Clasificación de Imágenes Usando Redes Neuronales*

Ivan Lizarazo, Samuel Mesa, Ricardo Cuitiva

Ingeniería Catastral y Geodesia
Universidad Distrital Francisco José de Caldas

Correo-e: ilizarazo@udistrital.edu.co, samuel_icg@linuxmail.org

Resumen

El presente trabajo examina los conceptos teóricos que permiten entender la utilidad de las redes neuronales multicapa en la clasificación de imágenes. Igualmente, en él se explica el algoritmo matemático del método de retropropagación que puede utilizarse para entrenar dichas redes y clasificar patrones que no pueden separarse mediante una superficie lineal. Los resultados de la aplicación del algoritmo usando un programa de álgebra computacional son comparados con la respuesta obtenida mediante el uso de un simulador de redes neuronales. Finalmente, se argumenta que el uso de las redes neuronales es un alternativa sólida para resolver problemas complejos de clasificación.

1 Introducción

El uso de algoritmos estadísticos para realizar clasificación es una técnica muy popular en teledetección. La mayoría de los usuarios tradicionales obtienen la clasificación de las imágenes usando el método de máxima probabilidad. El proceso de decisión asigna cada pixel a aquella clase en la cual la probabilidad de pertenencia sea mayor, sobre la base de una distribución estadística determinada a partir de unos pixeles cuya clase se conoce y que sirven de »entrenamiento». Esta regla de decisión estadística tiene una contraparte geométrica que permite entender que un pixel se clasifica sobre la base de su posición en un espacio multiespectral respecto a una superficie de separación. Este hecho permite adoptar una interpretación geométrica de la clasificación sin necesidad de partir de métodos o modelos estadísticos.

2 Discriminación Lineal

2.1 Concepto de un Vector de Pesos

Considere el espacio bi-espectral de dos clases como se muestra en la Figura 1, en el que una linea recta permite separar las dos clases. Esta línea recta puede extenderse a un espacio multidimensional y constituir un *hiperplano* o superficie de decisión para clasificación multiespectral. En dos dimensiones, la ecuación del hiperplano puede ser expresada como

$$w_1\, x_1 + w_2\, x_2 + w_3 \;=\; 0 \tag{1}$$

donde x_i son las coordenadas del nivel digital del espacio multiespectral y w_i son constantes, usualmente llamadas *pesos*. Existen tantos pesos como número de bandas existan en los datos, más uno. En general, para un número de bandas N, la ecuación de la superficie lineal es

$$w_1 x_1 + w_2 x_2 + \ldots + w_N x_N + w_{N+1} \;=\; 0 \tag{2}$$

que puede ser escrita como

$$\boldsymbol{w}^t \boldsymbol{x} + w_{N+1} \;=\; 0 \tag{3}$$

donde $\boldsymbol{x}$ es el vector de coordenadas y $\boldsymbol{w}$ es el vector de pesos.

*. This document has been written using the GNU TEXMACS text editor (see www.texmacs.org).

En la vida real la posición de la superficie de separación no se conoce inicialmente. El entrenamiento de un clasificador lineal lo que hace es determinar un conjunto apropiado de pesos que coloque la superficie de decisión de manera correcta entre los dos conjuntos de muestras de entrenamientos. No existe necesariamente una solución única, sino que existe un número infinito de diferentes hiperplanos de decisión que podrían ser capaces de separar las dos clases.

Para un conjunto de datos determinado, una ecuación explicita de la superficie de separación puede ser obtenida usando la regla de distancia mínima, que se basa en los vectores promedio que definen las dos clases. Un método alternativo consiste en seleccionar una superficie arbitraria y luego iterar hasta encontrar una posición aceptable. Aunque este último método no es muy usado, es útil tomarlo como base para entender algunos de los conceptos utilizados en redes neuronales.

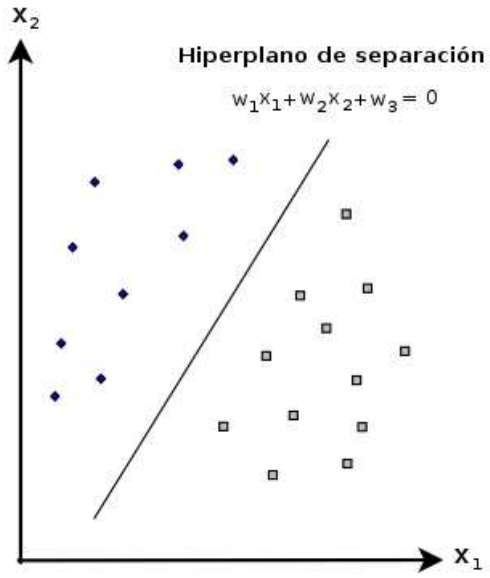


Figura 1. Espacio multiespectral de dos dimensiones, con dos clases de pixel que pueden ser separadas por una superficie lineal

2.2 Evaluación de Membresía de Clase

El cálculo de (3) será únicamente cero cuando los valores de x estén sobre la superficie de decisión. En la Figura 1, los pixeles de una clase están a la izquierda de ese plano y los de la otra clase a su derecha. Así que, una vez que se identifica una superficie de decisión a través de un entrenamiento, la regla de decisión se expresa como

$$\begin{aligned} x \in \text{clase}\,1 \quad &\text{si} \quad \boldsymbol{w}^t\boldsymbol{x} + w_{N+1} > 0 \\ x \in \text{clase}2 \quad &\text{si} \quad \boldsymbol{w}^t\boldsymbol{x} + w_{N+1} < 0 \end{aligned} \tag{4}$$

2.3 Entrenamiento

Es posible definir un vector aumentado de acuerdo a

$$y \;=\; [x^t, 1]^t$$

Si en (3) se incluye el término w_{N+1} dentro del vector de pesos, se obtiene

$$w = [w^t, w_{N+1}]^t \tag{5}$$

de manera que la superficie de decisión se puede expresar como

$$\boldsymbol{w}^t y \;=\; 0 \tag{6}$$

de manera que la regla de decisión en (4) se puede plantear de otra manera

$$\begin{array}{rl} x \in \text{clase } 1 \quad \text{si} \quad \boldsymbol{w}^t y > 0 \\ x \in \text{clase2} \quad \text{si} \quad \boldsymbol{w}^t y < 0 \end{array} \tag{7}$$

