Clasificación de texturas mediante redes neuronales

GONZALO PAJARES (*); VICTORIANO MORENO (**); JESÚS M. DE LA CRUZ (***)

RESUMEN Muchas aplicaciones basadas en imágenes necesitan conocer las diferentes clases de texturas que aparecen en la imagen para diferentes propósitos. En varias técnicas de ciencias e ingeniería, puede resultar útil extraer una importante información subyacente en la imagen. En este trabajo, utilizamos el modelo del Perceptrón Multicapa, para realizar la clasificación de textura de una imagen en color. Se trata de un modelo de red ampliamente utilizado en clasificación. El método propuesto es una técnica clásica de clasificación supervisada. Los patrones de entrada a la red son vectores tridimensionales y contienen información de textura para las tres bandas espectrales (Rojo, Verde y Azul) para cada imagen. La información que utilizamos es una medida del histograma del nivel de gris de una región. El número de clases se establece previamente y para cada clase se selecciona un número de patrones a partir de las imágenes disponibles. Se realiza un análisis comparativo entre el método propuesto y el denominado Learning Vector Quantization como método supervisado. Los resultados experimentales hacen que el método sea útil para clasificación de texturas.

TEXTURE CLASSIFICATION BY MEANS OF NEURAL NETWORKS

ABSTRACT *Most image-based applications require knowing the different classes of textures appearing in the image for different purposes. In several sciences and engineering techniques the texture classification can be useful to extract a very important underlying information in the image. In this work we use the widely used Multi-Layer Perceptron neural network to perform the texture classification for colour images. The proposed method is a classical supervised learning approach. The input pattern vectors are three-dimensional ones and contain texture information for the three spectral bands (red, green and blue) for each image. The information used is a measure of the gray-level histogram of a region. The number of classes is established previously and for each class a number of patterns is selected from the available images. A comparative analysis is carried out against the supervised Learning Vector Quantization. The experimental results make the method useful for texture classification.*

Palabras clave: Perceptrón Multicapa, Clasificación de patrones, red neuronal, discriminación de textura, aprendizaje supervisado, vector quantization, learning.

1. INTRODUCCIÓN

Muchas aplicaciones basadas en imágenes requieren discriminar entre diferentes tipos de textura, subyacentes en las imágenes, con diferentes propósitos. Así en una obra de ingeniería civil puede resultar de interés determinar si el terreno presenta estructuras de naturaleza rocosa, vegetal, arenisca, etc. En aplicaciones de Medio Ambiente o Protección Civil, puede ser útil determinar el grado de deforestación debido a diferentes causas, a saber: incendios forestales, desertización o agresiones al entorno. En Agricultura puede resultar de interés identificar los diferentes tipos de cultivos para estimar la producción de una determinada zona.

El método que se presenta está especialmente diseñado para la clasificación de texturas en imágenes de entornos naturales. La naturaleza de las estructuras subyacentes en este tipo de imágenes (bosques, lagos, cultivos agrícolas, terrenos montañosos, etc.) hace que la clasificación de dichas estructuras sea un problema de elevada complejidad. Los métodos de clasificación se han utilizado en numerosas aplicaciones con éxito en aquellas imágenes donde los objetos a clasificar se pueden extraer con facilidad y a partir de ellos una serie de propiedades discriminantes fácilmente computables. Recordemos en este sentido las técnicas de reconocimiento de caracteres mediante la utilización de momentos invariantes (Pajares y Cruz 2001, Gonzalez y Woods 1993), o por poner otro ejemplo menos intuitivo, la clasificación de impurezas en el algodón (Lieberman y Patil 1997). En estos tipos de imágenes los elementos a clasificar se manifiestan perfectamente distinguibles sobre un fondo de distinto color, con lo que se pueden extraer con cierta facilidad. Sin embargo, en imágenes de escenas naturales los elementos a clasificar no se manifiestan tan claramente y por supuesto sus formas no suelen aportar información relevante. En este tipo de imágenes resulta más útil recurrir a técnicas de tratamiento estadístico y por tanto, considerar la naturaleza estadística de las texturas subyacentes en las imágenes.

Una forma de discernir entre diferentes texturas es comparar sus estadísticas del *nivel de gris de primer orden* (Nalwa, 1993), si esto lo formalizamos y recurrimos a una

(*)(**) Dpto. Teledetección de Indra Espacio
Mar Egeo 4, Polígono Industrial 1
28830 San Fernando de Henares.-MADRID

(*)(***) Dpto. Arquitectura de Computadores y Automática.
Facultad de CC. Físicas.- Universidad Complutense
28040 MADRID
Phone: 34.1.3 96 33 55. Fax: 34.1. 3 96 39 12.E-mail: gpajares@indra.es

técnica de clasificación, seremos capaces de determinar de forma automática entre diferentes texturas, que se encuentran establecidas como clases de texturas.

El método que proponemos se basa en el modelo de red neuronal conocido como el Perceptrón Multicapa (PM), ampliamente utilizado en clasificación. Se fundamenta en la retro-propagación del error (González y Woods 1993). Los patrones de entrada a la red son vectores, que contienen información estadística relativa a los niveles de gris de primer orden convenientemente tratados.

Las redes neuronales artificiales se han utilizado para clasificación de estructuras naturales en imágenes aéreas y de satélite, entre otros podemos citar a Ji (2000), Heermann y Khazenie (1992), Bischof y col. (1992), Civco (1993), siendo alternativas válidas para las técnicas de clasificación estadística como proponen Paola y Schowengerdt (1997). En este último trabajo podemos encontrar un exhaustivo estudio sobre el PM. Los clasificadores basados en redes neuronales no paramétricas tienen la gran ventaja de no tener que presuponer modelos de comportamientos estadísticos a priori. Por tanto, las redes neuronales funcionan bien en los casos en que los datos son altamente no Gaussianos y esto ocurre en clasificaciones que incorporan medidas de textura, tal y como sugieren Lee y col. (1990) o Augusteijin y col. (1995) y en clasificaciones de datos procedentes de varias fuentes Benediktsson y col. (1990), Gong (1996) o Bruzzone y col. (1997).

