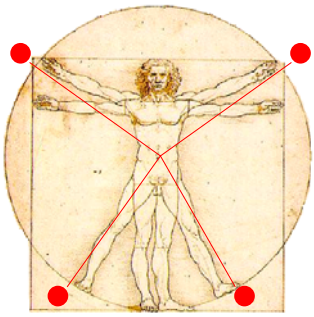
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN IV . AÑO 2006

SEPARATA



INTERFAZ VISUAL PARA EL PROTOTIPADO RÁPIDO DE CLASIFICADORES DE GAJOS DE MANDARINA BASADOS EN REDES NEURONALES

Basil M. Al-Hadithi, Manuel R. Rueda, Juan S. Muro

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior

Villanueva de la Cañada (Madrid)

ISSN: 1696-8085
Editor: Julio Merino García tecnologia@uax.es

INTERFAZ VISUAL PARA EL PROTOTIPADO RÁPIDO DE CLASIFICADORES DE GAJOS DE MANDARINA BASADOS EN REDES NEURONALES

Basil M. Al-Hadithi[a], Manuel R. Rueda[b], Juan S. Muro[c]

[a)]Dr Ing. Industrial,
Area de tecnologías de información y comunicaciones, Escuela Politécnica Superior, Universidad Alfonso X el Sabio. Avda. De la Universidad nº1, Villanueva de la Cañada, 28691 Madrid. España.
Tlf.:918109234, email: bmal@uax.es

[b)]Ing. de Telecomunicación,
rmanuelroldan@hotmail.com

[c)]Dr Ing. Industrial,
Departamento de Tecnología Electrónica, Escuela Técnica Superior de Ingenieros Industriales, Universidad Politécnica de Cartagena (Murcia).
Campus Moralla del Mar, 30202, Cartagena. España. Tlf.:968325380, email: Juan.Suardiaz@upct.es

RESUMEN: Este trabajo presenta una herramienta visual desarrollada sobre el entorno Matlab dedicada al control de calidad del proceso de envasado de gajos de mandarina en conserva a partir de imágenes extraídas de una cinta transportadora ubicada en un proceso industrial. El entorno desarrollado permite establecer una clasificación con índices de calidad de gajos, ofreciendo una emulación en cierto modo de un sistema de lógica difusa, aproximándolo a dicha teoría. Además de esto, la aplicación desarrollada se muestra como una plataforma multiclasificadora debido a las múltiples opciones que ofrece, lo que permitiría utilizarla para cualquier tipo de aplicación de reconocimiento de patrones, independientemente de la naturaleza de los mismos.

PALABRAS CLAVE: Reconocimiento de patrones, Redes neuronales, Visión artificial, Matlab

ABSTRACT: *This work describes a visual interface developed using Matlab, which can be utilized for quality control purposes within an tangerine segment classification from images acquired from a moving conveyor belt in an industrial process. This environment offers a classification with quality factors, aproximating the fuzzy logic theory. Moreover the developed application turns up as a multiclassification platform due to its numerous options, which will allow applying it for any type of pattern recognition task, regardless of its kind.*

KEYWORDS: Pattern recognition, Neural networks, Artificial vision, Matlab

1. Introducción

La inspección visual mediante operadores humanos es un tipo de inspección que utilizan algunas empresas españolas dedicadas a la comercialización de frutas en conserva. En el caso que se describe, gajos de mandarina en conserva, es habitual encontrar un conjunto de 4 a 6 inspectores a ambos lados de una cinta transportadora que se mueve con una velocidad de 2m/min, de la que tendrán que retirar los gajos de mandarinas que consideren no válidos, tal y como puede apreciarse en la figura 1.

Figura 1: Inspección basada en inspectores humanos.

La inspección basada en operadores humanos presenta dos desventajas importantes: una falta de homogeneidad en la inspección y una tasa importante de errores de clasificación asociados al cansancio (Davidson et al., 1999). Mair (Mair, 1998) afirma que el 10% de los costes de producción son debidos a fallos de los inspectores humanos, Smith (Smith, 1993) calculó que la eficiencia de inspección que ofrecen los operarios suele rondar el 80% y Poltzeitner & Schwingshakl (Poltzeitner et al., 1992) dan un valor del 55% si la inspección se lleva a cabo en sistemas que demanden una clasificación rápida. Todo esto hace que actualmente los empresarios traten de buscar alternativas automatizadas en estos procesos de inspección.

En el caso de las conservas de frutas, la automatización del proceso de inspección no es una tarea trivial. Las normativas asociadas a la producción de conservas, como pueden ser los estándares de calidad para la exportación de productos en conservas, establecen categorías comerciales de calidad para la producción de gajos de mandarinas en conserva. Cada nivel de calidad está delimitado por un porcentaje máximo de gajos rotos y defectos en el pelado. En el caso de los gajos de mandarinas surge una complejidad adicional debido a que el contorno de los gajos buenos presenta una

variabilidad natural. Además, su forma también es variable, lo que complica el proceso de desarrollo de un sistema automático de inspección. El objetivo abordado en el presente trabajo es mejorar el proceso de inspección actual desarrollando un sistema automatizado que permita asegurar un funcionamiento en línea dentro del propio proceso de producción.

Este tipo de procesos automatizados ya han sido abordados con anterioridad mediante distintos tipos de clasificadores con mayor o menor éxito, incluyendo las redes neuronales, pero esta vez se propone una revisión del sistema. Se ha escogido Matlab (MathWorks, 2006), programa ampliamente extendido, cuyos requerimientos no son elevados, de forma que cualquier pequeña planta podría utilizarlo en caso de necesitar cambiar algún algoritmo y a la par aprovechar la posibilidad de compilar el desarrollo y generar un ejecutable que pueda ser utilizado en cualquier máquina sin necesidad de tener Matlab instalado. Además, se ha desarrollado un sistema que permite catalogar los gajos no sólo como buenos o malos, sino que incluso permite dar una indicación sobre la calidad del gajo, de forma que se puedan establecer diferentes categorías que se correspondan con distintas utilidades de los gajos, variando desde una calidad de presencia óptima, adecuada para aplicaciones en pastelería y hostelería, hasta aplicaciones en las que el aspecto no sea tan importante, como pueda ser el mercado de zumos.

2. Redes Neuronales

Las redes neuronales han sido una novedad en los últimos años y se han estado empleando en numerosas aplicaciones. Entre sus principales ventajas, se menciona el hecho de no ser lineales, y de no requerir un modelo fenológico para describir la distribución de los datos, ya que pueden aprender a partir de muestras y permiten implementar de forma sencillo algoritmos de discriminación basados en funciones no lineales si fura necesario.

Las Redes Neuronales surgieron del movimiento conexionista, que nació junto con la IA (Inteligencia Artificial) simbólica o tradicional. Esto ocurrió hacia los años 50, con algunos de los primeros ordenadores de la época y las posibilidades que ofrecían. Esta teoría intenta representar el conocimiento desde el estrato más básico de la inteligencia: el estrato físico. Se apoya en la teoría biológica neuronal: el secreto para el aprendizaje y el conocimiento se halla directamente relacionado con la estructura del cerebro, concretamente con las neuronas y la interconexión entre ellas.

Las principales ideas, y por lo tanto el vocabulario y la notación, se ha heredado de múltiples disciplinas como la teoría de decisión estadística (Chow, 1957), la teoría de interruptores (Winder, 1963), la teoría de los autómatas (Pask, 1962), la teoría de control (Widrow, 1964), análisis lingüística (Ledley, 1964), la teoría de la información (Kamentsky, 1964), la programación matemática (Rosen, 1965), los estudios nerviosos y teoría de redes neuronales (Rosenblatt, 1957), la teoría de lógica difusa (Georgiev, 1987) y la teoría genética (Fisher, 1930).