Es usual pensar que $\boldsymbol{w}^t \boldsymbol{y} = 0$ define una superficie lineal en un espacio multiespectral x (ahora definido como y), en el cual los coeficientes de las variables (y_1, y_2, etc.) son los pesos $w_1, w_2,$ etc. Sin embargo, es también posible interpretar que la ecuación describe una superficie lineal en la cual y son los coeficientes y w son las variables. Esta superficies ocurre en un espacio de coordenadas que tiene ejes w_1, w_2, etc. Una versión bidimensional de este *espacio de pesos*, como se podría denominar, se muestra en la Figura 2, en la cual se han graficado un número de *hiperplanos patrones*; estas son superficies lineales que pasan a través del origen y que tienen como sus coeficientes, los componentes de los vectores de pixel (aumentados). Así, mientras en un espacio multiespectral, los pixeles están representados por puntos, en un espacio de pesos, ellos aparecen como superficies lineales. De la misma manera, un conjunto de coeficientes de pesos definirá una superficie en un espacio multiespectral pero será un punto en un espacio de pesos.

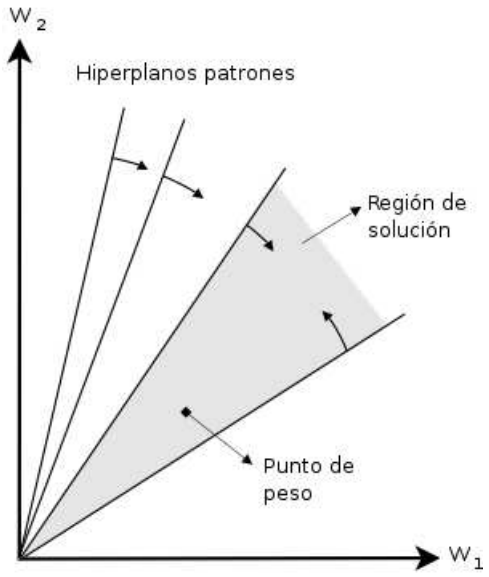


Figura 2. Representación de los pixeles como hiperplanos y de los vectores pesos como puntos, en un "espacio de pesos". La flechas indican el lado de cada plano de pixel en el que los pesos deben caer para una clasificación correcta.

En el espacio de pesos, la regla de decisión dada en (7) también aplica, sin embargo, ahora verifica que el *punto de peso* esté en el lado apropiado del *hiperplano del patrón*. Por ejemplo, en la Figura 3 se muestra un simple punto de peso que está en el lado correcto de cada pixel y de esta manera define una superficie de decisión útil en el espacio multiespectral. En el diagrama, las flechas son graficadas a cada hiperplano del pixel para indicar el lado en el cual el punto de peso debe caer para que la prueba de la regla de decisión (7) sea exitosa para todos los pixeles. El propósito de entrenar el clasificador lineal es asegurar que el punto que define el peso esté localizado dentro de alguna parte de la *región de solución*. Si, a través de algunos ensayos iniciales, el punto que representa el peso se localiza en alguna parte diferente del espacio de pesos lo que hay que hacer es moverlo para que caiga en la región de solución.

Suponga que se hace una estimación inicial del vector de pesos $\boldsymbol{w}$, pero que al hacerlo el punto que representa el peso está en el lado equivocado de un hiperplano de un pixel particular como se ilustra en la Figura 3.

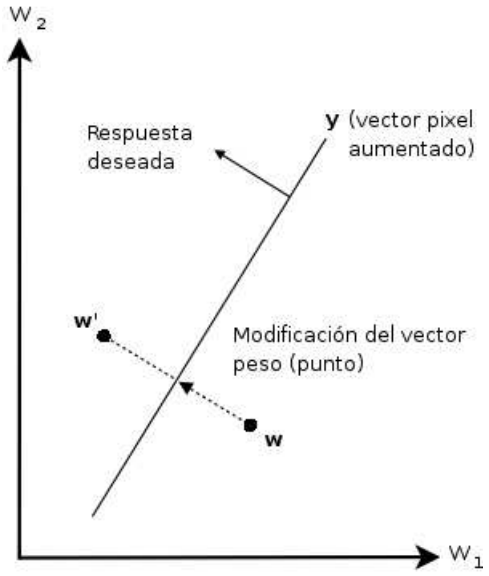


Figura 3. Modificación del punto de peso para obtener una respuesta correcta.

En la figura el punto se tiene que mover al otro lado del vector de pixel para dar una respuesta correcta de acuerdo con la regla de decisión (7). La manera directa en el cual el punto se puede modificar es moverlo atravesando el hiperplano que representa el pixel. Esto puede ser alcanzado agregando una cantidad escalada del vector pixel al vector de pesos. La nueva posición del punto de pesos es entonces

$$w' = w + c\,y \tag{8}$$

donde c es llamado el incremento de la corrección, cuyo valor está determinado por la cantidad del desplazamiento del punto de peso original de manera perpendicular al hiperplano del pixel. Si ese incremento es suficientemente grande el punto de peso será movido a través del plano del pixel tal como se requiere. Cuando se ha modificado el vector de pesos, el producto en (7) es

$$\begin{aligned} w^{'t}\,y &= w^t\,y + c\,y^t\,y \\ &= w^t y + c|y|^2 \end{aligned}$$

Es claro que si el valor inicial $w^t y$ fue erróneamente negativo, entonces un valor positivo de c le dará un valor positivo a $w'^t y$; de otra modo, un valor negativo de c corregirá un valor positivo inicial erróneo del producto.

Usando la prueba de membresia de clase (7) y la formula de corrección (8) se puede realizar un procedimiento de entrenamiento no paramétrico iterativo, que se conoce como retroalimentación por corrección de error (error correction feedback).

Primero, se escoge arbitrariamente un punto pesos de en una posición inicial. Luego, los vectores que representan los pixeles son examinados de a uno cada vez. Si la posición del peso actual clasifica un pixel correctamente entonces no se hace nada, pero si no lo hace hay que modificarlo -usando (8)- con respecto a ese vector de pixeles. Este procedimiento se tiene que repetir para cada pixel dentro del conjunto de entrenamiento, y el conjunto total de vectores se debe examinar tantas veces como se requiera para ir moviendo el punto de peso dentro de la región de solución. Si las clases se pueden separar linealmente entonces se alcanza una solución.

