	0.0543	0.0836	0.0766	0.0641
$\hat{\beta}_4$	0.3049	-0.0464	0.1218	-0.0657	-0.1117	-0.1078
$\hat{\beta}_5$	0.4373	0.3096	0.1686	0.0635	0.0536	0.0454
$\hat{\beta}_6$	1.7360	1.3186	0.8239	0.4830	0.4900	0.3853
$\hat{\beta}_7$	-0.0857	-0.0428	-0.0318	-0.0206	-0.0151	-0.0041
$\hat{\beta}_8$	-0.5795	-0.4415	-0.4455	-0.0904	-0.0229	0.0634
$\hat{\beta}_9$	-0.0926	-0.0539	-0.0327	-0.0567	-0.0655	-0.0655
$\hat{\beta}_{10}$	-0.4317	-0.2634	-0.1871	-0.1998	-0.2728	-0.2869
$\hat{\beta}_{11}$	-	0.1057	0.0851	0.0559	0.0553	0.0591
$\hat{\beta}_{12}$	-	-0.0652	-0.0362	0.0021	0.0049	0.0033
$\hat{\beta}_{13}$	-	-0.0023	-0.0051	-0.0007	0.0026	0.0063
$\hat{\beta}_{14}$	-	-	0.0909	0.0604	0.0541	0.0491
$\hat{\beta}_{15}$	-	-	-0.0371	-0.0207	-0.0128	-0.0125
$\hat{\beta}_{16}$	-	-	-0.0118	-0.0022	-0.0028	-0.0033
$\hat{\beta}_{17}$	-	-	-	0.1311	0.1050	0.1010
$\hat{\beta}_{18}$	-	-	-	-0.0188	-0.0138	-0.0217
$\hat{\beta}_{19}$	-	-	-	-0.0066	-0.0035	-0.0042
$\hat{\beta}_{20}$	-	-	-	-	0.0605	0.0504
$\hat{\beta}_{21}$	-	-	-	-	-0.0168	-0.0188
$\hat{\beta}_{22}$	-	-	-	-	-0.0174	-0.0160
$\hat{\beta}_{23}$	-	-	-	-	-	0.0336
$\hat{\beta}_{24}$	-	-	-	-	-	0.0243
$\hat{\beta}_{25}$	-	-	-	-	-	0.0022

- Características comparativas.

	PAU	A	B	C	D	E
|Error máximo|	0.8823	0.6396	0.5809	0.4699	0.4438	0.4417
MSE	0.1071	0.0731	0.0605	0.0303	0.0228	0.0222

- Evolución de las estimaciones respecto al Target.

Target	PAU	A	B	C	D	E
1.8929	1.8746					
1.6352	1.6151	1.8700	1.8674	1.5050	1.5971	1.6006
1.3600	1.3601	1.3022	1.3905	1.3401	1.3183	1.3131
1.7522	1.4514	1.4457	1.3903	1.6008	1.6142	1.5816
2.2314	1.7390	2.0868	2.0820	2.3556	2.2978	2.2628
1.3119	1.4974	1.4841	1.1907	1.3823	1.5467	1.5475
1.8305	1.8075	1.7065	1.6984	1.9706	1.8683	1.8460
1.3684	1.0006	0.9953	1.1494	1.1291	1.2070	1.2067
2.0420	1.6824	1.9917	1.9262	2.0379	2.1942	2.2357
1.9457	1.5045	1.5689	1.7669	1.7912	1.8935	1.9108
1.3891	1.5243	1.3814	1.3437	1.3568	1.3498	1.3781
1.3946	1.5390	1.4405	1.4856	1.5394	1.4848	1.4683
1.2197	1.3999	1.2845	1.2966	1.3174	1.2785	1.2721
1.7606	1.5805	1.7421	1.7063	1.6798	1.6519	1.6302
1.3904	1.5895	1.4384	1.3728	1.4198	1.4223	1.3888
1.9820	1.9308	1.7798	1.7450	1.9739	2.0026	1.9317
1.4931	1.5852	1.6626	1.5759	1.6502	1.6290	1.5737
2.3467	1.9623	1.8783	1.9408	2.2064	2.1628	2.1413
1.7281	1.3841	1.4368	1.5395	1.8240	1.7308	1.7312
1.4727	1.5045	1.5665	1.5884	1.5770	1.5636	1.5541
3.0502	2.1679	2.4106	2.4693	2.5803	2.6064	2.6085
2.4565	1.9572	1.9929	1.9690	2.2817	2.3906	2.4275
1.7489	1.4142	1.6289	1.7701	1.9273	1.8693	1.8334
2.5043	1.8944	2.0484	1.9554	2.0798	2.2231	2.2617
1.7488	1.5988	1.5191	1.6695	1.9055	1.9782	1.9849
1.7209	1.6816	1.5373	1.7160	1.7964	1.7069	1.6872

- Evolución de los errores.