En la actualidad, se están mezclando también con los denominados clasificadores borrosos (fuzzy). Todos aquellos actos en los que interviene el ser humano se hallan normalmente regidos por procesos lógicos de inferencia con un alto grado de incertidumbre. Es por ello que se ha tratado de reformular la lógica bivalente clásica, donde las proposiciones sólo pueden ser verdaderas o falsas por otros sistemas de lógica más generales que puedan incluir diferentes grados de verdad o certidumbre en las proposiciones. Entre este nuevo tipo de sistemas de lógica se encuentra la denominada lógica borrosa o "fuzzy". Esta nueva Lógica, fue inicialmente analizada por el lógico polaco Lukasiewicz, con sus estudios sobre lógica multivaluada. Posteriormente, Lofti Zadeh (Zadeh, 1965), desde una perspectiva procedente básicamente del campo de la teoría de control de sistemas y la inteligencia artificial, enunció la denominada lógica borrosa o difusa (fuzzy logic). Zadeh plantea un nuevo tipo de conjuntos, denominado conjuntos borrosos a los cuales sus elementos pueden pertenecer en cierto grado desde 0, que implica la exclusión total de ese elemento a dicho conjunto, hasta 1, que supone la pertenencia total. Por tanto, cualquier proposición que enuncia la pertenencia de un determinado elemento a un conjunto borroso será cierta o falsa en la misma medida en que ese elemento pertenezca o no a él. Toda esta teoría ha generado un nuevo marco de desarrollo de calificadores, que si bien no son objeto del presente trabajo, sí constituyen un interesante punto de vista que actualmente está generado líneas de investigación que combinen las técnicas de clasificación mediante redes neuronales con técnicas de inferencia borrosa.

A continuación se introducen las bases fundamentales de los diferentes tipos de redes neuronales que han constituido la base del trabajo desarrollado.

2.1. *Backpropagation*

Fue desarrollado por David Rumelhart y Geoffrey Hinton (Rumelhart et al., 1986). Este algoritmo utiliza una función de error asociada a la red, buscando el estado estable de mínima energía o de mínimo error a través del camino descendente de la superficie del

error. Por ello realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

El método que sigue la regla delta generalizada para ajustar los pesos es exactamente el mismo que el de la regla delta utilizada en ADALINE (Widrow y Hoff, 1960). Los pesos se actualizan de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida (error).

El desarrollo del algoritmo se hará para una red con una capa oculta y después se generalizará para redes con más de una capa oculta. Cuando se presenta a la entrada un patrón de entrenamiento, éste se propaga a través de las conexiones existentes produciendo una entrada neta n en cada una de las neuronas de la siguiente capa. La entrada neta a la neurona j de la siguiente capa debido a la presencia de un patrón de entrenamiento en la entrada es el mostrado en la ecuación (1).

$$n_j^0 = \sum_{i=1}^{p} w_{ji}^0 u_i + b_j^0 \tag{1}$$

donde ω^0_{ji} es el peso que une la componente i de la entrada con la neurona j de la primera capa oculta, u_i es la componente i del vector U de entrada y b^0j es el umbral (ganancia) de la neurona j de la capa oculta. El superíndice 0 representa la capa a la que pertenece cada parámetro, en este caso la capa oculta.

La salida de cada neurona de la capa oculta es la mostrada en la ecuación (2).

$$a_j^0 = f^0 \left(\sum_{i=1}^{p} w_{ji}^0 u_i + b_j^0 \right) \tag{2}$$

donde f^0 es la función de activación de las neuronas de la capa oculta. Las salidas s^0j de las neuronas de la capa oculta son las entradas de las neuronas de la capa de salida, ecuación (3).

$$n_k^s = \sum_{j=1}^{m} w_{kj}^s a_j^0 + b_k^s \tag{3}$$

donde ω^s_{kj} es el peso que une la neurona j de la capa oculta con la neurona k de la capa de salida, que tiene ṣ neuronas, a^0_j es la salida de la neurona j de la capa oculta, que tiene ṃ neuronas, b^s_k es el umbral de la neurona k̩ de la capa de salida y n^s_k es la entrada neta a la neurona k de la capa de salida.
La red produce una salida final descrita por la ecuación (4).

$$a_k^s = f^s\left(n_k^s\right) = f^s\left(\sum_{j=1}^{m} w_{kj}^s a_j^0 + b_k^s\right) \tag{4}$$

La salida de la red de cada neurona s^s_k se compara con la salida deseada y_k para calcular el error en cada unidad de salida, ecuación (5).

$$\varepsilon_k = \left(y_k - a_k^s\right) \tag{5}$$

El error medio cuadrado debido a cada patrón de entrada se puede expresar como el sumatorio de la ecuación (6).

$$\varepsilon_k = \left(\varepsilon_k^2\right) = \frac{1}{n}\sum_{k=1}^{n} \varepsilon_k^2 \tag{6}$$

Este proceso se repite para cada patrón de entrenamiento (N veces). Para que el proceso de aprendizaje sea adecuado, el algoritmo debe actualizar todos los pesos y umbrales de la red minimizando el error medio cuadrado total, ecuación (7).

$$e^2 = \sum_{p=1}^{N} e_p^2 \tag{7}$$

El error que produce una red neuronal en función de sus pesos genera un espacio de n dimensiones, siendo n el número de pesos de conexión de la red. Al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la que la función del error tendrá un mayor crecimiento. Por lo tanto, deberá tomarse la dirección negativa del gradiente para obtener el mayor decrecimiento de la función y así poder encontrar su punto mínimo. Una vez que se tiene el gradiente de la función de error se puede proceder a la actualización de todas las capas de la red empezando por la de salida. Para la capa de salida la actualización de pesos y umbrales viene dada por las ecuaciones (8) y (9), respectivamente.

$$w_{kj}(t+1) = w_{kj}(t) - 2\alpha\varepsilon_k^s \tag{8}$$

$$b_k(t+1) = b_k(t) - 2\alpha\varepsilon_k^s \tag{9}$$

Después de actualizar los pesos y umbrales de la capa de salida se procede a la actualización de los respectivos de la capa oculta, ecuaciones (10) y (11).

$$w_{ji}(t+1) = w_{ji}(t) - 2\alpha\varepsilon_j^0 u_i \tag{10}$$

$$b_j(t+1) = b_j(t) - 2\alpha\varepsilon_j^0 \tag{11}$$

Como puede observarse, el algoritmo Backpropagation utiliza la misma técnica de aproximación en pasos descendientes que emplea el algoritmo LMS (*Least Mean Square*). La única complicación está en el cálculo del gradiente que es indispensable para realizar la propagación de la sensibilidad.

Este algoritmo de aprendizaje, con algunas variantes, es el más utilizado y quizá al mismo tiempo el más criticado por su extremada lentitud, su sensibilidad a los parámetros (ganancia, inicialización aleatoria) y la falta de garantías acerca de la obtención de un resultado óptimo (Fahlman, 1988). En efecto, el método de descenso por gradiente sólo garantiza la obtención de un mínimo local de la función de error, debido a que sólo se tiene información local de la superficie del error y no se sabe lo lejos o cerca que se está del punto mínimo. Por ello, la elección de la tasa de aprendizaje tiene gran importancia. Es conveniente avanzar con incrementos pequeños, ya que en otro caso se corre el riesgo de pasar por encima del punto mínimo y oscilar alrededor de él sin converger. Con incrementos demasiado pequeños la velocidad de convergencia disminuye mucho aunque se asegura alcanzar el mínimo.

Otra crítica frecuente al algoritmo de Backpropagation es la de que no se trata de un algoritmo local. Un algoritmo local es aquél en el cual el proceso que se ejecuta en cada unidad sólo depende de sus entradas y salidas objetivo actuales y de los datos almacenados en su memoria.
En los últimos años han aparecido numerosas variantes de este método de entrenamiento. Se pueden encontrar descripciones extensas y detalladas en (Widrow y Lehr, 1990) y (Yu et al., 1995).

3.2. *Redes de Base Radial*

Las redes neuronales con funciones de base radial (RBF, *Radial Basis Functions*) muestran una estructura básica muy similar al Perceptrón multicapa, con una capa de entrada, una única capa oculta (en el Perceptrón multicapa pueden ser varias), y una capa de salida, como muestra la figura 2. La principal diferencia entre ambas arquitecturas radica en las funciones de activación utilizadas. En el Perceptrón se utilizan funciones de activación globales, normalmente sigmoides, que provocan problemas en términos de velocidad de convergencia y consecución de una solución global única al problema de optimización, puesto que la técnica implica la resolución de un problema no lineal con mínimos locales.