Este trabajo está organizado como sigue, en la sección 2 se introduce la técnica para extraer las propiedades de las texturas subyacentes y su encapsulación como vectores tridimensionales de entrenamiento. En la sección 3 se introduce el concepto y funcionamiento del PM y el mecanismo de retro-propagación. Hemos preferido ofrecer al lector un desarrollo profundo sobre el funcionamiento de este tipo de redes, con el fin de facilitar su implementación sin necesidad de recurrir a otras fuentes, a la vez que le sirve como un tutorial. En la sección 4 se describe la topología de la red, los parámetros asociados con la misma, se realiza la valoración de los resultados obtenidos y un análisis comparativo frente a otra técnica de clasificación supervisada muy conocida, se trata del método Learning Vector Quantization (LVQ) en terminología inglesa y que preferimos denominarlo con dicha terminología. Finalmente, en la sección 5 se analizan los resultados.

2. EXTRACCIÓN DE PROPIEDADES

Antes de seleccionar las propiedades a extraer para la clasificación de texturas, hemos considerado algunas propiedades de textura encontradas en la literatura, Lee y col. (1990) o Augusteijin y col. (1995), entre ellas ciertas propiedades derivadas de las matrices de co-ocurrencia (homogeneidad, correlación, entropía) y de la transformada de Fourier (magnitud máxima, magnitud media, energía de la magnitud, varianza de la magnitud) e incluso los filtros de Gabor (con diferentes frecuencias y orientaciones). Aunque, los resultados obtenidos parecen prometedores, entrañan la dificultad de tener que definir la dirección y distancias necesarias para calcular las matrices de co-ocurrencia, o la frecuencia y orientación en los filtros de Gabor, así como la región donde se debe aplicar la transformada de Fourier. En cualquiera de los tres casos la elección resulta difícil, por lo que hemos recurrido a las propiedades de textura basadas en las estadísticas del *nivel de gris de primer orden* (Nalwa, 1993), ya que únicamente requieren definir las dimensiones de la ventana donde se van a calcular, la definición de las dimensiones de ventanas es una práctica habitual en muchas técnicas del tratamiento de imágenes.

Por *primer orden* se entienden las estadísticas en las que se ven involucrados pixels simples en contraposición a las estadísticas de más de un pixel (pares, tripletes, etc.). En las estadísticas de primer orden se puede utilizar el histograma del nivel de gris de la textura, cuya normalización proporciona la función de densidad de probabilidad de la imagen caracterizada por la textura. Se puede pues, comparar los histogramas normalizados del nivel de gris de imágenes de texturas, o utilizar varias medidas derivadas, tales como la media, la mediana o la varianza. Sea z una variable aleatoria que representa la intensidad discreta de la imagen, y sea $p(z_i)$ con $i=1,2,..L$ su histograma correspondiente, con L el número de niveles de intensidad diferentes. El momento n-ésimo de z respecto de la media se define como,

$$\mu_n(z) = \sum_{i=1}^{L} (z_i - m)^n p(z_i) \qquad [1]$$

donde m es el valor medio de z (el nivel medio de la región cuya textura está siendo considerada).

$$m = \sum_{i=1}^{L} z_i p(z_i) \qquad [2]$$

el momento de segundo orden, denominado varianza $\sigma^2(z)$ es de particular interés para la descripción de texturas. La varianza es una medida del contraste de intensidad que se puede usar para obtener descriptores de suavidad relativa (Fu y col. 1988), por ejemplo, el valor

$$R = 1 - \frac{1}{1+\sigma^2(z)} \qquad [3]$$

es 0 para áreas con intensidad constante $\sigma^2(z)=0$ y es 1 para valores grandes de $\sigma^2(z)$, por consiguiente obtenemos valores restringidos al intervalo [0,1). El momento de tercer grado es una medida de la oblicuidad del histograma, mientras que el de cuarto orden es una medida de cuán plano es el histograma. A partir del momento de quinto orden no es tan fácil relacionar los momentos con la forma del histograma, pero sirven para obtener más información cuantitativa de discriminación de texturas (Fu y col. 1988).

La aplicación, que presentamos, está diseñada para el tratamiento de imágenes multiespectrales, de suerte que para cada banda espectral se obtiene un valor R_i dado por (3), donde i hace referencia a la banda espectral correspondiente. En el caso de imágenes en color i = R, G, B, es decir, banda del Rojo, Verde y Azul. En todos los casos, R_i se obtiene para la misma localización espacial (x, y) en las diferentes bandas espectrales y se calcula sobre un entorno de vecindad alrededor de dicha localización espacial. La dimensión de los vectores patrón de entrada a la red depende del número de bandas espectrales utilizadas. En el caso de imágenes en color RGB, estos vectores son tridimensionales, de la forma $\mathbf{x} = (R_R, R_G, R_B)$. Con el método que proponemos, integramos dos tipos de información, a saber: cromática y de textura. Esto último constituye uno de los principales hallazgos, que se refleja en los resultados obtenidos al sustituir los atributos de textura por las simples componentes espectrales RGB, es decir, cuando los patrones de entrada a la red son vectores de la forma $\mathbf{x} = (R, G, B)$ en lugar de los vectores utilizados por nuestro método.

En la Figura 1 se sintetiza este proceso en función de un número N dado de bandas espectrales,

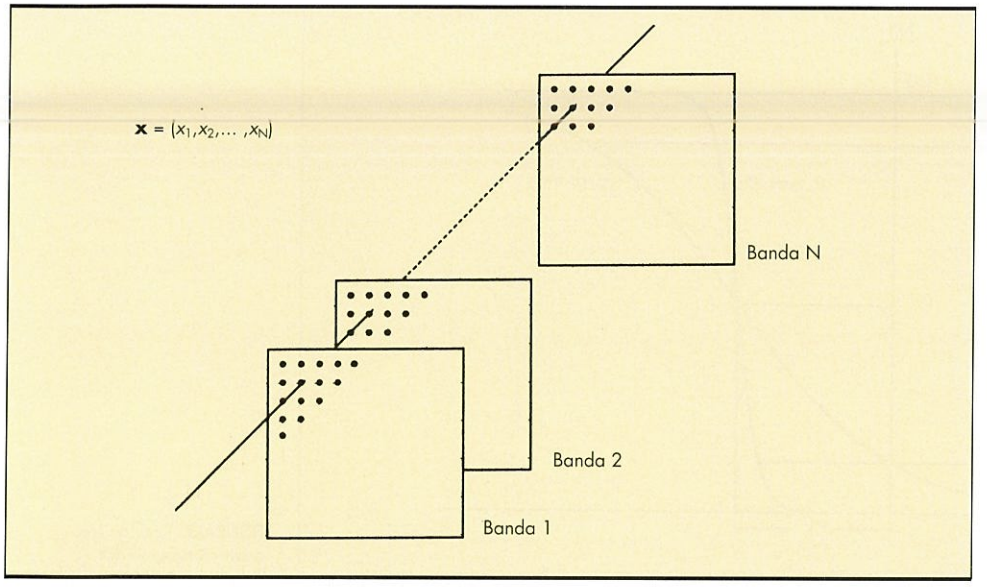

FIGURA 1. Patrón de entrada en función de las bandas espectrales dadas.