2.4 Especificación en el Incremento de la Corrección

Existen varias maneras de escoger el valor del incremento de la corrección, c. La más simple es igualar el valor de c a una constante positiva o negativa (de acuerdo al cambio que se requiera en el producto $\boldsymbol{w}^t\boldsymbol{y}$). Un valor muy usado es tomar $c = \pm 1$ de manera que la aplicación de (8) simplemente lo que hace es adicionar o restar el valor del vector del pixel y de esta manera se hace un entrenamiento muy rápido.

Otra posibilidad es seleccionar un incremento de corrección proporcional a la diferencia entre la respuesta del clasificador y el valor deseado

$$c = \eta(t - w^t y)$$

de manera que (8) puede escribirse

$$\begin{aligned} w' &= w + \Delta w \\ \text{con} \qquad \Delta w &= \eta(t - w^t y)y \end{aligned} \tag{9}$$

donde t es la respuesta deseada del patrón de entrenamiento y y $w^t y$ es la respuesta actual; η es un valor que controla el grado de corrección aplicada. Usualmente t se escoge como $+1$ para una clase y -1 para la otra.

2.5 Clasificación - La Unidad Lógica de Umbral (TLU)

Después de entrenar un clasificador lineal de dos categorías, se obtiene la versión del vector de pesos final w, que se puede utilizar con pixeles nuevos para adjudicarles una etiqueta de clase, aplicando la regla de decisión que se indica en (7). La regla de clasificación se puede representar como se muestra en la Figura 4a. En el diagrama consiste de varios elementos que tienen un peso, otro elemento que hace una sumatoria y un elemento de salida que realiza la selección máxima. Estos elementos unidos se conocen como Unidad Lógica de Umbral (*Treshold Logic Unit (TLU)).* Este concepto es similar al de un *procesador elemental (PE)* utilizado en redes neuronales, pero en ella el umbral de salida se reemplaza por una función más general y la ruta de la unidad de entrada de patrones aumentado es realmente incorporado dentro de la función de salida. En la Figura 4b, el elemento de umbral simple ha sido reemplazado por un bloque funcional que realiza la adición del coeficiente de peso final a la suma ponderada de los componentes de pixel de entrada y luego realiza una operación de umbral o una operación no lineal más general.

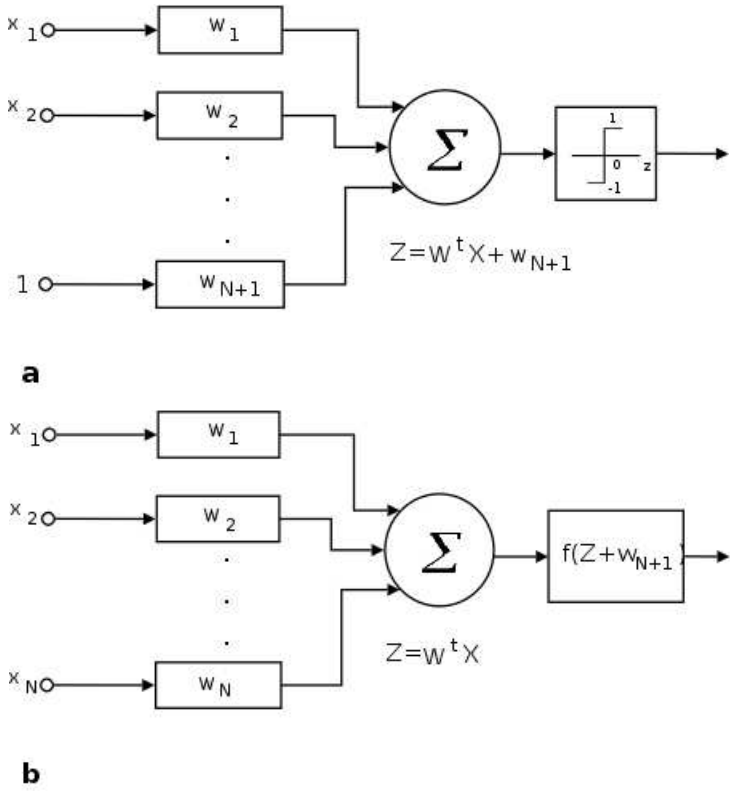


Figura 4. **a** Una representación gráfica de (7). **b** Representación más usual de un elemento de procesamiento en donde la función de umbral se ha generalizado.

3 La aproximación de Redes Neuronales

La estructura de nodos de la Figura 4b puede conducir a un teorema de entrenamiento de clasificación no lineal multicategoria, siempre y cuando el elemento de salida no aplique una operación de umbral a las entradas ponderadas sino que aplique una operación asintótica y de esta forma matemáticamente diferenciable.

3.1 Procesador Elemental (PE)

El nodo de procesamiento esencial en una red neuronal es un elemento como el mostrado en la Figura 5a. La operación es descrita como

$$o \;=\; f\big(\boldsymbol{w}^t\boldsymbol{x}+\theta\big) \tag{10}$$

donde θ es un umbral (algunas veces igualado a cero), $\boldsymbol{w}$ es un vector de coeficientes de pesos y $\boldsymbol{x}$ es el vector de entradas. Para el caso especial en donde las entradas son los valores de las bandas de un vector de pixeles multiespectrales, podría ser recomendado que el valor de umbral θ tome el valor del coeficiente de pesos w_{N+1} de (2). Si la operación f es una operación de umbral, este elemento de procesamiento se podría comportar como una TLU. En general, el numero de entradas a un nodo será definido por la topología de la red lo mismo que por la dimensionalidad de los datos.