Target	PAU	A	B	C	D	E
1.8929	0.0183					
1.6352	0.0201	-0.2348	-0.2322	0.1302	0.0381	0.0346
1.3600	-0.0001	0.0578	-0.0305	0.0199	0.0417	0.0469
1.7522	0.3008	0.3065	0.3619	0.1514	0.1380	0.1706
2.2314	0.4924	0.1446	0.1494	-0.1242	-0.0664	-0.0314
1.3119	-0.1855	-0.1722	0.1212	-0.0704	-0.2348	-0.2356
1.8305	0.0230	0.1240	0.1321	-0.1401	-0.0378	-0.0155
1.3684	0.3678	0.3731	0.2190	0.2393	0.1614	0.1617
2.0420	0.3596	0.0503	0.1158	0.0041	-0.1522	-0.1937
1.9457	0.4412	0.3768	0.1788	0.1545	0.0522	0.0349
1.3891	-0.1352	0.0077	0.0454	0.0323	0.0393	0.0110
1.3946	-0.1444	-0.0459	-0.0910	-0.1448	-0.0902	-0.0737
1.2197	-0.1802	-0.0648	-0.0769	-0.0977	-0.0588	-0.0524
1.7606	0.1801	0.0185	0.0543	0.0808	0.1087	0.1304
1.3904	-0.1992	-0.0480	0.0176	-0.0294	-0.0319	0.0016
1.9820	0.0512	0.2022	0.2370	0.0081	-0.0206	0.0503
1.4931	-0.0921	-0.1695	-0.0828	-0.1571	-0.1359	-0.0806
2.3467	0.3844	0.4684	0.4059	0.1403	0.1839	0.2054
1.7281	0.3440	0.2913	0.1886	-0.0959	-0.0027	-0.0031
1.4727	-0.0317	-0.0938	-0.1157	-0.1043	-0.0909	-0.0814
3.0502	0.8823	0.6396	0.5809	0.4699	0.4438	0.4417
2.4565	0.4993	0.4636	0.4875	0.1748	0.0659	0.0290
1.7489	0.3347	0.1200	-0.0212	-0.1784	-0.1204	-0.0845
2.5043	0.6099	0.4559	0.5489	0.4245	0.2812	0.2426
1.7488	0.1500	0.2297	0.0793	-0.1567	-0.2294	-0.2361
1.7209	0.0393	0.1836	0.0049	-0.0755	0.0140	0.0337

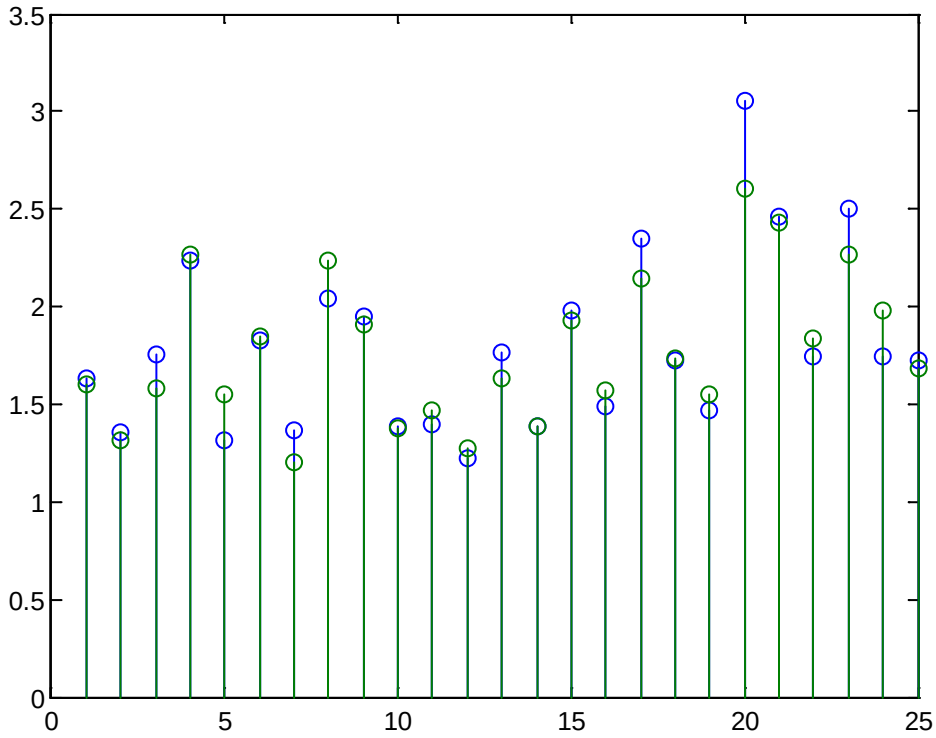
8.2.1.3.3. Conclusiones.