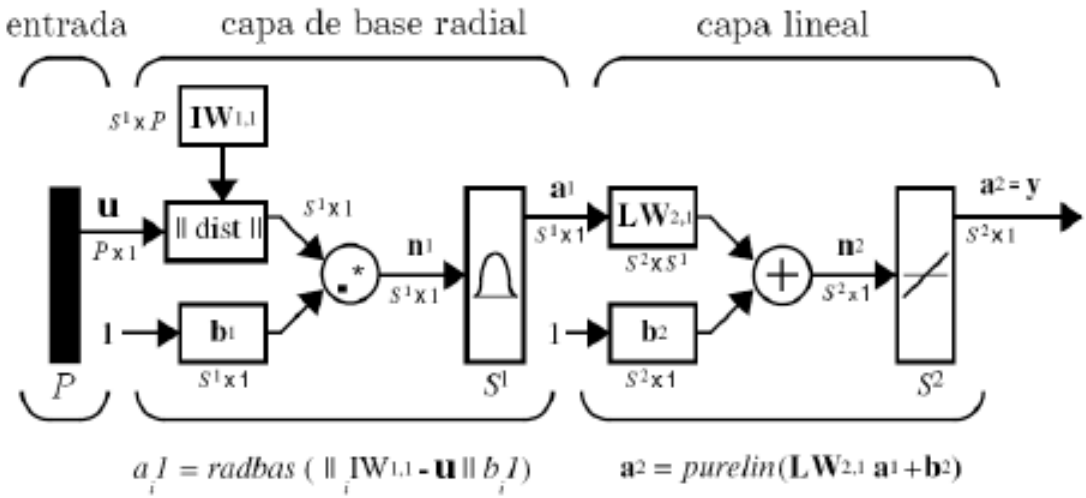


Figura 2: Topología de una RBF.

Las redes RBF se basan en funciones de activación locales, normalmente de tipo Gaussiano. Existe una base teórica sobre los problemas de representación y convergencia de estas redes en (Georgiev, 1987). Se ha demostrado que este tipo de redes presentan la propiedad de la mejor aproximación (Best approximation property) (Girosi, 1990).

Se trata de un método de interpolación (Powell, 1987), es decir que el resultado obtenido es exacto para las muestras de aprendizaje. El modelo se basa en definir un espacio lineal de funciones, de dimensión N (número de muestras de entrenamiento), en el que se escoge como base el conjunto de funciones $\{f(\| u - W_{ik} \|) : 1 \leq i \leq N\}$ donde W_i ($iW_{1,1}$) es el i-esimo vector fila de la matriz de pesos y contiene los centros de las funciones de las S1 neuronas de la capa oculta para cada componente de la entrada.
Los pesos W_i asociados a cada nodo en la capa oculta constituyen el centro del nodo en el espacio de entrada a la red. Cuando se propagan las entradas hacia adelante, cada nodo en esta capa calcula la distancia Euclídea entre su centro y el vector de entrada (u_1, u_2..., u_n) y el resultado se pasa a través de una función de activación local no lineal, normalmente la función gaussiana, para obtener el valor de activación para cada nodo, ecuaciones (12) y (13).

$$a_j = f(\|u - W_i\|, \sigma_i) \tag{12}$$

$$f(x, \sigma) = e^{-\frac{x^2}{\sigma^2}} \tag{13}$$

Estos valores se normalizan para toda la capa oculta y se obtiene el valor de salida para cada nodo de la misma, ecuación (14).

$$h_j = \frac{a_j}{\sum_{k=1}^{S^1} a_k} \tag{14}$$

Cada nodo en la capa de salida calcula su valor como la suma ponderada de los valores proporcionados por la capa oculta, ecuación (15) (Wendland, 2002).

$$y_i = \sum_{j=1}^{S^1} w_{ij} h_j \tag{15}$$

donde S^1 es el número de neuronas de la capa de base radial y S^2 es el número de neuronas de la capa lineal de salida.

El algoritmo de aprendizaje para los pesos de la última capa suele estar basado en el error en la salida de una forma similar al algoritmo de retropropagación (Backpropagation) del error. Sin embargo, para la capa oculta es diferente, y dependerá de la forma en que se generen y distribuyan inicialmente los nodos de dicha capa.

A las ventajas ya citadas de este tipo de redes hay que añadir que no presentan el problema de olvido del Perceptrón multicapa. Puesto que cada neurona tiene una influencia local en la función global generada, la adquisición de nuevo conocimiento relativo a una determinada zona del espacio de entrada a la red tiene poca influencia sobre lo ya aprendido en otra zona. Esa misma naturaleza local de las funciones de activación permite velocidades de aprendizaje muy superiores a las del Perceptrón multicapa. Esto las hace muy adecuadas para ser utilizadas con técnicas que requieran un aprendizaje online. Como inconvenientes se debe citar su peor capacidad de generalización y extrapolación respecto de los sistemas basados en Perceptrón multicapa, lo cual es motivado, como es evidente, por esa naturaleza local de las funciones de activación.

3.3. *Red de Elman*

La red de Elman típicamente posee dos capas, pero difiere de la red convencional de dos capas en que la capa oculta tiene una realimentación desde su salida a su entrada. Esta realimentación, que es posible apreciar con más detalle en la figura 3, permite a la red aprender a reconocer y generar patrones temporales o variantes en el tiempo (Elman, 1990). El retardo de la conexión almacena valores del instante de tiempo anterior, que

pueden ser utilizados en el instante actual. Esta característica provoca además que los resultados en distintas simulaciones puedan ser distintos. Concretamente, si dos redes de Elman con los mismos pesos y umbrales se alimentan con la misma entrada, pueden producir salidas diferentes con las mismas iteraciones, debido a que pueden presentar diferentes estados de realimentación.

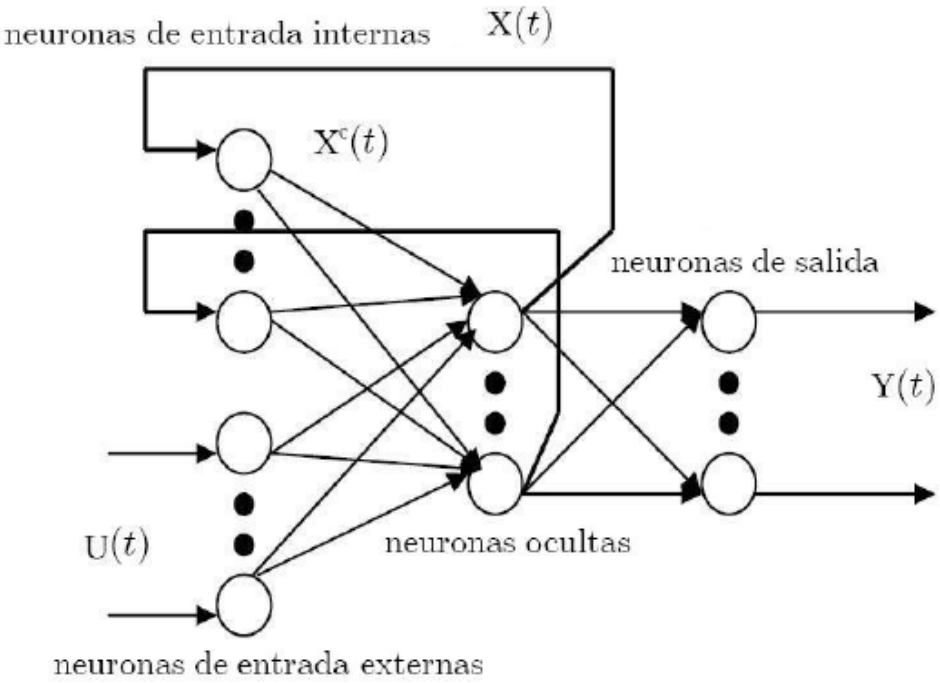


Figura 3: Estructura de la red de Elman.