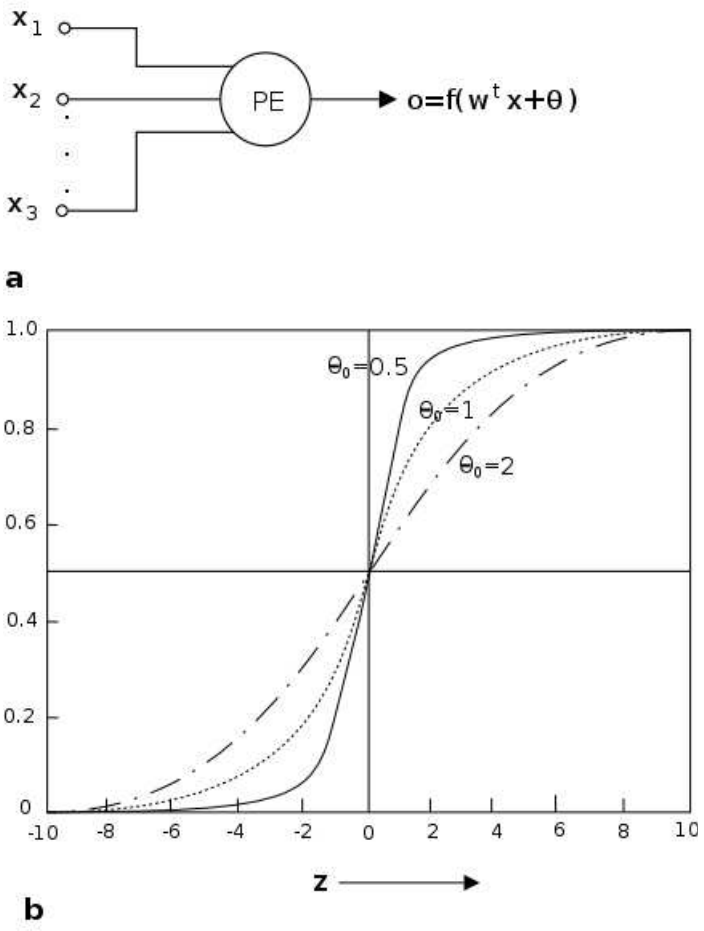


Figura 5. **a** Procesador elemental de una red neuronal. **b** Gráficas de (11) para diferentes valores de θ_0

La expresión más común para la función de activación es

$$f(z) \;=\; \frac{1}{1+e(-z/\theta_0)} \tag{11}$$

donde el argumento de z es $w^t x + \theta$ como se indicó en (10) y θ_0 es una constante. Esto asegura que el valor sea 1 para z muy grandes y positivo y 0 para z grandes y negativos y de esta manera es un umbral asintótico. La salida del producto $w^t x$ es un escalar simple, cuando se gráfica con $\theta = 0$ en (11) y como aparece en la Figura 5b. Para θ_0 muy pequeños la función de activación se aproxima a una función de umbral, generalmente $\theta_0 = 1$.

Una red neuronal para uso en el análisis de imágenes de sensores remotos será como la que aparece en Figura 6, la cual es un clasificador en capas compuesto de procesadores elementales del tipo mostrado en la Figura 5a. Existe una capa de nodos de entrada que tiene la función de distribuir las entradas a los elementos de procesamiento de la siguiente capa, y si es necesario escalarlos, y una capa de salida en la cual se va a proporcionar la información de la etiqueta de las diferentes clases. Usualmente una sola capa oculta puede ser suficiente, aunque el número de nodos de la capa oculta no se puede determinar con facilidad.

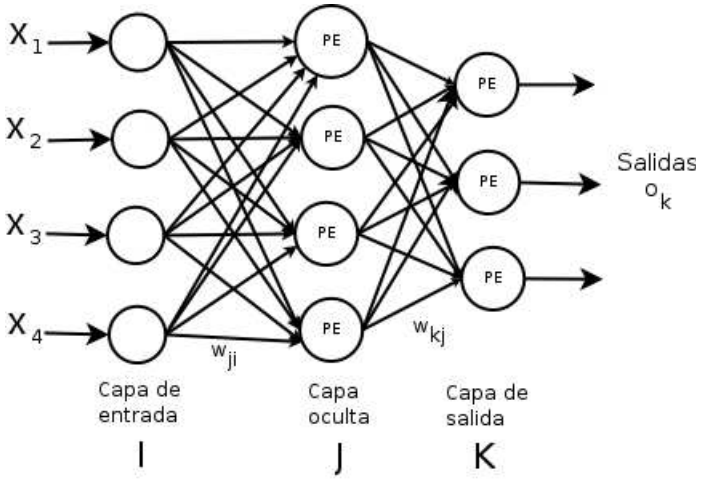


Figura 6. Una red neuronal perceptron multicapa, y la nomenclatura usada para la derivación del algoritmo de backpropagation

3.2 Entrenamiento de la Red Neuronal - Retropropagación (backpropagation)

Antes de que se pueda desarrollar una clasificación, la red de la Figura 6 tiene que ser entrenada. Esto permitirá determinar el vector de pesos w y el umbral θ en (10) para cada procesador elemental conectado a la red. Se puede observar que la constante θ_0 en (11) que gobierna el gradiente de la función de activación como se vio en la Figura 5b, está generalmente predefinida y no necesita ser estimada a partir del uso de los datos de entrenamiento.

La Figura 6 indica la nomenclatura usada. Las tres capas son numeradas como I, J, K donde K es la salida. El conjunto de pesos que conecta los procesadores elementales de la capa i con aquellos de la capa j generalmente están representados por w_{ji} y aquellos pesos entre la capa j y k se representa por w_{kj}. Siempre habrá un número muy grande de estos pesos, pero para derivar el algoritmo de entrenamiento no es necesario referirse a ellos individualmente. De manera similar la función de activación general tiene como argumento z_i y salidas o_i y puede ser utilizada para representar todos los argumentos y las salidas en la capa correspondiente.

Para los PE j y k (10) es

$$o_j = f(z_j) \quad \text{con} \quad z_j = \sum_j w_{ji} o_i + \theta_j \tag{12}$$

$$o_k = f(z_k) \quad \text{con} \quad z_k = \sum_k w_{kj} o_j + \theta_k \tag{13}$$

La sumatoria en (12) y (13) se muestra con respecto a los índices j y k. Esto debe entenderse como la sumatoria de todas las entradas de los PEs de las capas J y K respectivamente. Las sumatorias son expresadas en términos de las salidas de la capa previa dado que estas salidas constituyen las entradas de los PE en cuestión.