Especialmente interesante en el análisis de los resultados obtenidos es ver la disminución del MSE conforme se van introduciendo asignaturas de la carrera.

Por un lado, cabe destacar que el valor del MSE obtenido mediante regresión lineal múltiple (0.1071) es mayor que el obtenido en la mejor iteración de RNA para ajuste de funciones (0.0874). Por tanto, mediante RNA hemos sido capaces de ajustar mejor el comportamiento de las variables.

Sin embargo, con RNA no se han observado mejoras significativas y progresivas conforme se introducen asignaturas de la carrera, cosa que si ha sucedido en regresión.

Al introducir todas las asignaturas difíciles de la carrera se ha obtenido un MSE de 0.0222, valor que mejor ligeramente el 0.0298 de la mejor iteración de RNA, que además, variaba con la inicialización de los pesos. Veamos la comparación de los Targets con los Outputs de la regresión para ver la calidad de los resultados:



Vemos que en 10 de los 25 casos, el error es inferior a 0.05 puntos; en 16 de los 25 alumnos, el error cometido es inferior a 0.1 puntos; y en 24 de los 25 ejemplos, el error cometido es inferior a 0.25 puntos. Por tanto, los resultados obtenidos son bastante buenos.

8.2.2. Estudio del abandono en Ingeniería de Telecomunicaciones.

8.2.2.1. Clasificación a partir de variables PAU.

Procedemos a realizar el análisis previo de los datos, con el cálculo de algunos indicadores significativos:

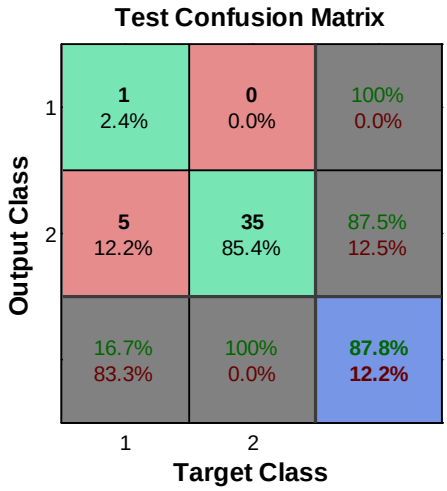
Variable	Media	Desviación típica
Nota PAU en la parte 1	6.9138	1.1122
Ranking parte 1/Total presentados	0.2737	0.2273
Nota PAU en la parte 2	6.7484	1.5689
Ranking parte 2/Total presentados	0.2727	0.2425

Calificación PAU	7.5495	1.0858
Ranking calificación PAU/Total presentados	0.2601	0.2416
Calificación física PAU	6.2684	2.2356
Ranking calificación física/Total presentados	0.3537	0.2639
Calificación Matemáticas PAU	7.2062	2.1323
Ranking calificación matemáticas/Total presentados	0.3337	0.2544

Las características PAU de los alumnos que ingresan en Telecomunicaciones son muy similares a las de los alumnos que deciden ingresar en Ingeniería Industrial.

8.2.2.1.1. No matriculados el segundo año.

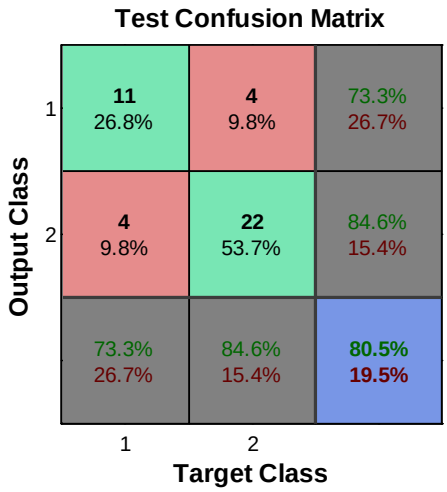
Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	275
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/20-2/20-3/20



Los resultados presentan las mismas características que para el caso de Ingeniería Industrial. Veamos qué sucede si lo que tratamos de predecir es el abandono oficial.

8.2.2.1.2. Abandono oficial.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	275
Variables consideradas	10
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/20



Al igual que para el caso de Ingeniería Industrial, los resultados mejoran significativamente.