3. EL PERCEPTRÓN MULTICAPA

Vamos a considerar en este apartado una red neuronal clásica, conocida como el PM, para abordar un problema multiclase. Esta red consta de varias capas de neuronas y el mecanismo consiste en la retro-propagación de los errores (*backpropagation* en terminología inglesa). Este modelo ha sido ampliamente considerado en la teoría de reconocimiento de patrones, concretamente en Paola y Schowengerdt (1997) se hace un minucioso estudio de esta red, respecto de su aplicación a problemas de clasificación.

3.1 ARQUITECTURA DEL PERCEPTRÓN MULTICAPA

La Figura 2 muestra la arquitectura básica del modelo de red neuronal que se considera.

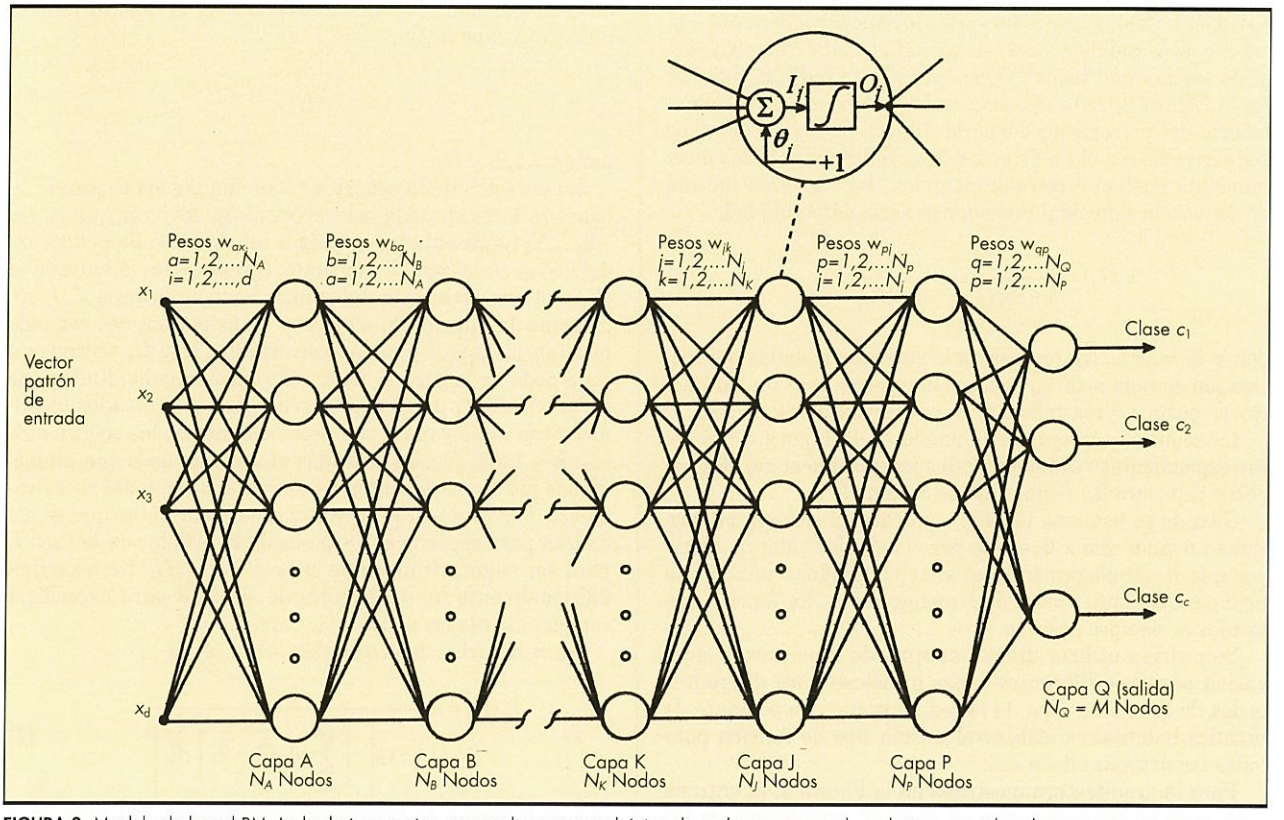

FIGURA 2. Modelo de la red PM. La burbuja superior muestra la estructura básica de cada neurona en la red, θ_j es considerado como otro peso.

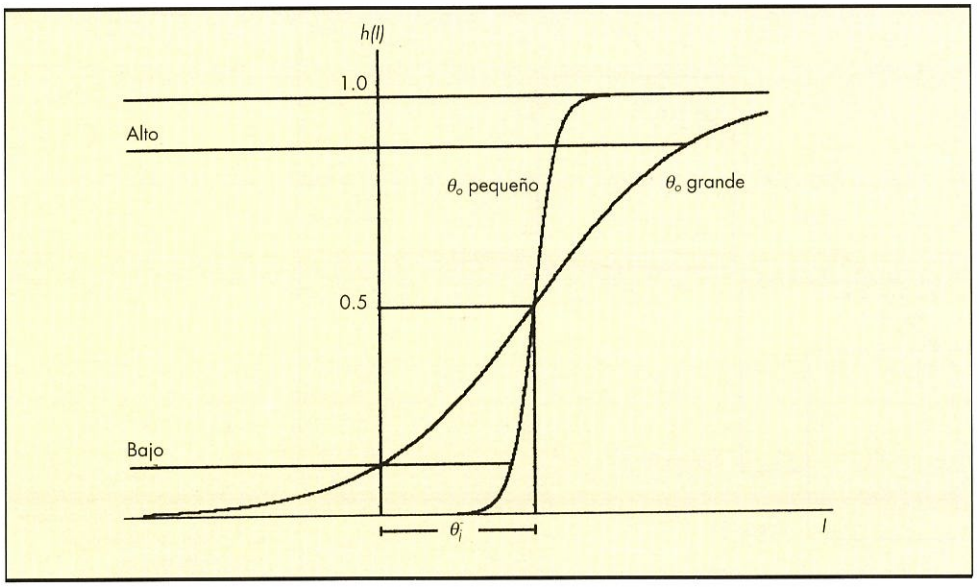

FIGURA 3. Función de activación sigmoidal.