Una red no entrenada o entrenada pobremente dará salidas erróneas. De manera, que una medida para evaluar que tan bien está funcionando una red durante el entrenamiento puede estimarse a partir de las salidas de la capa anterior (K). Una medida útil es usar la sumatoria de los errores de salida al cuadrado. El error cometido por la red puede ser expresado como

$$E = \sum_k (t_k - o_k)^2 \tag{14}$$

donde t_k representa la salida deseada o la salida objetivo, y o_k representa la salida real del PE de la capa de salida en respuesta a los pixeles de entrenamiento. La suma se realiza sobre todos los PEs de la capa de salida.

Una estrategia de entrenamiento muy utilizada consiste en ajustar los pesos en los PE hasta que el error se minimice, estado en el cual las salidas reales son muy cercanas a las salidas deseadas.

Una aproximación muy común para ajustar los pesos y minimizar el valor de una función del cual ellos son argumentos, es modificar los valores proporcionalmente al negativo de la derivada parcial de la función. Esto se llama la técnica del descenso del gradiente. De esta manera para los pesos que conectan las capas J y K se obtiene

$$\text{con} \qquad \begin{aligned} w'_{kj} &= w_{kj} + \Delta w_{kj} \\ \Delta w_{kj} &= -\eta \frac{\partial E}{\partial w_{kj}} \end{aligned}$$

donde η es una constante positiva que controla la cantidad de ajuste. Esto requiere una expresión para la derivada parcial, que puede ser determinada con la regla de la cadena

$$\frac{\partial E}{\partial w_{\mathrm{kj}}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}} \tag{15}$$

cada término de los cuales debe ser evaluado.

Para (13) y (11) tomando como $\theta_0 = 1$, se obtiene

$$\frac{\partial o_k}{\partial z_k} = f'(z_k) = (1 - o_k)\, o_k \tag{16}$$

$$y \qquad \frac{\partial z_k}{\partial w_{kj}} = o_j \tag{17}$$

tomando la Ecuación (14) se obtiene

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \tag{18}$$

Así que la corrección que debe aplicarse a los pesos es

$$\Delta w_{kj} = \eta (t_k - o_k)(1 - o_k)\, o_k\, o_j \tag{19}$$

todos los términos en esta expresión son conocidos así que puede realizarse un ajuste en los pesos que enlazan a la capa oculta con la de salida.

Ahora considerar los pesos de los enlaces de la capas I y J. El ajuste de los pesos es

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

de manera similar que los desarrollos anteriores se obtiene

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial o_j} (1 - o_j)\, o_j\, o_i$$

diferente al caso de la capa de salida k, no puede obtenerse una expresión para la derivada parcial sobrante desde la formula de error, dado que o_j no son las salidas en la capa final. En cambio, puede expresarse la derivada en términos de la regla de la cadena involucrando los PEs de salida. Específicamente

$$\begin{aligned}\frac{\partial E}{\partial o_j} &= \sum_k \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial o_j} \\ &= \sum_k \frac{\partial E}{\partial z_k} w_{kj}\end{aligned}$$

la derivada parcial sobrante puede obtenerse de (16) y (18) como

$$\frac{\partial E}{\partial z_k} = -(t_k - o_k)(1-o_k)o_k$$

así que se obtiene

$$\Delta w_{ji} = \eta\,(1-o_j)\,o_j\,o_i \sum_k (t_k - o_k)(1-o_k)\,o_k\,w_{kj} \tag{20}$$

teniendo determinado el valor de w_{kj} en (19), es posible encontrar los valores w_{ji} dado que los términos en (20) son conocidos o pueden calcularse fácilmente

Para conveniencia matemática se define

$$\delta_k = (t_k - o_k)(1-o_k)\,o_k \tag{21}$$

y

$$\begin{aligned}\delta_j &= (1-o_j)\,o_j \sum_k (t_k - o_k)(1-o_k)\,o_k\,w_{kj} \\ &= (1-o_j)o_j \sum_k \delta_k\,w_{kj}\end{aligned} \tag{22}$$

por último se obtiene

$$\Delta w_{kj} = \eta\,\delta_k\,o_j \tag{23}$$

$$\Delta w_{ji} = \eta\,\delta_j\,o_i \tag{24}$$

para observar el requerimiento de una función de activación diferenciable deben compararse las dos últimas ecuaciones con (9).

Los umbrales θ_j y θ_k en (12) y (13) se encuentran de la misma manera que los valores de los pesos en la (23) y (24), pero con las entradas correspondientes seleccionadas iguales a 1.

Teniendo el desarrollo matemático, se describe cómo se realiza el entrenamiento. La red se inicializa con un conjunto arbitrario de pesos de manera tal que pueda funcionar para proporcionar una salida cualquiera. Luego los pixeles de entrenamiento son examinados uno a la vez por la red. Para un pixel dado la salida de la red es calculada por las ecuaciones de la red. Casi con toda certeza la salida será incorrecta, es decir la salida o_k de ese pixel no coincidirá con la salida da deseada u objetivo t_k del pixel, como se indicó cuando se hizo su etiquetado en los datos de entrenamiento. Por tanto, hay que ajustar los pesos de los PE en (23) usando la definición de δ_k en (21). Con estos nuevos valores de δ_k y por lo tanto w_{kj}, (22) y (24) se pueden aplicar para encontrar nuevos valores de pesos en las capas anteriores. De esta manera el efecto del error en las salidas se propaga hacia atrás para corregir los pesos. Esta técnica es conocida como *retropropagación*.

[Pao, 1989] recomienda que los pesos no sean corregidos en cada examen de un pixel de entrenamiento único sino que se haga las correcciones de todos los pixeles del conjunto de entrenamiento en un solo ajuste. Así para p patrones de entrenamiento los ajustes generales están dados por

$$\Delta w'_{kj} = \sum_p \Delta w_{kj} \qquad \text{y} \qquad \Delta w'_{ji} = \sum_p \Delta w_{ji}$$

después de que los pesos han sidos ajustados los pixeles son nuevamente examinados por la red y las salidas se recalculan para ver si corresponden a las clases deseadas. Usualmente habrá errores y el proceso de ajuste de pesos debe repetirse. Este proceso se itera tantas veces como sea necesario para que la red responda la clase correcta que tiene los pixeles de entrenamiento o hasta que el número de errores en la clasificación de los pixeles de entrenamiento sea reducido en un nivel aceptable.