Alumnos que abandonan y se predicen como no abandonos:

Alumno	PAU 1	R_PAU 1	PAU 2	R_PAU2	PAUT	R_PAUT	FIS	R_FIS	MAT	R_MAT
1	7,40	0,13	5,30	0,48	7,20	0,28	5,20	0,53	3,90	0,81
2	7,50	0,12	8,40	0,05	8,80	0,04	7,50	0,25	9,50	0,06
3	8,10	0,04	8,50	0,03	8,60	0,05	7,00	0,15	9,60	0,16
4	7,60	0,15	8,70	0,03	8,90	0,04	8,20	0,06	9,00	0,19

Alumno	Abandona	No abandona
1	0,2805	0,7661
2	0,0511	0,9233
3	0,109	0,879
4	0,0489	0,9087

A pesar de la certeza que presenta en general la red, lo alumnos abandonan.

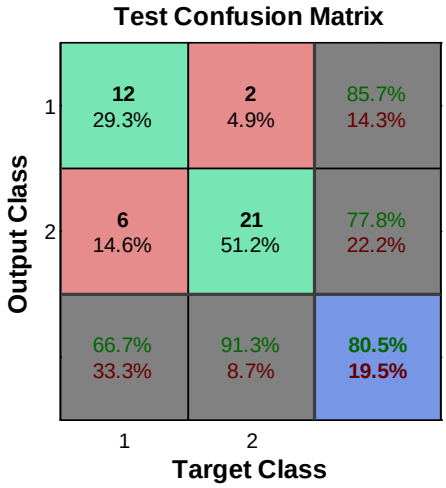
Alumnos que no abandonan y se predicen como abandonos:

Alumno	PAU 1	R_PAU 1	PAU 2	R_PAU2	PAU T	R_PAUT	FIS	R_FIS	MAT	R_MAT
1	5,60	0,61	6,00	0,36	6,80	0,41	1,70	0,87	9,20	0,10
2	6,70	0,29	5,80	0,35	7,50	0,20	4,80	0,57	7,90	0,19
3	7,10	0,22	5,90	0,38	5,80	0,74	4,80	0,42	7,00	0,41
4	7,50	0,12	7,30	0,16	6,90	0,37	7,10	0,31	8,80	0,18

Alumno	Abandona	No abandona
1	0,9879	0,0189
2	0,6796	0,3939
3	0,9450	0,1134
4	0,7240	0,3322

Otras arquitecturas:

Arquitectura: Capa oculta i/neuronas capa oculta i	1/30-2/30



8.2.2.2. Clasificación con variables de primer curso.

Veamos algunos indicadores de los datos:

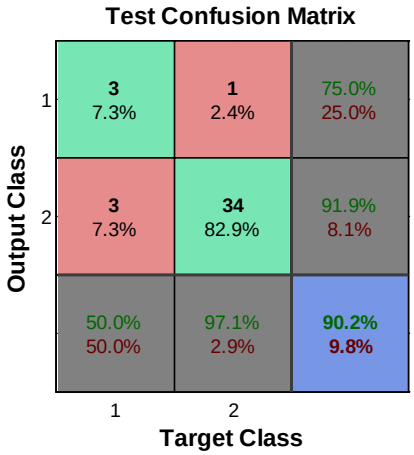
Variable	Media	Desviación típica
Créditos matriculados	63.1473	2.3878
Tasa evaluados	0.7461	0.2568
Tasa presentados	0.9060	0.3265
Tasa rendimiento	0.5343	0.3359
Tasa éxito	0.6568	0.3045
Número de convocatorias	16.2800	3.5285
Tasa NP	0.3661	0.2697
Tasa 0	0.2400	0.1587
Tasa 1	0.2310	0.1686
Tasa 2	0.1184	0.1553
Tasa 3	0.0310	0.0745
Tasa 4	0.0136	0.0547

Se trata de alumnos que de media, se presentan bastantes veces el primer año de matrícula, aunque la tasa de exámenes aprobados no llega, de media, ni al 40%.

8.2.2.2.1. Alumnos no matriculados el segundo año.

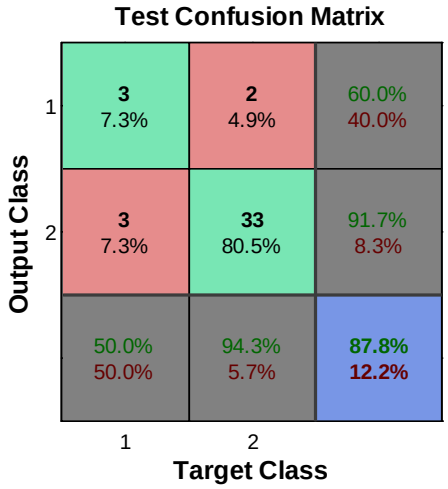
Consideramos tanto variables PAU como variables de primer curso.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	275
Variables consideradas	22
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25



Quitamos las variables PAU, dejando únicamente las de primer curso, ya que vimos en apartados anteriores la poca influencia de los resultados PAU.