Normalmente las funciones de activación de la capa recurrente suelen ser de tipo tangente hiperbólica (‘tansig’) y las de la capa de salida de tipo lineal (‘purelin’)(Matlab Help, 2006). Esta configuración, como ya se ha comentado, permite a la red aproximar cualquier función (con un número finito de discontinuidades) con la precisión deseada. La única condición es que la capa oculta debe tener suficientes neuronas.
Si la función que se necesita aproximar aumenta de complejidad será necesario incrementar el número de neuronas.
Debido a la similitud de la estructura de la red de Elman con las redes de propagación hacia adelante de dos capas, esta red puede entrenarse con cualquier variante del algoritmo de retropropagación del error como son el Levenberg-Mardquardt o el del gradiente conjugado El entrenamiento de la red puede resumirse en los siguientes pasos (Elman, 1993), (Pham y Liu, 1995):

- Presentar los patrones de entrenamiento a la red, calcular la salida con los valores iniciales de los pesos, comparar la salida con los patrones objetivo y calcular la secuencia de error.
- Propagar inversamente el error para encontrar el gradiente del error para cada conjunto de pesos y ganancias.
- Actualizar todos los pesos y ganancias con el gradiente calculado según el algoritmo de retropropagación.

Uno de los principios sobre los que se apoya la teoría de redes neuronales es que cualquier red neuronal, sea cual sea su estructura, tamaño, complejidad y potencia, tiene su equivalente en una red de una sola capa. Lo que ésta teoría no proporciona son las herramientas para encontrarla, ni siquiera asegura que sea posible encontrarla.
Debido a esto, surge un problema muy importante: generalmente, una red es tanto mejor cuantas más neuronas y capas tiene, pero en cambio, su tiempo de entrenamiento se incrementa de forma exponencial. Esto obliga a buscar una solución de acuerdo a nuestras necesidades, que generalmente mantendrá un buen compromiso entre tamaño y tiempo de entrenamiento.
La red más ampliamente usada es sin duda el Perceptrón multicapa con algoritmo Backpropagation. Es la red neuronal por excelencia, siendo muy versátil y con unas capacidades aceptables para la mayoría de aplicaciones. Tal y como se ha visto, adolece del problema de sobreentrenamiento, momento en el cual la red no solo deja de aprender si no que comienza a olvidar lo aprendido.
Además de esto, el Backpropagation es un algoritmo de convergencia lenta, lo que lo hace poco adecuado para aplicaciones en tiempo real, al contrario de lo que sucede con las funciones de base radial.

En este trabajo se usan 3 de las redes indicadas más arriba: dos redes no recurrentes, como son el Perceptrón multicapa y las redes de base radial, y otra de tipo recurrente como es la Elman, proporcionando de este modo una visión global de las posibilidades, ventajas y carencias de los tipos de redes más importantes.

3. Implementación y Estudio de un Clasificador de Patrones Mediante MATLAB

Como ya se ha comentado anteriormente, MATLAB ha sido el entorno de desarrollo elegido debido a su versatilidad y a su amplia oferta de complementos (toolboxes) listos para usarse. Además de esto, MATLAB es un entorno que está ampliamente implantado, con aplicaciones en todo tipo de sectores.

3.1. *Exposición del Caso: Inspección Visual de Gajos de Mandarinas*

En nuestro caso, se obtienen imágenes de una cinta transportadora de una cadena productora, de modo que en cada fotografía haya un solo gajo, posicionado tal y como lo haya dispuesto la maquinaria anterior, que no tiene porque ser siempre igual.
Se realizará un tratamiento digital de las mismas, para posteriormente obtener las características que se estimen oportunas. Una vez hecho esto, se dispondrá de un conjunto de datos que permitirán entrenar una red neuronal que se irá ajustando para obtener el mínimo error.

3.2. *Elección de las Características de los Gajos*

A partir de un estudio previo sobre cuáles son las variables que mejor pueden definir la forma de un gajo de mandarina, se han elegido 4 características que han sido las que mejores propiedades discriminantes han presentado de todas las analizadas.

La primera de ellas es la longitud de la proyección horizontal del gajo, de modo que si un gajo se encuentra seccionado en su eje longitudinal, esta característica permitiría discriminarlo. En la figura 4.a es posible observar un gajo bueno con la proyección horizontal representada. Sin embargo en la figura 4.b se observa que la característica mencionada aparece muy distorsionada.

La segunda característica escogida es, como cabría esperar, la proyección vertical del gajo, que igualmente permite discriminar aquellos gajos que hayan sido seccionados o que simplemente sean defectuosos. La figura 4.c muestra un gajo bueno con la proyección vertical representada. En la figura 4.d se puede comprobar la diferencia.

La tercera característica que se ha tenido en cuenta es el área del gajo. De este modo, los gajos que no tengan ciertas partes de su estructura tendrán un área menor, lo cual permitirá diferenciarlos. En el caso de que el gajo sea mucho más grande de lo normal también lo discriminará, obteniendo de ese modo unos gajos muy similares en cuanto al área.
La cuarta característica se corresponde con la extracción de los datos para la evaluación de la expresión $\frac{(Perímetro)^2}{Área}$ que surge de contemplar la posibilidad de que un gajo se haya roto como pasa en el caso de la figura 4.e. En el citado caso no es posible discriminarlo por el área del gajo, ya que las zonas de color blanco no se computan, y, a

menos que la rotura haga que el gajo esté muy deteriorado y active cualquiera de las discriminaciones por proyecciones, el gajo sería catalogado como bueno, cuando su estado no lo permite.

Al escoger el cociente $\frac{(Perímetro)^2}{Área}$ se consigue que si el gajo tiene un área normal pero su perímetro es excesivo el cociente se dispare y discrimine a dicho gajo. La razón de poner el perímetro al cuadrado responde a razones de unidades, al ser el área bidimensional y el perímetro unidimensional, además de reforzar el carácter discriminante de esta característica.

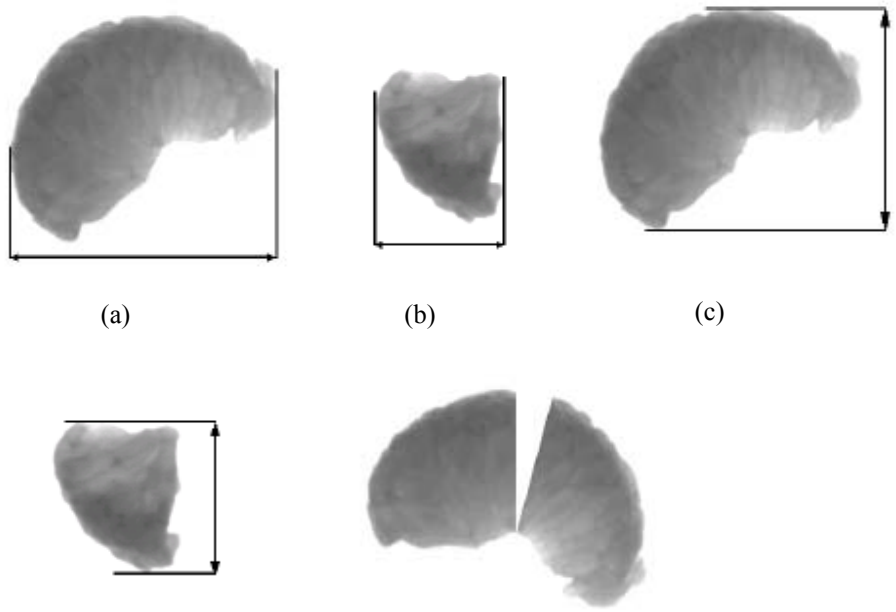


Figura 4: (a) P ión horizontal de la longit n gajo bueno, (b): Proyección horizontal de la longitud de un gajo malo, (c) Proyección vertical de la longitud de un gajo bueno, (d): Proyección vertical de la longitud de un gajo malo,(e): Imagen de un gajo discriminado por la cuarta característica.

Otras muchas características podrían ser empleadas con discutible mayor o menor efectividad, pero la experiencia acumulada indica que estos 4 indicadores se muestran suficientemente potentes a la hora de proceder al reconocimiento.