Consta de capas de neuronas de cómputo estructuradas de forma idéntica y colocadas de forma que la salida de cada neurona en una capa proporciona la entrada de cada neurona en la siguiente capa. El número de neuronas en la primera capa, llamada capa A, es N_A. A menudo, $N_A = d$, es la dimensión de los vectores de entrada, es decir de los patrones. El número de neuronas en la capa de salida llamada capa Q, es N_Q. Este número es igual al número de clases c para las que la red se va a entrenar y sobre las que luego podrá clasificar, es decir reconocer, un determinado vector de entrada. La red reconoce un vector **x** como perteneciente a la clase c_i si la salida *i-ésima* de la red es "alta" mientras las otras salidas son "bajas". Como muestra la burbuja superior en la Figura 2, cada neurona tiene la misma forma que el modelo del perceptrón, donde la función de activación debe ser derivable con el fin de poder desarrollar la regla de entrenamiento mediante retro-propagación. La siguiente función de activación sigmoidal tiene la necesaria derivabilidad.

$$h_j(I_j) = \frac{1}{1+\exp\left[-(I_j+\theta_j)/\theta_o\right]} \qquad [4]$$

donde I_j, j=1,2,...,Nj, representa la entrada al elemento de activación de cada nodo en la capa J de la red, θ_j es un parámetro de ajuste y θ_o controla la forma de la función sigmoidal.

La ecuación (4) aparece dibujada en la Figura 3 con los correspondientes parámetros y los límites que se consideran alto y bajo para las respuestas de cada nodo.

Cuando se usa esta función sigmoidal el sistema proporciona un valor alto a la salida para cualquier valor de I_j mayor que θ_j. Análogamente, el sistema produce una salida baja para cualquier valor de I_j menor que θ_j. La función sigmoidal es siempre positiva.

Se podrían utilizar diferentes tipos de funciones de activación para las diferentes capas o incluso para diferentes nodos de la misma capa de la red neuronal. No obstante, la práctica habitual es utilizar el mismo tipo de función para todas las neuronas de la red.

Para la arquitectura mostrada en la Figura 2, la entrada a un nodo en cualquier capa es la suma promediada de las salidas que provienen de la capa previa. Suponiendo la capa K como la capa que precede a la J, la entrada al elemento de activación de cada nodo en la capa J, denominada I_j resulta,

$$I_j = \sum_{k=1}^{N_K} w_{jk} O_k \qquad [5]$$

para j = 1,2,...,N_j, donde N_j es el número de nodos en la capa J, N_K es el número de nodos en la capa K y w_{jk} son los pesos que modifican las salidas O_k de los nodos en la capa K antes de que sean suministradas a los nodos en la capa J. Las salidas de la capa K son,

$$O_k = h_k(I_k) \qquad [6]$$

para k = 1,2,...,N_k.

La notación de la ecuación (6) es fundamental porque se usa a lo largo de toda esta sección. Se advierte que I_j, j = 1,2,...,N_j representa la entrada a un *elemento de activación* del nodo j en la capa J. Por tanto, I_1 representa la entrada al elemento de activación del primer nodo en la capa J, I_2 representa la entrada al elemento de activación del segundo nodo en la capa J y así sucesivamente. Hay N_k entradas a cada nodo en la capa J, pero cada entrada individual puede ser *promediada* de forma diferente. Las N_k entradas al primer nodo en la capa J son promediadas por los coeficientes w_{1k}, k = 1,2,...,N_k; las entradas al segundo nodo son promediadas por los coeficientes w_{2k}, k = 1,2,...,N_k; y así sucesivamente. Por tanto, el número total de coeficientes que se necesitan para especificar los pesos de las salidas de la capa K para ser suministradas a la capa J es N_J xN_k. Se necesitan adicionalmente N_J coeficientes de ajuste θ_j para especificar completamente los nodos en la capa J.

La sustitución de la ecuación (5) en (4) da

$$h_j(I_j) = \frac{1}{1+\exp\left[-\left(\sum_{k=1}^{N_K} w_{jk} O_k + \theta_j\right)\Big/\theta_o\right]} \qquad [7]$$

que es la forma de la función de activación utilizada en el resto de esta sección.

Durante el entrenamiento, la adaptación de las neuronas en la capa de salida es una cuestión simple, ya que se conoce la salida de cada nodo. El principal problema en el entrenamiento de una red multicapa estriba en el ajuste o aprendizaje de los pesos en las denominadas *capas ocultas,* es decir en aquellas otras diferentes a la capa de salida.

3.2 ENTRENAMIENTO POR RETRO-PROPAGACIÓN

Comenzamos en la capa de salida. El error al cuadrado total entre las respuestas deseadas, r_q y las correspondientes respuestas actuales, O_q, de los nodos en la capa Q de salida, es

$$E_Q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2 \qquad [8]$$

donde N_Q es el número de nodos en la capa de salida Q y el factor de 1/2 se utiliza por conveniencia en la notación para tomar la derivada más tarde.

El objetivo consiste en desarrollar una regla de entrenamiento similar a la regla delta, que permita el ajuste de los pesos en cada una de las capas, buscando un mínimo a una función de error de la forma de (8). Como en el caso de la regla delta, ajustando los pesos en proporción a la derivada parcial del error con respecto a los pesos se logra este resultado. En otras palabras,

$$\Delta w_{qp} = -\alpha\frac{\partial E_Q}{\partial w_{qp}} \qquad [9]$$

donde la capa P precede a la capa Q, Δw_{qp} es tal y como se define en (10), y α es un incremento de corrección positivo.

$$w(k+1) = w(k) - \alpha(k)\sum_{i=1}^{n}\frac{(w^t x_i - fd_i)x_i}{\|x_i\|} \qquad [10]$$

donde fd_i es el valor esperado para la entrada $\mathbf{x}_i$.