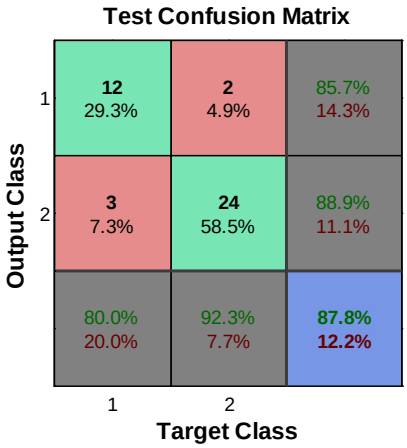
Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	275
Variables consideradas	12
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/20



El análisis que se desprende es que la inclusión de variables PAU ni mejora ni empeora los resultados, por lo que no influye en el abandono de primer curso. Las variables del primer año de matrícula tampoco clasifican bien a los alumnos. Destaca la tendencia de la red a clasificar a los alumnos como no abandono. Concluimos que con las variables consideradas, predecir el abandono de primer curso mediante redes neuronales artificiales para las titulaciones consideradas no es posible.

8.2.2.2.2. Abandono oficial.

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	275
Variables consideradas	22
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/25



Alumnos que abandonan y la red predice como no abandonos:

Alumno	1	2	3
Cred_matr	66.0000	61.5000	64.5000
Tasaevaluados	0.8182	1.0000	0.8140
Tasapresentados	0.7500	1.1951	1.0233
Tasarendimiento	0.6364	0.8049	0.4419
Tasaexito	0.7778	0.8049	0.5429
NumConv	16.0000	13.0000	17.0000
TasaNP	0.5000	0.0769	0.3529
Tasa0	0.1250	0.3077	0.4118
Tasa1	0.2500	0.5385	0.2353
Tasa2	0.1250	0.0769	0
Tasa3	0	0	0
Tasa4	0	0	0
PAU_1	7.3000	6.7000	8.1000
R_PAU_1	0.1791	0.2570	0.0353
PAU_2	6.6000	8.9000	8.5000
R_PAU_2	0.2433	0.0209	0.0330
PAU_T	7.8000	8.2000	8.6000
R_PAU_T	0.1774	0.0992	0.0461

FISICA	4.6000	9.4000	7.0000
R_FISICA	0.4427	0.0577	0.1523
MAT	7.8000	8.7000	9.6000
R_MAT	0.2625	0.1289	0.1550

Alumno	1	2	3
Abandona	0.3541	0.0432	0.2788
No abandona	0.8321	0.9714	0.7351

Se trata de alumnos con tasas de éxito, rendimiento y presentadas bastante buenas y con calificaciones y ranking PAU buenos también. En cambio, sus tasas 1, 2, 3, 4 no son especialmente buenas. La red muestra bastante certeza a la hora de clasificarlos como no abandonos aunque, finalmente, abandonan.

Alumnos que no abandonan y la red predice como abandonos:

Alumno	1	2
Cred_matr	61.5000	64.5000
Tasaevaluados	0.8049	0.5581
Tasapresentados	1.2683	0.8605
Tasarendimiento	0.1707	0.2093
Tasaexito	0.2121	0.3750
NumConv	19.0000	18.0000
TasaNP	0.3684	0.5556
Tasa0	0.5263	0.3333
Tasa1	0.1053	0.1111
Tasa2	0	0
Tasa3	0	0
Tasa4	0	0
PAU_1	6.4000	5.8000
R_PAU_1	0.3885	0.4740
PAU_2	4.7000	7.5000
R_PAU_2	0.6593	0.1299
PAU_T	6.1000	7.4000
R_PAU_T	0.6533	0.2502
FISICA	3.0000	7.2000
R_FISICA	0.6910	0.3019
MAT	5.3000	8.5000
R_MAT	0.6818	0.2165

Alumno	1	2
Abandona	0.7560	0.5691
No abandona	0.3168	0.1669

Estos dos alumnos tienen tasas de éxito y rendimiento muy inferiores a los otros tres señalados antes (en los que la red también se equivoca) a pesar de haberse presentado tantas o más veces que aquellos. Además en la PAU obtuvieron notas peores y su tasa de aprobados apenas ronda el 10%. Se entiende, por tanto, que la red los clasifique como abandonos, aunque finalmente no abandonen. La clasificación correcta de estos alumnos depende de variables no consideradas en este estudio relacionadas con su motivación, gustos, situación económica.