3.3. *Programa Desarrollado: "Gajos"*

El programa se ha desarrollado con un propósito muy sencillo y definido: a partir de unas fotos de gajos de mandarina obtenidas en una cadena de producción, obtener un criterio que permita clasificar los gajos analizados como buenos o malos.
Para ello se dispone de una base de datos de fotografías de gajos de mandarina de las calidades posibles, que el programa deberá analizar y a continuación extraer sus características.
Lo primero que se hace es importar las imágenes, que están en escala de grises. Se cargan como una matriz de n capas a modo de array. Como el principal factor de selección será la forma del gajo, el siguiente paso consiste en realizar una binarización, de la imagen anterior, lo cual consiste en establecer un umbral de decisión sobre el nivel de gris, de modo que separe el blanco (fondo) del negro (objeto). Empíricamente, y tras multitud de ensayos realizados con gajos de distinta calidad con las mismas condiciones de iluminación, se ha seleccionado un umbral de 0.7[1] que permite eliminar pequeños salientes en los bordes que podrían por un lado interferir en el cálculo del perímetro, y por otro en el cálculo de las proyecciones horizontal y vertical del gajo.
Una vez que la imagen (de 640x480 píxeles), ha sido binarizada, se dispone de una matriz de ceros y de unos, en nuestro caso de 640 columnas y 480 filas, sobre la que ya es posible calcular las variables características antes comentadas. Para ello se procede a situar el gajo de modo que las características discriminantes de las proyecciones verticales y horizontales sean más efectivas. El fin último es situar todos los gajos en la misma posición, independientemente de cómo estén realmente posicionados en la cinta transportadora, sobre la que vienen en diferentes posiciones, tal y como puede verse en la figura 5.

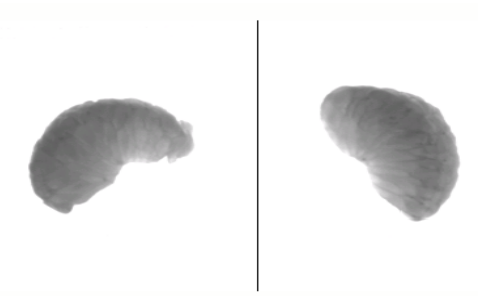


Figura 5: Posición de distintos gajos en la cinta transportadora.

[1] Matlab suele trabajar con datos de tipo double y codifica las imágenes en escala de grises de 0 (negro puro) al 1 (blanco puro)

La posición estándar para el cómputo de las variables características que se ha elegido es la de maximización de la proyección del eje x. Para ello se rota la imagen 360 grados con un paso de 20, de modo que al hacer el recorte del gajo sea máxima dicha proyección, escogiéndose entonces esa posición como la adecuada del gajo. Una vez que el gajo ha sido colocado en esa posición, se calculan ambas proyecciones, contando los píxeles de la base y el lado, el área (representada en color verde), con ajustes por diagonales (es decir, 4 píxeles en línea recta y 4 píxeles en diagonal no computan como la misma distancia a la hora de calcular el área) y el $\frac{(Perímetro)^2}{Área}$, cogiendo el área calculada anteriormente y calculando el perímetro como el número de píxeles que forman la frontera entre las dos zonas. A modo meramente informativo también se halla el centroide del gajo, y se representa con un punto verde, como puede verse en la figura 6.

Figura 6: Imagen de un gajo analizado.

Una vez que el programa cuente las características de todos los gajos, se procede al entrenamiento de una red neuronal, que más adelante permitirá clasificar otros gajos, discriminando en línea dentro del proceso de producción entre gajos aptos y no aptos.

En el programa desarrollado, se han contemplado los 3 tipos de redes neuronales comentados anteriormente: Feed-Forward, Elman y Radial Basis.Estas 3 redes han sido elegidas por varias razones: en cuanto a la Feed-Forward, es una de las primeras redes neuronales desarrolladas, considerando muy interesante el estudio de sus resultados. La red Elman es una red más nueva, que sigue utilizando el algoritmo Backpropagation, pero revisa diversos conceptos de las anteriores redes neuronales. Como revisión del modelo anterior será interesante estudiar el cambio en los resultados. La red Radial Basis es una red neuronal de las consideradas más potentes, debido a su sencillez, su potencia y su rapidez. Estudiando sus resultados es posible comprobar la mejora que han seguido las redes neuronales, al mismo tiempo que se observa una red que no está

basada en Backpropagation. Otra razón de la elección de estas 3 redes es por su idéntico formato de los datos de entrada, de supervisión y de salida, que hacen mucho más sencillo el manejo de los datos. En el caso de las redes Elman y Feed-Forward, es necesario especificar además el número de neuronas de las capas ocultas, así como su función de activación. Se han contemplado todas las funciones de activación que MATLAB implementa:

- Radial Basis, considerada simplemente como función de activación representa una campana gaussiana.
- Tangente de sigmoide.
- Logaritmo de sigmoide.
- Lineal.

Se ha considerado tras diversas pruebas que los dos primeros tipos de redes obtienen su mejor rendimiento con un número de 2 a 4 capas ocultas, de modo que se decidió incluir soporte desde una capa oculta hasta 5. También se ha incluido la posibilidad de variar tanto el algoritmo de entrenamiento (Levenberg-Mardquardt, Descenso del Gradiente, Descenso del Gradiente con Propagación hacia Atrás del Momento, Regulador Bayesiano, y BFGS quasi-Newton) como el tipo de error que se quiere minimizar (media cuadrática, media cuadrática con regularización, media cuadrática derivativa, y error absoluto de la media).

La implementación realizada es capaz de variar desde su GUI (*Graphical User Interface*) tanto el número de iteraciones deseadas como un error mínimo a alcanzar, dejando automáticamente un control sobre el gradiente, de modo que si se reduce a cero el entrenamiento cese por su incapacidad para seguir siendo eficiente. Otro de los controles de entrenamiento implementados incluye una supervisión del error alcanzado en el entrenamiento. Gracias a esto, si el error alcanzado no es inferior a un umbral que se ha establecido en 0.1, la red vuelve a ser entrenada de nuevo, y así hasta un máximo de 5 reentrenamientos, transcurridos los cuales si la red sigue sin descender su error, se proseguirá con el resto de las operaciones normales, no sin antes advertir al usuario del suceso acontecido mediante un mensaje en pantalla.

Una vez entrenada la red se ha considerado importante, además del error alcanzado, la simulación de los datos entrenados para ver el comportamiento de la red, simulación que también se realiza justo antes del entrenamiento para comprobar la mejoría obtenida gracias al mismo. Esta simulación se realiza introduciendo los gajos en la red para comprobar la salida obtenida. En teoría, las salidas de los patrones entrenados, al ser simulados de nuevo deberían dar como resultado la salida propuesta por el supervisor, pero debido a que el error obtenido en el entrenamiento no es exactamente 0, como

sucederá en las de Base Radial, con un error del orden de 1×10^{-27}, las salidas no serán cero o uno, sino un valor intermedio de la misma en función de lo parecido que sea el patrón simulado a lo considerado como patrón bueno. Esto ha llevado a definir el parámetro que se ha denominado "*fiabilidad de la clasificación*", y a la hora de clasificar los gajos mediante el mismo procedimiento, se definen el denominado "*índice de seguridad*" y el de "*calidad del gajo clasificado*". A la hora de obtener este índice se ha tenido en cuenta lo siguiente: si los límites están situados en el uno y el cero, el umbral estará en el 0.5. Cuando el resultado de la simulación sea un 0.1 se clasificará como un 0, pero se asignará una fiabilidad del 80%. Siguiendo este razonamiento, un 0.6 será tomado como un 1, pero con un índice de fiabilidad de sólo 20 %. Como era de esperar, tanto el uno como el cero tienen una fiabilidad del 100 %. Estos índices se promedian para obtener la fiabilidad de la red como clasificador justo antes del entrenamiento y al terminar.

Tras entrenar la red ya es posible utilizar el clasificador. Para ello es necesario proporcionar como entradas las 4 características de un gajo en la capa de entrada, para obtener a la salida el resultado de la clasificación, cuyo análisis de resultados se basa en los índices anteriores con una salvedad. Para las redes Elman y Radial Basis se ha comprobado tras multitud de pruebas que no sólo se obtiene una clasificación bipolar de "0", ó "1", sino que cuanto mejor es el gajo más próximo a uno está el resultado, y viceversa.