El error E_Q es una función de las salidas, O_q, que en definitiva son funciones de las entradas I_q. Utilizando la regla de la cadena, podemos evaluar la derivada parcial de E_Q como sigue,

$$\frac{\partial E_Q}{\partial w_{qp}} = \frac{\partial E_Q}{\partial I_q}\frac{\partial I_q}{\partial w_{qp}} \qquad [11]$$

A partir de (5),

$$\frac{\partial I_q}{\partial w_{qp}} = \frac{\partial}{\partial w_{qp}}\sum_{p=1}^{N_P} w_{qp}O_p = O_p \qquad [12]$$

Sustituyendo estas dos últimas ecuaciones en (9) se obtiene,

$$\Delta w_{qp} = -\alpha\frac{\partial E_Q}{\partial I_q}O_p = \alpha\delta_q O_p \quad \text{donde} \quad \delta_q = -\frac{\partial E_Q}{\partial I_q} \qquad [13]$$

para obtener $\partial E_Q/\partial I_q$, usamos la regla de la cadena para expresar la derivada parcial en términos de la razón de cambio de E_Q con respecto a O_q y la razón de cambio de O_q con respecto a I_q, esto es,

$$\delta_q = -\frac{\partial E_Q}{\partial I_q} = -\frac{\partial E_Q}{\partial O_q}\frac{\partial E_q}{\partial I_q} \qquad [14]$$

De la ecuación (8)

$$\frac{\partial E_Q}{\partial O_q} = -(r_q - O_q) \qquad [15]$$

y de la ecuación (6)

$$\frac{\partial O_q}{\partial I_q} = \frac{\partial}{\partial I_q}h_q(I_q) = h_q'(I_q) \qquad [16]$$

sustituyendo (15) y (16) en (14) obtenemos,

$$\delta_q = (r_q - O_q)h_q'(I_q) \qquad [17]$$

que es proporcional a la cantidad de error (r_q - O_q). La sustitución de las anteriores ecuaciones en (13) resulta,

$$\Delta w_{qp} = \alpha(r_q - O_q)h_q'(I_q)O_p = \alpha\delta_q O_p \qquad [18]$$

después de especificar la función $h_q(I_q)$, todos los términos en (18) son conocidas o pueden ser observadas en la red. En otras palabras, tras la presentación de cualquier patrón de entrenamiento a la entrada de la red, nosotros conocemos la respuesta deseada, r_q de cada nodo de salida. El valor O_q de cada nodo de salida puede ser observado así como I_q, la entrada a los elementos de activación de la capa Q, y la salida de los nodos en la capa P, O_p. Por tanto, ya sabemos cómo ajustar los pesos que modifican las uniones entre la última y penúltima capa en la red.

Continuando hacia atrás, a partir de la capa de salida vamos a ver qué sucede en la capa P. Procediendo de la misma manera,

$$\Delta w_{pj} = \alpha(r_p - O_p)h_p'(I_p)O_j = \alpha\delta_p O_j \qquad [19]$$

donde el término de error es,

$$\delta_p = (r_p - O_p)h_p'(I_p) \qquad [20]$$

Con la excepción de r_p todos los términos en las ecuaciones (19) y (20) son conocidos o pueden ser observados en la red. El término r_p carece de sentido en una capa interna porque no se conoce la respuesta de un nodo interno en términos del patrón de la entrada, ya que sólo es posible especificar la respuesta deseada en las salidas de la red donde tiene lugar la clasificación. Si conociéramos esa información en los nodos internos, no serían necesarias tantas capas. Por tanto, necesitamos un método para obtener δ_p en términos de las cantidades que son conocidas o que pueden ser observadas en la red.

Volviendo a la ecuación (14) podemos escribir el término de error para la capa P como sigue,

$$\delta_q = -\frac{\partial E_P}{\partial I_p} = -\frac{\partial E_P}{\partial O_p}\frac{\partial O_p}{\partial I_p} \qquad [21]$$

El término $\partial O_p/\partial I_p$ no presenta dificultades, como antes, resulta

$$\frac{\partial O_q}{\partial I_p} = \frac{\partial}{\partial I_p}h_p(I_p) = h_p'(I_p) \qquad [22]$$

que es conocido una vez que se especifica h_p puesto que I_p puede ser observado. El término que produjo r_p era la deri-

vada $\partial E_p/\partial O_p$, de modo que este término debe expresarse de forma que no contenga a r_p. El término $\partial E_p/\partial O_p$ representa una cierta sensibilidad de E_P a la salida del nodo O_P. El nodo O_P exhibe su influencia sobre E_P a través de todos los nodos que le suceden (recordemos que estamos en un procedimiento de retro-propagación). Por tanto, usando la regla de la cadena y la influencia de todos los nodos de la capa Q, se tiene,

$$-\frac{\partial E_P}{\partial O_p} = -\sum_{q=1}^{N_Q} \frac{\partial E_P}{\partial I_q}\frac{\partial I_q}{\partial O_p} = \sum_{q=1}^{N_Q}\left(-\frac{\partial E_P}{\partial I_q}\right)\frac{\partial}{\partial O_p}\sum_{p=1}^{N_P} w_{qp}O_p = \tag{23}$$

$$= \sum_{q=1}^{N_Q}\left(-\frac{\partial E_P}{\partial I_q}\right)w_{qp} = \sum_{q=1}^{N_Q}\delta_q w_{qp}$$

donde el último paso se deriva de la ecuación (13). Sustituyendo las ecuaciones (22) y (23) en (21) se obtiene la expresión deseada para δ_p:

$$\delta_p = h_p'(I_p)\sum_{q=1}^{N_Q}\delta_q w_{qp} \tag{24}$$

El factor δ_p se puede obtener ahora, puesto que todos sus términos son conocidos. Entonces las ecuaciones (22) y (24) establecen completamente la regla de entrenamiento para la capa P. La importancia de la ecuación (24) es que obtiene δ_p a partir de cantidades δ_q y w_{qp}, que son términos ya obtenidos en la capa inmediatamente siguiente a la capa P. Después de que el término de error y los pesos han sido obtenidos para la capa P, esas cantidades pueden usarse para de forma similar obtener el error y los pesos para la capa inmediatamente precedente a la capa P. En otras palabras hemos encontrado una vía para propagar el error hacia atrás en la red, comenzando con el error en la capa de salida.

Podemos resumir y generalizar el proceso de entrenamiento como sigue. Para cualesquiera dos capas K y J, donde la capa K es la capa inmediatamente anterior a la capa J, obtener los pesos w_{jk}, que modifican las conexiones entre esas dos capas, usando

$$\Delta w_{jk} = \alpha\delta_j O_k \tag{25}$$

Si la capa J es la capa de salida, δ_j es

$$\delta_j = (r_j - O_j)h_j'(I_j) \tag{26}$$

Si la capa J es una capa interna y la capa P es la siguiente capa (a la derecha), entonces δ_j está dada por,

$$\delta_j = h_j'(I_j)\sum_{p=1}^{N_P}\delta_p w_{jp} \tag{27}$$

para $j=1,2,\ldots,N_j$.