8.2.2.3. Clasificación con variables de segundo curso.

Los indicadores generales relacionados con estas variables son los siguientes:

Variable	Media	Desviación típica
Créditos matriculados	75.0536	12.3610
Tasa evaluados	0.8127	0.2085
Tasa presentados	1.0144	0.2676
Tasa rendimiento	0.6113	0.2960
Tasa éxito	0.7162	0.2636
Número de convocatorias	21.6607	5.4708
Tasa NP	0.3475	0.2555
Tasa 0	0.2314	0.1352
Tasa 1	0.2320	0.1412
Tasa 2	0.1417	0.1497
Tasa 3	0.0352	0.0705
Tasa 4	0.0122	0.0493

Comparación entre las variables del primer año de matrícula y el segundo año de matrícula:

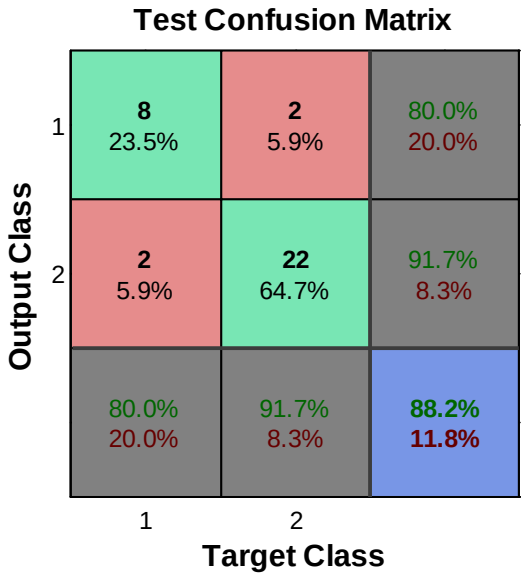
Variable	Media primero	Media segundo
Créditos matriculados	63.1473	75.0536
Tasa evaluados	0.7461	0.8127
Tasa presentados	0.9060	1.0144
Tasa rendimiento	0.5343	0.6113
Tasa éxito	0.6568	0.7162

Número de convocatorias	16.2800	21.6607
Tasa NP	0.3661	0.3475
Tasa 0	0.2400	0.2314
Tasa 1	0.2310	0.2320
Tasa 2	0.1184	0.1417
Tasa 3	0.0310	0.0352
Tasa 4	0.0136	0.0122

Vemos que en segundo, los alumnos tienden a matricularse de 10 créditos más que en primer curso aproximadamente. El resto de indicadores son, de media, bastante parecidos, si bien es cierto que el número de convocatorias a las que puede presentarse el alumno es superior en segundo que en primero. Este hecho está motivado seguramente porque se han matriculado de más créditos.

Entrenamos la red:

Función de entrenamiento.	Trainscg
Ejemplos de entrenamiento	224
Variables consideradas	34
Conjuntos: Train/Validation/Set (%)	70/15/15
Arquitectura: Capa oculta i/neuronas capa oculta i	1/35



Los resultados que da la red son especialmente buenos, con una tasa de acierto que ronda el 80% para alumnos que abandonan y un 92% para alumnos que no abandonan. Analicemos los casos en los que la red se equivoca:

Alumnos que abandonan y la red predice como no abandonos:

	Alumno	1	2
Primero	Cred_matr	61.5000	61.5000
	Tasaevaluados	0.9024	1.0000
	Tasapresentados	1.0732	1.1951
	Tasarendimiento	0.4390	0.8049
	Tasaexito	0.4865	0.8049
	NumConv	16.0000	13.0000
	TasaNP	0.3125	0.0769
	Tasa0	0.4375	0.3077
	Tasa1	0.2500	0.5385
	Tasa2	0	0.0769
	Tasa3	0	0
	Tasa4	0	0
Segundo	Cred_matr	75.0000	90.0000
	Tasaevaluados	0.9200	0.8167
	Tasapresentados	1.3400	0.7833
	Tasarendimiento	0.7800	0.4833
	Tasaexito	0.8478	0.5918
	NumConv	23.0000	22.0000
	TasaNP	0.2609	0.4545
	Tasa0	0.3043	0.2727
	Tasa1	0.2609	0.2273
	Tasa2	0.1304	0.0455
	Tasa3	0	0
	Tasa4	0.0435	0
PAU	PAU_1	7.8000	6.7000
	R_PAU_1	0.0834	0.2570
	PAU_2	7.2000	8.9000
	R_PAU_2	0.1157	0.0209
	PAU_T	7.9000	8.2000
	R_PAU_T	0.1437	0.0992
	FISICA	7.2000	9.4000
	R_FISICA	0.2009	0.0577
	MAT	8.1000	8.7000
	R_MAT	0.1717	0.1289