Una variante al método descrito es incluir en el algoritmo un término denominado *momento* (momentum), que consiste en agregar al cálculo de los pesos un término adicional proporcional al incremento de la iteración o época anterior

$$\Delta w_{kj}(t+1) = \eta\,\delta_k^{\mu} + \alpha\,\Delta w_{kj}(t) \tag{25}$$

Siendo α un parámetro entre 0 y 1, que se suele tomar muy próximo al 1 ($\alpha \approx 0.9$). De esta manera, si el incremento en un determinado peso tiene siempre el mismo signo, las actualizaciones, en cada iteración serán mayores; sin embargo si los incrementos en ciertos pesos oscilan (a veces son positivos, otras negativos), el incremento efectivo (acumulado) se reduce al cancelarse. Así, en zonas estrechas y profundas de la hipersuperficie de error (con forma de valle angosto), los pesos correspondientes a la dimensión estrechas (que sin el término de momento oscilarían de un lado al otro del valle) sufren incrementos pequeños, mientras que los de las direcciones que descienden directamente al fondo se ven potenciados [Bishop, 1994 y citado por Martín y Sanz, 2001]. Esta es una manera de aumentar el ritmo de aprendizaje efectivo en determinadas direcciones.

4 Ejemplo de aplicación del algoritmo de retropropagación

Para la implementación de una **ANN** se requiere inicialmente de algunas consideraciones importantes. Primero, debe seleccionarse el número de capas, generalmente una red de tres capas es suficiente, con la finalidad que la primera capa distribuya los componentes de los patrones de entrada a cada PE de la capa oculta. Es decir, las neuronas de la capa de entrada no realizan ningún cómputo o procesamiento, únicamente envían la información a las neuronas de la siguiente capa.

Posteriormente se realiza la selección del número de PE en cada capa. La capa de entrada generalmente tiene tantas neuronas como bandas existan. El número de elementos en la capa de salida depende de las salidas utilizadas para representar las respuestas de la red. Alternativamente, la posibilidad de usar salidas binarias, por ejemplos puede utilizarse dos PE para representar cuatro salidas y tres PE para representar ocho salidas y así sucesivamente.

Como guía general, se recomienda que el número de PE en la capa oculta o capas de procesamiento deben ser iguales o mayor que el número de nodos de la capa de entrada [Lippman, 1987 y citado por Richards y Jia, 1999].

Como ejemplo instructivo, se considera la implementación de una **ANN** para el desarrollo de la solución de un problema de clasificación espectral. En la Figura 7 se muestra dos clases de datos, con tres patrones de entrenamiento en cada uno, organizados de manera tal que no exista separación lineal. La Figura 8 muestra la **ANN** utilizada para llevar a cabo la discriminación de los datos.

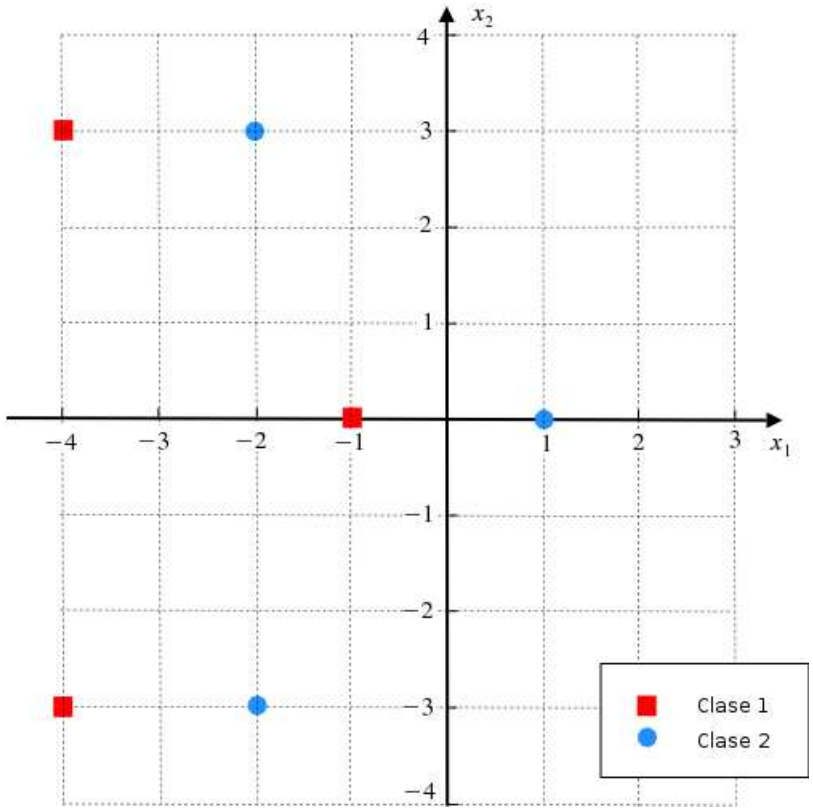


Figura 7. Dos conjuntos de datos que representan dos clases que no pueden separarse linealmente

El conjunto de patrones de entrenamiento tanto de las entradas como salidas deseadas son

	c_{11}	c_{12}	c_{13}	c_{21}	c_{22}	c_{23}
Banda x_1	-4	-4	-1	-2	-2	1
Banda x_2	3	-3	0	-3	3	0
Salida objetivo	0	0	0	1	1	1

Tabla 1. Patrones de entrenamiento para el problema de clasificación de la Figura 7