Utilizando la función de activación en la ecuación (7) con θ_o=1, se obtiene,

$$h_j'(I_j) = O_j(1-O_j) \tag{28}$$

en cuyo caso (26) y (27) asumen las siguientes formas particularmente atractivas,

$$\delta_j = (r_j - O_j)O_j(1-O_j) \tag{29}$$

para la capa de salida, y

$$\delta_j = O_j(1-O_j)\sum_{p=1}^{N_P}\delta_p w_{jp} \tag{30}$$

para las capas internas. En ambas ecuaciones (29) y (30) $j=1,2,\ldots,N_j$.

Las ecuaciones (27) y (28) constituyen la *regla delta generalizada* para el entrenamiento de la red neuronal multicapa de la Figura 2. El proceso comienza con un conjunto de pesos arbitrarios a través de la red. Luego la aplicación de la regla delta generalizada en cualquier paso iterativo implica dos fases básicas. En la primera fase se presenta un vector de entrenamiento a la red y se permite que se propague a través de las capas para obtener la salida O_j para cada nodo. Las salidas O_q de los nodos en la capa de salida son comparadas con sus respuestas deseadas, r_q para generar los términos de error δ_q. La segunda fase implica un movimiento hacia atrás a través de la red, durante el cual la señal de error apropiada se pasa a cada nodo y se actualizan los correspondientes pesos. Este procedimiento también se aplica a los pesos θ_j, como se ha expuesto anteriormente en detalle, éste factor es tratado simplemente como un peso adicional que modifica una entrada con un valor de la unidad en la suma conjunta de cada nodo en la red.

Es una práctica habitual observar el error de la red, así como los errores asociados con patrones individuales. En una sesión de entrenamiento satisfactoria, el error de la red decrece con el número de iteraciones y el procedimiento converge a un conjunto estable de pesos que muestran solamente pequeñas fluctuaciones con un cierto entrenamiento adicional. El procedimiento habitual seguido para establecer si un patrón ha sido clasificado correctamente durante el entrenamiento es determinar si la respuesta del nodo en la capa de salida asociada con la clase patrón a la cual pertenecía el patrón, es alta, mientras el resto de los nodos tienen salidas bajas, como se ha definido anteriormente.

Después de que el sistema ha sido entrenado, se clasifican los patrones usando los parámetros establecidos durante la fase de entrenamiento. Operando normalmente, todas las conexiones de retro-alimentación están desconectadas. Entonces, a cualquier patrón de entrada se le permite propagarse a través de las diferentes capas de la red, y el patrón se clasifica como perteneciente a la clase cuyo valor en el nodo de salida es alto, mientras los demás son bajos. Si más de una salida es etiquetada como alta o si ninguna de las salidas es etiquetada alta, la elección es o bien declarar una mala clasificación o simplemente asignar el patrón a la clase del nodo de salida con el valor numérico más alto.

4. CONFIGURACIÓN DE LA RED Y RESULTADOS EXPERIMENTALES

Tomando como ejemplo la imagen de la Figura 4 se diferencian 4 zonas, que se asocian con otras tantas clases. Se trata pues, de una imagen multiespectral de tres bandas, concretamente las tres bandas del espectro visible RGB que corresponden a una imagen de color. En dicha imagen se identifican cuatro clases de interés: agua (c_1), rocas (c_2), masa de arbustos (c_3) y zona de hierbas secas (c_4).

FIGURA 4. Imagen multiespectral (color RGB) con cuatro clases de interés: agua (c_1), rocas (c_2), masa de arbustos (c_3) y una zona de hierbas secas (c_4).

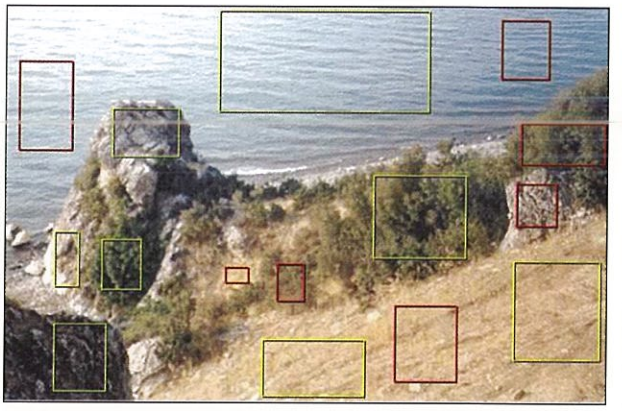

FIGURA 5. Los rectángulos amarillos contienen las muestras de entrenamiento utilizadas y los rojos las muestras a clasificar tras el entrenamiento.

A partir de las consideraciones del apartado 3.1, la arquitectura de la red consta de una capa de entrada A con tres neuronas, siendo $N_A = 3$, la dimensión de los vectores de entrada, es decir de los patrones. El número de neuronas en la capa de salida Q, es $N_Q = 4$, este número es igual al número de clases c para las que la red se va a entrenar. Los valores R_i definidos por la ecuación (3) se deben calcular sobre un entorno de vecindad alrededor de cada localización (x,y), en nuestros experimentos hemos probado varios tamaños de vecindades, que se corresponden con ventanas centradas en (x,y) y con tamaños desde $3x3$ hasta $41x41$. Tras los resultados experimentales, se ha observado que si la ventana es muy pequeña (dimensión $3x3$), no se capturan correctamente las propiedades de textura que se desean y si es muy grande (dimensión $41x41$) se dispara el coste computacional, sin conseguir mejoras aparentes en los resultados. En este trabajo hemos obtenido unos buenos resultados con ventanas de dimensión $15x15$, que ha sido finalmente la dimensión seleccionada.

Resulta conocido para la comunidad científica que este tipo de redes entrañan ciertas dificultades, por ejemplo, la elección del número de capas ocultas, el número de neuronas en cada capa así como la elección de los parámetros, en particular la razón de aprendizaje constituye siempre un problema, que está sin resolver hasta la fecha (Bishop 1996, Chan y col. 2001). Lippman (1987) sugiere que una arquitectura con no más de dos capas ocultas puede ser suficiente, mientras que Bishop (1996) llega a hacer algunos estudios para concluir que una arquitectura con una sola capa oculta y un elevado número de neuronas en dicha capa oculta es también suficiente. Paola y Schowengerdt (1997) concluyen que redes con capas ocultas con tres o menos neuronas no realizan bien la clasificación. En ninguno de los casos anteriores se llega a proporcionar el número óptimo ni de capas ocultas ni de neuronas en las capas ocultas. No obstante, teniendo en cuenta los criterios anteriores y basándonos en la experiencia, hemos optado por elegir dos capas ocultas, con seis veces más neuronas en cada capa que las neuronas de las capas externas con las que conectan, es decir, 18 neuronas para la primera capa oculta, que está en contacto con la capa de entrada y 24 neuronas para la segunda capa oculta que está en contacto con la capa de salida. Insistimos de nuevo, que estos números no deben ser una referencia universal y sólo la experiencia puede llegar a demostrar cuáles son los números óptimos.