Este mismo índice que permitió indicar la fiabilidad de la clasificación ahora permite informar de la calidad del gajo, de modo que si el resultado de la clasificación es "1", la calidad será del 100 %, y si el resultado es 0,6 la calidad será del 60%. Estos indicadores de calidad podrían resultar de mucha utilidad a la hora de discernir entre distintas calidades de gajos, adecuando la selección al segmento de mercado para el que se destine el producto final. Pongamos como ejemplo una planta de producción en la que se utiliza los gajos muy malos para piensos, los regulares para zumo y los buenos para vender en hostelería.

Otra idea que surgió a raíz de observar esta característica a la hora de clasificar de las redes Backpropagation implementadas fue la clasificación por colores. Tomando como colores patrón el verde puro para el gajo bueno y el rojo puro para el gajo malo, se utilizó la colorimetría RGB para obtener una amplia gama de colores intermedia, calculada automáticamente. El color que corresponde al verde puro es el [0 1 0], y el del rojo es [1 0 0]. El ajuste que se le realiza a esos colores es el siguiente: [1-salida_red salida_red 0], de modo que una salida de la red correspondiente a un gajo bueno, que es

"1", daría como resultado un color [0 1 0], y una salida intermedia por ejemplo de 0.6 daría como resultado un color [0.4 0.6 0]. En la figura 7 puede verse el resultado buscado y obtenido.

Figura 7: Ajuste de color en clasificación por colores.

3.3.1. Red Feed-forward

La figura 8 representa el entrenamiento de una red neuronal con 16 neuronas distribuídas en tres capas en forma de 8-4-4 con función de activación tangente de sigmoide. El error de entrenamiento es 0,1216, el porcentaje de gajos bien clasificados es 78,13, y los mal clasificados representan el 21,88 %. En cuanto a la fiabilidad, arroja un resultado de 56,74 %. El tiempo empleado para el entrenamiento ha sido de 31,6 segundos.

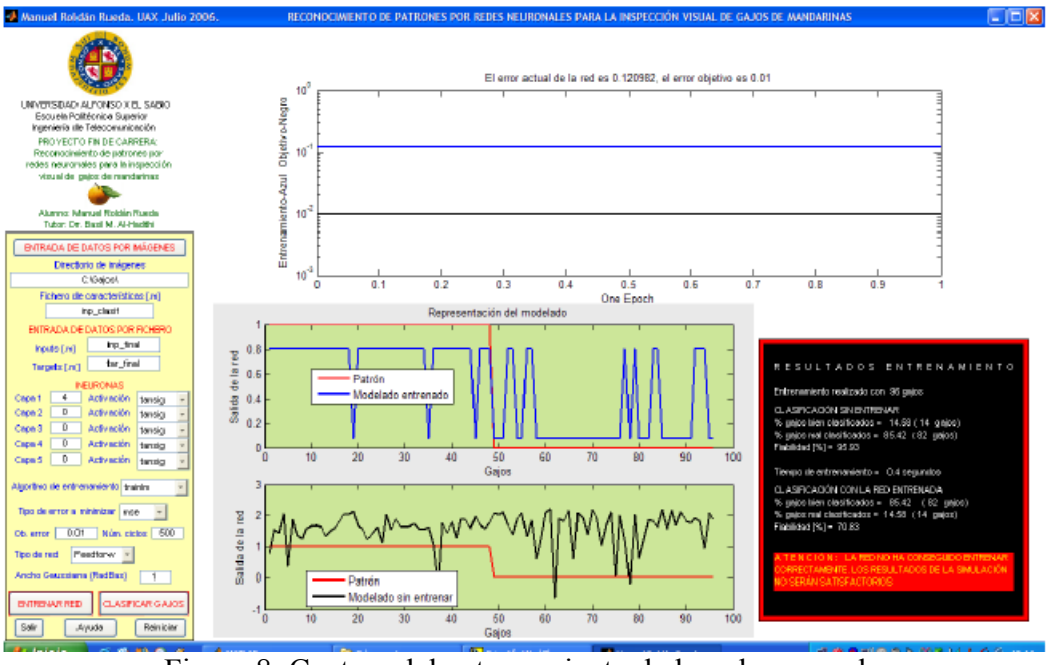


Figura 8: Captura del entrenamiento de la red neuronal.

Variación de la función de activación de las neuronas de cada capa

A continuación se ha procedido a variar las funciones de activación de las neuronas por capas, siguiendo los mismos ejemplos del apartado anterior. La figura 9 representa el entrenamiento de una red neuronal con 4 neuronas distribuídas en una capa, con función de activación radial basis. El error de entrenamiento es 0,08868, el porcentaje de gajos bien clasificados es 87,5, y los mal clasificados representan el 12,5 %. En cuanto a la fiabilidad, arroja un resultado de 73,18 %. El tiempo empleado para el entrenamiento ha sido de 9,6 segundos.

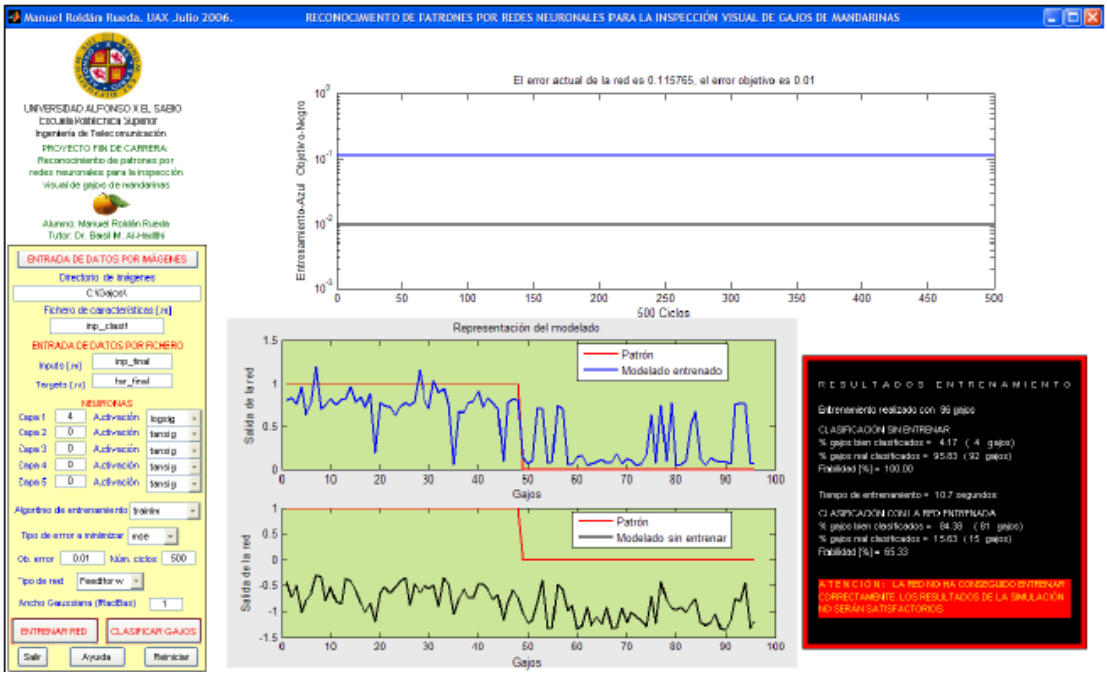


Figura 9: Captura del entrenamiento de la red neuronal.

Variación del tipo de error a minimizar

La figura 10 representa el entrenamiento de una red neuronal que minimiza el error de la media cuadrática con regulación, con 16 neuronas distribuídas en tres capas, la primera de ellas con 4 neuronas con función de activación tangente de sigmoide, la segunda con 8 neuronas con activación radial basis y la tercera con 4 neuronas con activación tangente de sigmoide. El error de entrenamiento es 0,49151, el porcentaje de gajos bien clasificados es 85,42, y los mal clasificados representan el 14,58%. En cuanto a la fiabilidad, arroja un resultado de 63,82%. El tiempo empleado para el entrenamiento ha sido de 25,1 segundos.