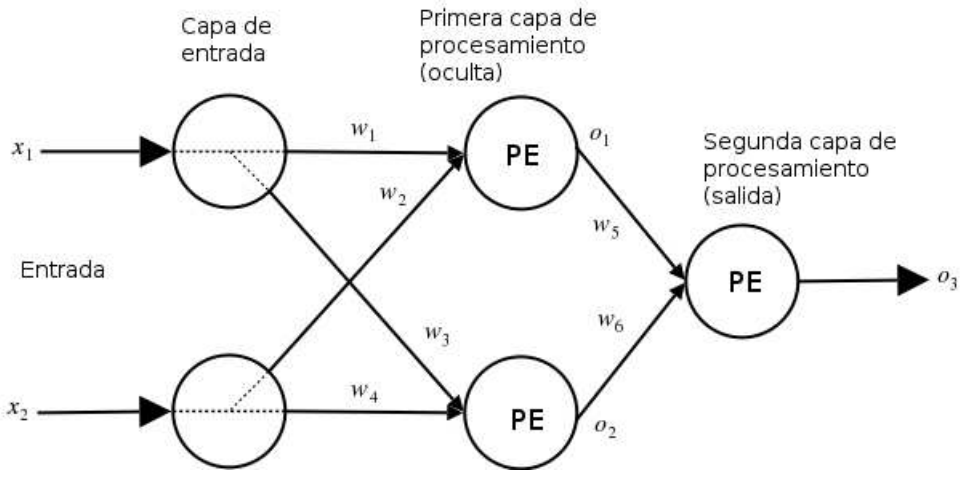


Figura 8. Red neuronal implementada para la solución de los datos de la Figura 7

La formulación de las ecuaciones de la red son las siguientes:

$$\begin{aligned} o_1 &= f(z_1), \quad z_1 = w_1\, x_1 + w_2\, x_2 \\ o_2 &= f(z_2), \quad z_3 = w_3\, x_1 + w_4\, x_2 \\ o_3 &= f(z_3), \quad z_3 = w_5\, o_1 + w_6\ o_2 + \theta \end{aligned}$$

El procedimiento del algoritmo de retropropagación a seguir es:

Paso 0. Inicializar los parámetros de entrenamiento y los pesos con pequeños valores arbitrarios o aleatorios.

$$\eta = 0.2,\ \alpha = 0.9$$

$$w_{J10} = -0.1875, w_{J11} = 0.9320, w_{J12} = -0.4577$$

$$w_{J20} = -0.6113, w_{J21} = -0.3974, w_{J22} = -0.2099$$

$$w_{K10} = 0.6656, w_{K11} = -0.5493, w_{K12} = 0.3705$$

Paso 1. Mientras el criterio o condición de parada no se cumpla, proceder con los pasos 2 al 9. El criterio de parada utilizado en este caso se refiere cuando el error global sea menor a $E = 0.05$. Por lo tanto se inicia calculando el error asociado a cada patrón y el error total. Se tienen los siguientes datos:

Patrones de entrenamiento X_1, X_2. Constituyen las entradas para la capa oculta J con componentes de umbral y los niveles digitales de las bandas x_1 y x_2

$$\begin{aligned} X_1 &= \begin{pmatrix} 1 & 1 & 1 \\ x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -4 & -4 & -1 \\ 3 & -3 & 0 \end{pmatrix} \\ X_2 &= \begin{pmatrix} 1 & 1 & 1 \\ x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -2 & -2 & 1 \\ -3 & 3 & 0 \end{pmatrix} \end{aligned}$$

Las matrices de pesos $W_J(0), W_K(0)$ para la época o iteración $t := 0$. Cada fila se refiere a los pesos tomados aleatoriamente y asociados a cada PE de la capa respectiva

$$\begin{aligned} W_J(0) &= \begin{pmatrix} w_{J10} & w_{J11} & w_{J12} \\ w_{J20} & w_{J21} & w_{J22} \end{pmatrix} = \begin{pmatrix} -0.1875 & 0.9320 & -0.4577 \\ -0.6111 & -0.3974 & -0.2099 \end{pmatrix} \\ W_K(0) &= \begin{pmatrix} w_{K10} & w_{K11} & w_{K21} \end{pmatrix} = \begin{pmatrix} 0.6656 & -0.5493 & 0.3705 \end{pmatrix} \end{aligned}$$

Los vectores de salidas T_1, T_2 correspondientes al patrón de entrenamiento 1 y 2, respectivamente

$$\begin{aligned} T_1 &= \begin{pmatrix} t_{11} & t_{12} & t_{13} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \\ T_2 &= \begin{pmatrix} t_{11} & t_{12} & t_{13} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \end{aligned}$$

Al propagar las entradas por medio de la regla de la suma ponderada se encuentra para el primer patrón de entrenamiento

$$\begin{aligned} Z_{J1} &= W_J(0)\, X_1 \\ Z_{J1} &= \begin{pmatrix} -0.1875 & 0.9320 & -0.4577 \\ -0.6111 & -0.3974 & -0.2099 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ -4 & -4 & -1 \\ 3 & -3 & 0 \end{pmatrix} \\ Z_{J1} &= \begin{pmatrix} -5.2887 & -2.5423 & -1.1195 \\ 0.3487 & 1.6079 & -0.2139 \end{pmatrix} \end{aligned}$$

Aplicar la función de activación sobre los valores de la regla de propagación Z_{J1}

$$\begin{aligned} O_{J1} &= f(Z_{J1}) \\ O_{J1} &= \begin{pmatrix} 1 & 1 & 1 \\ 0.0050 & 0.07294 & 0.2460 \\ 0.5863 & 0.8331 & 0.4467 \end{pmatrix} \end{aligned}$$

donde f es la función sigmoide definida por (11), y los componentes con valor de 1 en la fila 1 se han agregado con el fin de constituir las salidas del elemento de umbral de la capa oculta J es decir, O_{J1} representa las nuevas entradas de la capa de salida K para el patrón de entrenamiento 1

Regla de propagación para la neurona de salida

$$\begin{aligned} Z_{k1} &= W_K(0)\, O_{J1} \\ Z_{K1} &= \begin{pmatrix} 0.6656 & -0.5493 & 0.3705 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0.0050 & 0.07294 & 0.2460 \\ 0.5863 & 0.8331 & 0.4467 \end{pmatrix} \\ Z_{K1} &= \begin{pmatrix} 0.8801 & 0.9342 & 0.6959 \end{pmatrix} \end{aligned}$$

Función de activación o de transferencia en la capa de salida

$$\begin{aligned} O_{K1} &= f(Z_{K1}) \\ O_{K1} &= \begin{pmatrix} 0.7068 & 0.7179 & 0.6673 \end{pmatrix} \end{aligned}$$