Otro de los puntos abiertos en este tipo de redes consiste en seleccionar el valor inicial de los pesos de conexión, normalmente se hace de forma aleatoria. Este hecho no implica más problemas que aumenta ligeramente el número de iteraciones sin mayores consecuencias. En este trabajo se han inicializado todos a la unidad.

Basándonos en el trabajo de Heermann y Khazenie (1992), hemos experimentado una estrategia adaptativa para seleccionar la razón de aprendizaje α. De este modo, la razón de aprendizaje se ajusta automáticamente tras una serie de iteraciones, se disminuye su valor si el error total cometido en todas esas iteraciones ha aumentado con respecto al error calculado previamente tras otra serie de iteraciones y se aumenta su valor en caso contrario. En nuestros experimentos, no hemos obtenido mejoras apreciables, es más, hemos comprobado que la red converge prácticamente para cualquier valor de la razón de aprendizaje en el intervalo [0,1], habiéndose elegido 0.5 por ser un valor intermedio. A este respecto, conviene advertir que no hemos detectado la inconsistencia apuntada por algunos autores en el sentido de que diferentes razones de aprendizaje conducen a pesos diferentes.

Por otro lado, los diferentes parámetros de ajuste θ_j se han fijado al valor 0.5, respecto del cual no se ha observado mejora alguna variando sus valores en el intervalo [0,1], finalmente el parámetro θ_o que controla la forma de la función sigmoidal, toma el valor de 0.2 en el rango [0,1], se ha elegido este valor relativamente bajo ya que se ha comprobado que cuanto menor es este valor menor es también el número de iteraciones necesario para alcanzar la convergencia. En esta última apreciación coincidimos con Bertels y col. (2000). En el ejemplo que presentamos se ha conseguido la convergencia en 15235 iteraciones. A este resultado se ha llegado considerando que el proceso iterativo se detiene cuando las fluctuaciones entre dos iteraciones consecutivas de todos los pesos están por debajo de un umbral, que en nuestro caso ha sido fijado al valor de 0.001.

La imagen original tiene una dimensión de 558 x 368 pixels por cada una de las tres bandas, esto significa que el número de patrones disponibles es de 205344. No obstante, sólo se ha utilizado un subconjunto de ellos. En la imagen de la Figura 5, los rectángulos amarillos contienen las muestras de entrenamiento utilizadas y los rojos las muestras a clasificar tras el entrenamiento para las cuatro clases. Estas muestras son seleccionadas por el usuario.

	Entrenamiento	Clasificación
c_1 *(agua)*	18300	7500
c_2 *(rocas)*	6400	2000
c_3 *(arbustos)*	7800	3100
c_4 *(hierbas secas)*	12000	4300

TABLA 1. Número de patrones utilizados para el entrenamiento y la clasificación de las cuatro clases.

En la Tabla 1 se muestra el número de patrones utilizados para cada una de las cuatro clases, tanto para el entrenamiento como para la clasificación posterior.

Con el propósito de poder comparar los resultados obtenidos con el método LVQ, a continuación introducimos una breve reseña de dicho método. Se trata de encontrar un vector representante $\mathbf{m}_j$ de cada una de las cuatro clases, donde $j = 1,..,4$. Las ecuaciones básicas para obtener los cuatro vectores representantes vienen dadas por,

$\mathbf{m}_j(k+1)=\mathbf{m}_j(k)+\alpha_j(k)[\mathbf{x}(k)-\mathbf{m}j(k)]$ si $\mathbf{x}$ se clasifica correctamente
$\mathbf{m}_j(k+1)=\mathbf{m}_j(k)-\alpha_j(k)[\mathbf{x}(k)-\mathbf{m}j(k)]$ si $\mathbf{x}$ se clasifica incorrectamente
$\mathbf{m}_j(k+1)=\mathbf{m}_j(k)$ de otro modo

donde Kohonen (1995) propone calcular la razón de aprendizaje de forma óptima como sigue,

$$\alpha_j(k)=\frac{\alpha_j(k-1)}{1+s(k)\alpha_j(k-1)}$$

$s(k)=+1$ si la clasificación es correcta, y $s(k)=-1$ si la clasificación es incorrecta. Se debe tener cierta precaución con $\alpha_j(k)$, ya que puede aumentar, siendo importante que su valor no sobrepase la unidad. Para los valores iniciales de $\alpha_j(k)$ se podrían elegir 0.5, pero siguiendo la recomendación de Kohonen (1995), en este trabajo los hemos elegido con un valor de 0.3. En base a la ecuación anterior la razón de aprendizaje decrece si la muestra se clasifica correctamente y crece en caso contrario.

Una vez terminado el proceso de aprendizaje utilizamos un criterio de mínima distancia para determinar a qué clase pertenece un nuevo patrón $\mathbf{x}$, que se presenta a la entrada. Una de las distancia más apropiadas es la distancia de Mahalanobis al cuadrado entre $\mathbf{x}$ y $\mathbf{m}_j$ (Duda y col. 2001),

$$d_M^2(x,m_j)=(x-m_j)^t C_j^{-1}(x-m_j)$$

ya que a través de la matriz de covarianza incluimos una medida de dispersión de las muestras dentro de la clase y además tenemos información de la correlación existente entre los diferentes atributos del vector $\mathbf{x}$. Esta matriz de covarianza se calcula de la siguiente manera,

$$C_j=\frac{1}{N-1}\sum_{i=1}^{N}(x_{ij}-m_j)(x_{ij}-m_j)^t$$

donde N es el número de muestras de entrenamiento $\mathbf{x}_i$ pertenecientes a cada una de las clases j con centro $\mathbf{m}_j$, t indica transpuesta.

El criterio de clasificación es dado un $\mathbf{x}$, para cada una de las $\mathbf{m}_j$ calcular la correspondiente distancia y clasificar $\mathbf{x}$ como perteneciente a la clase que proporcione la mínima distancia.