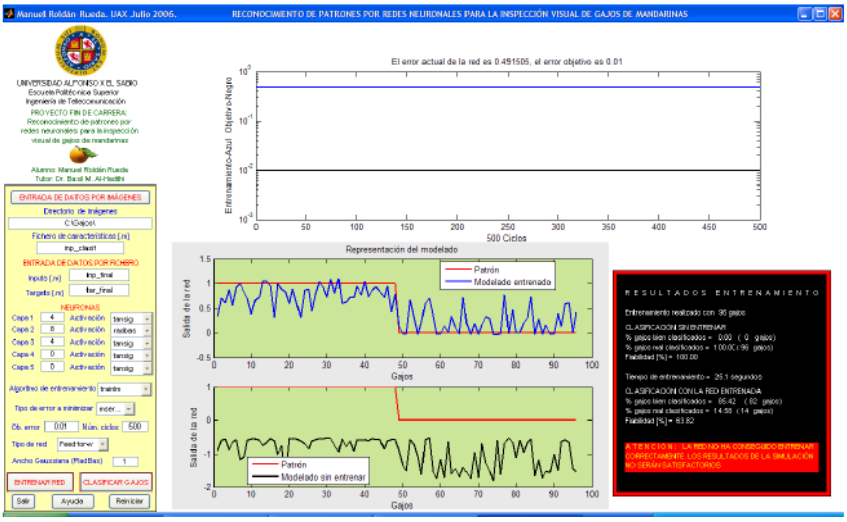


Figura 10: Captura del entrenamiento de la red neuronal.

3.3.2. Red Elman

La figura 11 representa el entrenamiento de una red neuronal con 16 neuronas distribuídas en tres capas en forma de 8-4-4. El error de entrenamiento es 0,11186, el porcentaje de gajos bien clasificados es 85,42, y los mal clasificados representan el 14,58 %. En cuanto a la fiabilidad, arroja un resultado de 80,39 %. El tiempo empleado para el entrenamiento ha sido de 64,8 segundos.

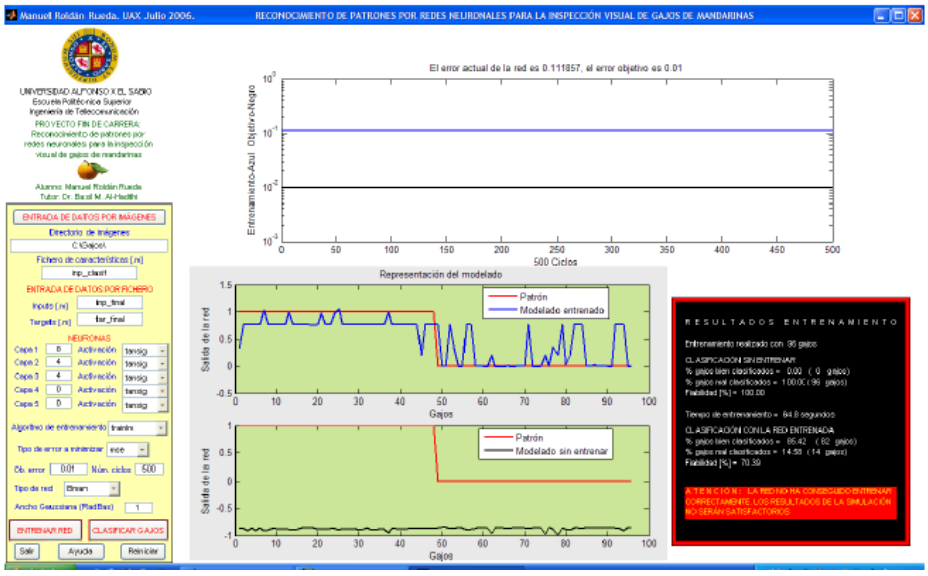


Figura 11: Captura del entrenamiento de la red neuronal.

3.3.3. Red Radial Basis

Variación del ancho de la función (spread)

La figura 12 representa el entrenamiento de una red neuronal radial basis con un ancho de campana (spread) de 1. El error de entrenamiento es $3{,}6401 \times 10^{-27}$. El sistema clasifica correctamente todos los gajos con una fiabilidad del 100% y tarda 0,1 segundos en entrenar.

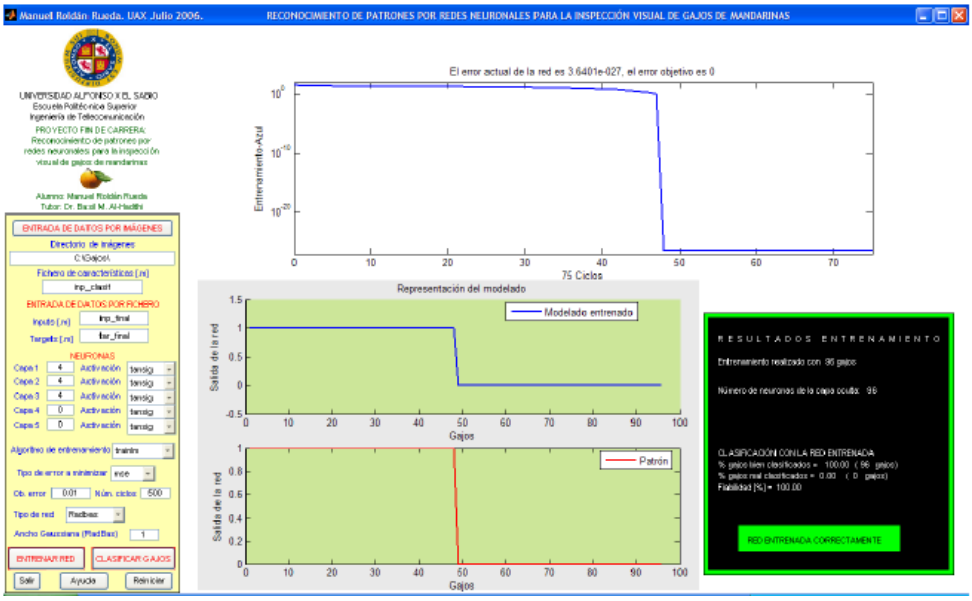


Figura 12: Captura del entrenamiento de la red neuronal.

3.4. *Pruebas de clasificación de gajos*

Partiendo de que se conoce que de las 20 de partida, 6 de ellas son malas y 3 dudosas, por lo tanto el resultado teórico debería ser tener de 11 a 14 buenas (verdes) y de 9 a 6 malas (rojas).

3.4.1. Red Feed-forward

Las siguientes gráficas corresponden a las variaciones realizadas en las capas y sus neuronas.

En la figura 13 es posible observar la clasificación que realiza una red de dos capas con 4 neuronas en la primera capa con funciones de activación tangente de sigmoide y 2 en la segunda capa con la misma activación. Clasifica como malos 6 gajos y como buenos 14. La media de la seguridad de la clasificación es un 66% y la desviación típica es de un 35 %.

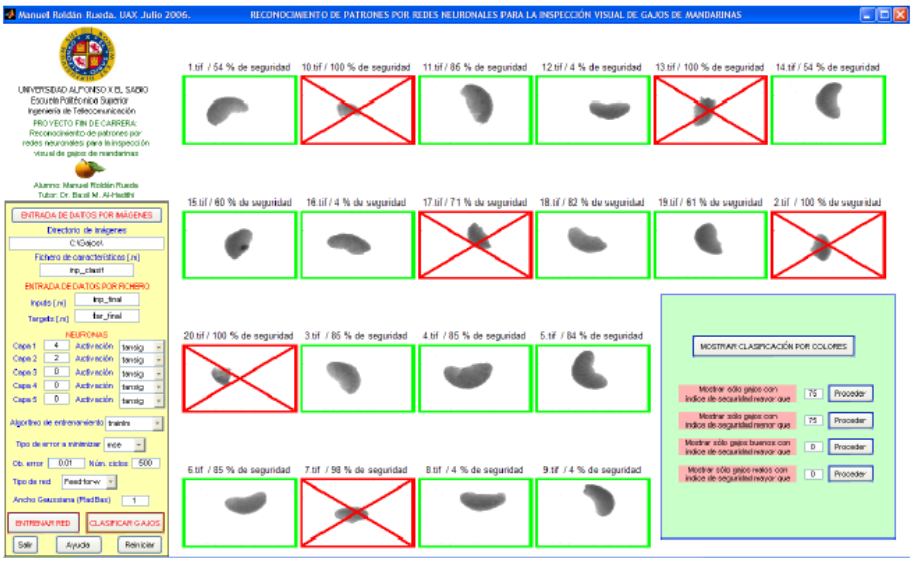


Figura 13: Captura de la clasificación de los gajos.