Alumno	1	2
Abandona	0.0007	0.0117
No abandona	0.9991	0.9902

La salida de la red muestra mucha seguridad al clasificar como no abandonos a estos alumnos. Se trata de alumnos con calificaciones PAU muy buenas. En primero, el alumno 1 tiene tasas de éxito y rendimiento por debajo de la media; en cambio, en segundo remonta, obteniendo tasas de éxito por encima de la media y tasas 1 (aprobado), 2 (notable) y 4 (matrícula de honor) en la media o incluso por encima de ella. Por ello la red lo clasifica como no abandono. El segundo alumno es bastante similar al anterior, si bien es cierto, que a pesar de obtener en su primer año de matrícula tasas de éxito y rendimiento por encima de la media, en segundo obtiene peores tasas en general.

Alumnos que no abandonan y la red predice como abandonos:

	Alumno	1	2
Primero	Cred_matr	61.5000	66.0000
	Tasaevaluados	0.5122	0.8409
	Tasapresentados	0.6585	0.9318
	Tasarendimiento	0.0732	0.8409
	Tasaexito	0.1429	1.0000
	NumConv	19.0000	15.0000
	TasaNP	0.6316	0.3333
	Tasa0	0.3158	0.1333
	Tasa1	0.0526	0.2000
	Tasa2	0	0.3333
	Tasa3	0	0
	Tasa4	0	0
Segundo	Cred_matr	57.0000	67.5000
	Tasaevaluados	0.8947	0.8222
	Tasapresentados	1.1579	1.0444
	Tasarendimiento	0.3684	0.3778
	Tasaexito	0.4118	0.4595
	NumConv	24.0000	22.0000
	TasaNP	0.5417	0.4091
	Tasa0	0.3333	0.3636
	Tasa1	0.0417	0.1818
	Tasa2	0.0833	0.0455
	Tasa3	0	0
	Tasa4	0	0

PAU	PAU_1	5.4000	8.6000
	R_PAU_1	0.6578	0.0164
	PAU_2	4.6000	6.5000
	R_PAU_2	0.6773	0.2871
	PAU_T	7.0000	7.4000
	R_PAU_T	0.3425	0.2377
	FISICA	3.9000	7.4000
	R_FISICA	0.5548	0.3448
	MAT	5.4000	5.9000
	R_MAT	0.6591	0.6181

Alumno	1	2
Abandona	0.9408	0.6542
No abandona	0.0945	0.2898

Comparados con los dos alumnos que la red clasifica erróneamente como no abandonos, estos alumnos tienen unas calificaciones PAU peores, especialmente en matemáticas y física. El alumno 1 obtiene tasas de éxito, rendimiento y aprobado (1) muy por debajo de la media tanto en primero como en segundo. El alumno 2, en su primer año de matrícula obtiene unos resultados buenos, por encima de la media. En cambio en segundo su rendimiento cae bastante, de ahí que la red lo clasifique como abandono aunque el alumno finalmente no abandone. Destacamos de nuevo que los errores de la red son, en cierto modo, entendibles por la influencia de variables no consideradas en este este estudio: motivación, gustos, situación económica.

8.2.2.4. Influencia de las variables PAU en clasificación de abandono.

TELECOMUNICACIONES PAU			
	0	1	Total
Todas las variables	82.8	75.0	80.5
Todas menos matemáticas	79.3	66.7	75.6
Todas menos física	82.8	58.3	75.6
Todas menos PAU	79.3	58.3	73.2
Todas menos PAU1	75.9	66.7	73.2
Todas menos PAU2	82.8	75.0	80.5
Todas menos notas	79.3	41.7	68.3
Todas menos clasificación	79.3	75.0	78.0

Sólo matemáticas	86.2	16.7	65.9
Sólo física	82.8	50.0	73.2
Sólo PAU	75.9	58.3	70.7
Sólo PAU1	72.4	58.3	68.3
Sólo PAU2	62.1	58.3	61.0

Vemos que con matemáticas como única variable de entrada, los resultados para la predicción de abandonos son bastante malos (16.7%). La red tiende a clasificar a la mayoría de los alumnos como no abandonos (35 de 41) acertando, lógicamente, un 86.2% de los no abandonos. Lo mismo ocurre con física. Ella, por sí sola, únicamente puede predecir el 50% de los abandonos.