En la Tabla 2 se muestra el porcentaje de aciertos obtenidos con los dos métodos descritos para las cuatro clases involucradas con las muestras de clasificación dadas en la Tabla 1. Además refleja los porcentajes de aciertos cuando los vectores de entrada son los valores de textura calculados en la vecindad de un pixel (x,y) y las simples componentes espectrales RGB.

A la vista de los resultados obtenidos, resulta evidente que los mejores resultados se obtienen con el PM, frente a la técnica LVQ. Esta conclusión es opuesta a la obtenida por Lieberman y Patil (1997) en el sentido de que en la clasificación de las impurezas del algodón, el método LVQ obtiene mejores resultados que el PM. Aunque se trata del mismo problema de clasificación, queda claro, que la distinta naturaleza de las estructuras a clasificar y por consiguiente la diferente naturaleza de las imágenes hace que un determinado método sea más o menos efectivo en determinadas aplicaciones. No obstante, la arquitectura del PM es mucho más compleja que la del método LVQ y requiere un mayor número de parámetros a considerar, por lo que esto debe tenerse en cuenta a la hora de su elección. Además, el proceso de entrenamiento de PM es mucho más lento que LVQ.

5. CONCLUSIONES

Se ha hecho un estudio pormenorizado sobre la topología de la red PM y los parámetros involucrados, proporcionando de este modo una valiosa información para clasificaciones de escenas naturales. Se han contrastado y validado las consideraciones propuestas por otros autores en relación a la topología y parámetros de la red PM.

Además, en este estudio se muestra que la red PM obtiene mejores resultados en la clasificación que el método LVQ, si bien se insiste en que el grado de complejidad para su implementación es mayor en PM que en LVQ.

	Perceptrón Multicapa		Learning Vector Quantization	
	Textura	RGB	Textura	RGB
c_1 *(agua)*	95%	91%	94%	94%
c_2 *(rocas)*	83%	82%	80%	77%
c_3 *(arbustos)*	86%	81%	81%	80%
c_4 *(hierbas secas)*	97%	96%	93%	90%

TABLA 2. Porcentaje de aciertos en la clasificación de las muestras de la Tabla 2.

Por otro lado, la combinación de información de textura con información espectral es más eficiente que el uso de sólo información espectral. Gracias a esto, el método propuesto resulta apropiado para el tipo de escenas naturales en el que se ha basado este estudio. Este método es ampliable a cualquier otro tipo de imágenes donde la información se contiene en varias bandas espectrales como es el caso de imágenes procedentes de sensores remotos tales como imágenes de satélite, lo que le confiere un carácter universal para clasificaciones temáticas en ese tipo de imágenes (bosques, lagos, cultivos, zonas áridas, etc.).

BIBLIOGRAFÍA

AUGUSTIJIN, M.F., CLEMENS, L.E. and SHAW, K.A. (1995). Performance of texture measures for ground cover identification in satellite images by means of a neural network classifier. *IEEE Transactions on Geoscience and Remote Sensing,* 33(3), 616-626.

BENEDIKTSSON, J.A., SWAIN, P.H. and ERSOY, O.K. (1990). Neural network approaches versus statistical methods in classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing,* 28(4), 540-552.

BERTELS, K. NEUBERG, L., VASSILIADIS, S. y PECHANEX, G. (2000). A Look Inside the Learning Process of Neural Networks. *Complexity,* 5(6), 34-38.

BISCHOF, H., SCHNEIDER, W. and PINZ, A. J. (1992). Multiespectral classification of Landsat images using neural networks. *IEEE Transactions on Geoscience and Remote Sensing,* 30(3), 482-490.

BISHOP, C. M. (1996). *Neural Networks for Pattern Recognition.* Oxford University Press, London.

BRUZZONE, L., CONESE, C. MASELLI, F. and ROLI, F. (1997). Multisource classification of complex rural areas by statistical and neural network approaches. *Photogrammetric Engineering & Remote Sensing,* 63(5), 523-533.

CHAN, J.C.W., CHAN, K.P. and YEH, A.G.O. (2001). Detecting the Nature of Change in an Urban Environment: A Comparison of Machine Learning Algorithms. *Photogrammetric Engineering & Remote Sensing,* 67(2), 213-225.

CIVCO, D.L. (1993). Artificial neural networks for land-cover classification and mapping. *Int. J. Geographical Information Systems,* 7(2), 173-186.

DUDA, R.O., HART, P.E. and STORK, D.G. (2001). *Pattern Classification.* Wiley, New York.

FU, K.S., GONZALEZ, R.C. y LEE, C.S.G. (1988). *Robótica: Control, detección, Visión e Inteligencia.* McGraw-Hill, Madrid.

GONG, P. (1996). Integrated analysis of spatial data from multiple sources: Using evidential reasoning and artificial neural network techniques for geological mapping. *Photogrammetric Engineering & Remote Sensing,* 62(5), 513-523.

GONZALEZ, R. C. y WOODS, R.E. (1993). *Digital Image Processing,* Addison-Wesley, Reading, MA.

HEERMANN, P.D. and KHAZENIE, N. (1992). Classification of multiespectral remote sensing data using a back-propagation neural network. *IEEE Transactions on Geoscience and Remote Sensing,* 30(1), 81-88.

JI, C.Y. (2000). Land-Use Classification of Remotely Sensed Data Using Kohonen Self-Organizing Feature Map Neural Networks. *Photogrammetric Engineering & Remote Sensing,* 66(12), 1451-1460.

KOHONEN, T. (1995). *Self-Organizing Maps.* Springer-Verlag, Berlin.

LEE, J., WEGER, R.C., SENGUPTA, S.K. and WELCH, R.M. (1990). A neural network approach to cloud classification. *IEEE Transactions on Geoscience and Remote Sensing,* 28(5), 846-855.

LIEBERMAN, M. A. y PATIL R. B. (1997). Evaluation of Learning vector Quantization to classify cotton trash. *Optical Engineering,* 36 (3), 914-921.

LIPPMANN, R.P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine,* 4(2), 4-22.

MARAVALL, D. (1993), *Reconocimiento de Formas y Visión Artificial,* RA-MA, Madrid.

NALWA, V.S. (1993). *A Guided tour of Computer Vision.* Addison-Wesley, Reading: MA

PAJARES, G. y CRUZ, J.M. (2001). *Visión por Computador: Imágenes digitales y Aplicaciones.* RA-MA, Madrid.

PAOLA, J.D. and SCHOWENGERDT, R.A. (1997). The effect of neural network structure on a multispectral land-use/land-cover classification. *Photogrammetric Engineering & Remote Sensing,* 63(5), 535-544.