3.4.2. Red Elman

En la figura 14 es posible observar la clasificación que realiza una red de una capa con 4 neuronas con funciones de activación tangente de sigmoide. Clasifica como malos 6 gajos y como buenos 14. La media de la seguridad de la clasificación es un 66% y la desviación típica es de un 23 %.

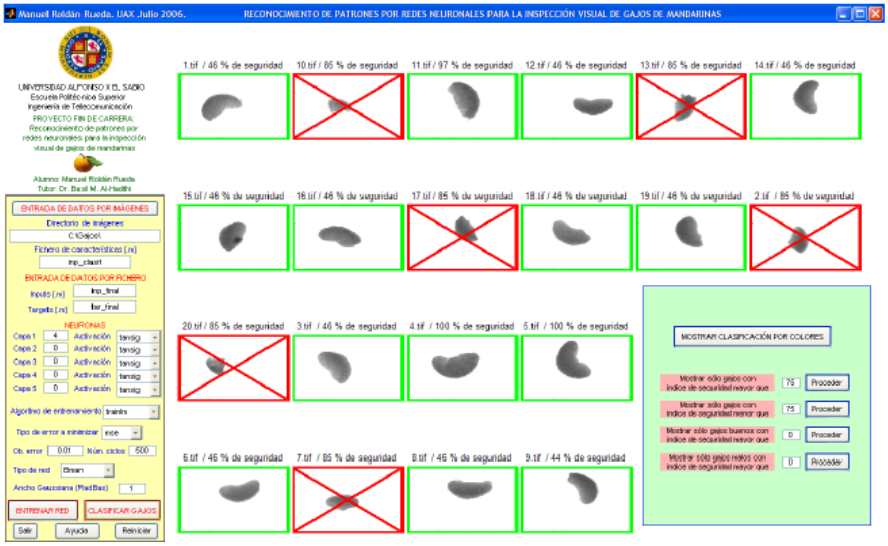


Figura 14: Captura de la clasificación de los gajos.

3.4.3. Red Radial Basis

Todas las clasificaciones analizadas basadas en redes de tipo radial basis clasifican 12 gajos como buenos y 8 como malos, con salidas bipolares, lo que implica una seguridad de clasificación del 100%. La figura 15 representa el entrenamiento de una red neuronal radial basis con un ancho de campana (spread) de 1.

Figura 15: Captura de la clasificación de los gajos.

4. Conclusiones

Este trabajo se ha presentado una interfaz simple y amigable orientada al proceso de clasificación de gajos de mandarina en conserva dentro de un proceso industrial utilizando MATLAB como herramienta de desarrollo, implementado todas las características que MATLAB ofrece a la hora de crear una red neuronal: redes de tipo Elman, Feed-Forward y Radial Basis. Las dos primeras además permiten variar su topología, tanto en número de neuronas, como en número de capas y las funciones de activación de las mismas, los tipos de error a minimizar, el algoritmo de entrenamiento y

el objetivo de error. Las redes de tipo Radial Basis permiten elegir el ancho de la campana gaussiana.
Como puede suponerse, también se ha incluido una rutina que entrene la red neuronal en modo "offline", lo que le da la potencia de poder variar los algoritmos utilizados en el proceso de inspección si así fuera necesario.

Como último bloque del sistema presentado figura el clasificador propiamente dicho, el cual analiza y cataloga los gajos presentes en la línea de producción ofreciendo distintas formas de discriminación, como pueden ser mediante un mínimo de calidad, o mediante una seguridad en la clasificación determinada. También se ofrece una interfaz que permite visualizar los gajos por colores, de modo que cuanto mejores sean serán mas verdes, y cuanto peor serán más rojos, lo que en principio nos permite definirlo como un acercamiento a la teoría difusa desde el campo de las redes neuronales.

5. Bibliografía

CHOW, C. K.(1957): An optimum character recognition system using decision functions, IRE Trans. Electronic Computers, vol. EC-6, págs. 247-254, Diciembre.
DAVIDSON, V., Chu, T., and Ryks, J. (1999), Fuzzy methods for automated inspection of food products, IEEE ISBN 0-7803-5211-4, págs. 909-912.
ELMAN, J. L.(1990): Finding structure in time, Cognitive Science, Vol. 14, Capítulo 10, págs. 179-211.
ELMAN, J. L.(1993): Learning and development in neural networks: The importance of starting small, Cognition 48, págs. 71-99.
FAHLMAN S. E.(1988): An Empirical Study Of Learning Speed In Back-Propagation Networks, CMV Technical Report, págs. 88-162.
FISHER, R. A.(1930): The Genetical Theory of Natural Selection, Oxford, Claredon Press.
GEORGIEV, A. A.(1987): Fitting of multivariate functions, Proceedings of the IEEE, 75, págs. 970-971,.
GIROSI, F., Poggio, F.(1990): Networks and the Best Approximation Property, Biol. Cybernetics, 63, págs. 169-176.
HECHT-NIELSEN, R.(1989): Neurocomputing, Addison-Wesley, Reading.
KAMENTSKY, L. A., Liu, C. N.(1964): A Theoretical and experimental study of a model for pattern recognition, Computer and Information Sciences, Washington, D.C., Spartan, págs 194-218.
LEDLEY, R., Wilson, J.(1964): Concept analysis by sintax processing, Proc. Am. Documentation Inst., vol.1, October, 1964.

MAIR, G. (1988): Industrial Robotics, Ed. Prentice Hall, New York.
MATHWORKS, (2006): The MathWorks Inc. http://www.mathworks.com
MATLAB HELP, (2006): ayuda online de la toolbox de redes de Matlab. http://www.mathworks.com
PASK, G.(1962): The simulation of learning and decision-making behaviour, Aspects of the Theory of Artificial Intelligence, New York, Plenum, págs. 165-211.
PHAM, D. T., Liu, X.(1995): Neural Networks for Identification, Prediction and Control, Springer, págs. 220-245.
POLTZEITNER, W., Schwingshakl, G. (1992): Real-time surface grading of profiled wooden boards, Industrial Metrology, págs. 283-298.
POWELL, M.(1987): Radial basis functions for multivariable interpolation: a review, J.C. Mason, M. G. Cox (Eds.), Algorithms for Approximation, págs. 143-167.
Rosen, J. B.(1965): Pattern separation by convez programming, J. Math. Anal. and Application, vol.10, págs. 123-134.
Rosenblatt, F.(1957): The perceptron: A perceiving and recognizing automaton, Report 85-460-1, Project PARA, Cornell Aeronautical Laboratory.
RUMELHART, D.E., Hinton, G.E., Williams(1986): R.J., Learning internal representations by error propagation, En: D.E. Rumelhart y J.L. McClelland (Eds.), "Parallel distributed processing" (págs. 318-362), Cambridge, MA: MIT Press.
WENDLAND, H.(2002): Fast evaluation of radial basis functions: Methods based on partition of unity, Approximation Theory X: Wavelets, Splines, and Applications, C. K. Chui, L. L. Schumaker, and J. Stöckler, Eds., Vanderbilt University Press, págs. 473-483.
WIDROW, B., Hoff, Jr. M. E.(1960): Adaptive switching circuits, IRE WESCON Convention Record, part 4, págs. 96-104.
WIDROW, B., Lehr, M.(1990): 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation, Proceedings of the IEEE, 78(9), págs. 1415-1442, Sept tiembre.
WIDROW, B., Smith, F.W.(1964): Computer and Information Sciences, Washington, D. C., Spartan, págs. 288-317, 1964.
WINDER, R. O.(1963): Threshold logic in artificial intelligence, Proc. IEEE Winter General Meeting, Sessiones on Artificial Intelligence, págs. 107-128, Enero.
YU, X., Chen, G., Cheng, S.(1995): Dynamic Learning Rate Optimization of the Backpropagation Algorithm, IEEE Trans on Neural Networks, Vol 6, No 3, págs. 669-677, Mayo.
ZADEH, L. (1965). Fuzzy sets. Information and Control, 8(3):pg 338-353.