Algo similar ocurre con PAU 2. La principal diferencia estriba en que PAU 2 por si sola únicamente predice bien el 62.1% de los no abandonos y el 58.3 de los abandonos. Quitarla del conjunto de entrada incluso mejora los resultados. La variable PAU 1 tiene un efecto parecido. Por si sola tiene una tasa de acierto aceptable para no abandono (75.9%) pero bastante baja para abandono (58.3%). Y lo mismo ocurre con la variable PAU.

La principal variable en la predicción son las notas ya que al dejar como única variable de entrada las notas (PAU1, PAU2, PAU, física y matemáticas) los resultados son casi tan buenos como los obtenidos con todas las variables, si bien es cierto que empeoran desde un 82.8 % hasta un 79.3 % para la predicción de no abandonos. Quitar las notas como variable de entrada (dejando únicamente los ranking) hace que el acierto en la predicción de abandonos caiga hasta un 41.7%.

8.2.2.5. Influencia de las variables de primer curso en clasificación de abandono.

TELECOMUNICACIONES PRIMERO.			
	0	1	Total
Todas variables	93.1	91.7	92.7
Todas menos creditosmatriculados	93.1	91.7	92.7
Todas menos numconv	93.1	91.7	92.7
Todas menos tasa0	93.1	91.7	92.7
Todas menos tasa1	93.1	83.3	90.2
Todas menos tasa2	93.1	91.7	92.7
Todas menos tasa3	93.1	91.7	92.7
Todas menos tasa4	93.1	83.3	90.2
Todas menos tasaevaluados	93.1	91.7	92.7

Todas menos tasaexito	93.1	91.7	92.7
Todas menos tasaNP	93.1	83.3	90.2
Todas menos tasapresentados	93.1	91.7	92.7
Todas menos tasarendimiento	93.1	91.7	92.7
Sólo creditosmatriculados	100.0	0.0	70.7
Sólo numconv	93.1	100.0	95.1
Sólo tasa0	89.7	25.0	70.7
Sólo tasa1	89.7	91.7	90.2
Sólo tasa2	82.8	33.3	68.3
Sólo tasa3	100.0	0.0	70.7
Sólo tasa4	100.0	0.0	70.7
Sólo tasaevaluados	72.4	75.0	73.2
Sólo tasaexito	86.2	100.0	90.2
Sólo tasaNP	89.7	66.7	82.9
Sólo tasapresentados	89.7	58.3	80.5
Sólo tasarendimiento	93.1	83.3	90.2

Por un lado vemos lo que ocurre con variables como Tasa0, Tasa2, Tasa3, Tasa4 y créditos matriculados. Se trata de variables que por sí solas (utilizadas independientemente como únicas variables de entrada) son incapaces de encontrar entre los datos de entrenamiento y validación seleccionados patrones que predigan con un éxito aceptable los alumnos que van a abandonar y los que no. Clasifican la mayor parte de los alumnos como no abandono, de ahí las altas tasas de acierto en este aspecto. Además, quitarlas del conjunto de entrada (es decir, utilizar como inputs todas las variables excepto alguna de las mencionadas), no supone perjuicios graves en lo que a tasa de acierto se refiere para la red.

Por otro lado tenemos el caso de la variable tasa evaluados, que por sí sola es capaz de acertar el 72.4% de los alumnos que no abandonan y el 75% de los alumnos que abandonan. Sin embargo, quitarla de la red no implica perjuicio alguno por la presencia de variables que por sí solas explican muy bien los datos y que se mencionarán más adelante. En la misma línea están las variables TasaNP y Tasapresentados. Empleadas como única variable de entrada consiguen obtener un buen porcentaje de acierto para la predicción de no abandonos; ello está motivado porque la mayor parte de los ejemplos de test son de alumnos que no abandonan (29 sobre 41), lo que implica que un fallo en este apartado no suponga porcentualmente tanto como un fallo para el caso de abandono. Por tanto, en los abandonos consiguen tasas de acierto bastante menores y que no superan el 70%, aun siendo el número de fallos cometidos para abandono y no abandono muy similar o igual.

Finalmente, tenemos el caso de variables que utilizadas como única variable de entrada son capaces de explicar por sí mismas los patrones de abandono y no abandono con tasas de acierto similares (y en algunos casos incluso superiores) a las obtenidas utilizando en el conjunto de entrada las doce variables de primero. Es el caso de: número de convocatorias, tasa rendimiento, tasa 1 y tasa éxito. Especial mención merece la variable asociada al número de convocatorias a las que un alumno podría presentarse. Por si sola, es capaz de predecir el 93.1 por ciento de los no abandonos y el 100% de los abandonos. De hecho, de los 41 ejemplos de test, sólo se equivoca en dos alumnos que predice como abandonos y que finalmente no abandonan.