Error asociado al patrón de entrenamiento 1

$$\begin{aligned} E_1 &= (T_1 - O_{K1})^2 \\ E_1 &= \begin{pmatrix} 0.4996 & 0.5154 & 0.4451 \end{pmatrix} \\ E_1 &= \sum_{\mu=1}^{p} E_1 \\ E_1 &= 1.4603 \end{aligned}$$

la suma de los errores se realiza sobre todos las muestras μ, $\mu = 1, 2, ..., p$ del patrón de entrenamiento 1

Para el patrón de entrenamiento 2

$$\begin{aligned} Z_{J2} &= W_J(0)\, X_2 \\ Z_{J2} &= \begin{pmatrix} -0.6784 & -3.4246 & 0.7445 \\ 0.8132 & -0.4462 & -1.0087 \end{pmatrix} \\ O_{J2} &= f(Z_{J2}) \\ O_{J2} &= \begin{pmatrix} 1 & 1 & 1 \\ 0.3366 & 0.0315 & 0.6779 \\ 0.6927 & 0.3902 & 0.2672 \end{pmatrix} \\ Z_{K2} &= W_K(0)\, O_{J2} \\ Z_{K2} &= \begin{pmatrix} 0.7373 & 0.7929 & 0.3922 \end{pmatrix} \\ O_{K2} &= f(Z_{K2}) \\ O_{K2} &= \begin{pmatrix} 0.6764 & 0.6884 & 0.5968 \end{pmatrix} \\ E_2 &= (T_2 - O_{K2})^2 \\ E_2 &= \begin{pmatrix} 0.1046 & 0.0970 & 0.1628 \end{pmatrix} \\ E_2 &= \sum_{\mu=1}^{p} E_2^2 \\ E_2 &= 0.3643 \end{aligned}$$

Por último el error global para evaluar el criterio de parada es

$$\begin{aligned} E &= E_1(0) + E_2(0) \\ E &= 1.4603 + 0.3643 \end{aligned}$$

$$E = 1.8246$$

por lo deben ajustarse los pesos con los valores encontrados hasta ahora

Paso 2. Para cada patrón (X_q, T_q) del conjunto de entrenamiento realizar los pasos 3 al 8.

Propagación hacia adelante

Paso 3. Cada unidad de entrada $(i = 1, 2, ..., l)$ recibe un valor de entrada x_q y envía la señal a todas las unidades de la capa siguiente (capa oculta)

Paso 4. Cada unidad oculta $(j = 1, 2, ..., m)$ aplica una regla de propagación generalmente utilizando la suma las entradas ponderadas (calculadas anteriormente)

$$\begin{aligned} Z_{J1} &= \begin{pmatrix} -5.2887 & -2.5423 & -1.1195 \\ 0.3487 & 1.6079 & -0.2139 \end{pmatrix} \\ Z_{J2} &= \begin{pmatrix} -0.6784 & -3.4246 & 0.7445 \\ 0.8132 & -0.4462 & -1.0087 \end{pmatrix} \end{aligned}$$

Aplicar al valor de la regla de propagación la función de trasferencia o activación para calcular el valor de salida

$$\begin{aligned} O_{J1} &= \begin{pmatrix} 0.0050 & 0.07294 & 0.2460 \\ 0.5863 & 0.8331 & 0.4467 \end{pmatrix} \\ O_{J2} &= \begin{pmatrix} 0.3366 & 0.0315 & 0.6779 \\ 0.6927 & 0.3902 & 0.2672 \end{pmatrix} \end{aligned}$$

y enviar la señal a todas las unidades de la capa siguiente (capa de salida).

$$\begin{aligned} O_{J1} &= \begin{pmatrix} 1 & 1 & 1 \\ 0.0050 & 0.07294 & 0.2460 \\ 0.5863 & 0.8331 & 0.4467 \end{pmatrix} \\ O_{J2} &= \begin{pmatrix} 1 & 1 & 1 \\ 0.3366 & 0.0315 & 0.6779 \\ 0.6927 & 0.3902 & 0.2672 \end{pmatrix} \end{aligned}$$

Paso 5. Cada unidad de salida $(k = 1, 2, ..., n)$ aplica la regla de propagación como la suma de los valores de entradas ponderadas

$$\begin{aligned} Z_{K1} &= \begin{pmatrix} 0.8801 & 0.9342 & 0.6959 \end{pmatrix} \\ Z_{K2} &= \begin{pmatrix} 0.7373 & 0.7929 & 0.3922 \end{pmatrix} \end{aligned}$$

y aplicar la función de activación para calcular su señal de salida

$$\begin{aligned} O_{K1} &= \begin{pmatrix} 0.7068 & 0.7179 & 0.6673 \end{pmatrix} \\ O_{K2} &= \begin{pmatrix} 0.6764 & 0.6884 & 0.5968 \end{pmatrix} \end{aligned}$$

Retropropagación de los errores

Paso 6. Cada unidad de salida $(k = 1, 2, ...n)$ recibe un patrón objetivo correspondiente al patrón de entrenamiento de entrada, calcular el término de error asociado (21)

$$\delta_K = \begin{pmatrix} t_{11} - o_{K11} & t_{12} - o_{K12} & t_{13} - o_{K13} \end{pmatrix} \begin{pmatrix} o_{k11}(1 - o_{k11}) & 0 & 0 \\ 0 & o_{k12}(1 - o_{k12}) & 0 \\ 0 & 0 & o_{k13}(1 - o_{k13}) \end{pmatrix}$$

$$\begin{aligned} \delta_{K1} &= \begin{pmatrix} -0.1465 & -0.1454 & -0.1481 \end{pmatrix} \\ \delta_{K2} &= \begin{pmatrix} 0.0708 & 0.0669 & 0.0970 \end{pmatrix} \end{aligned}$$

calcule el término de corrección de los pesos (23) (usado para después actualizar w_{kj})

$$\begin{aligned} \Delta W_{K1} &= \eta\, \delta_{K1}\, O_{J1}^t \\ \Delta W_{K1} &= \begin{pmatrix} -0.0880 & -0.0096 & -0.0546 \end{pmatrix} \\ \Delta W_{K2} &= \begin{pmatrix} 0.0469 & 0.0183 & 0.0202 \end{pmatrix} \end{aligned}$$

Pero el ajuste total corresponde a la suma total de los incrementos de todos los patrones de entrenamiento

$$\Delta W_K = \begin{pmatrix} -0.0411 & 0.0088 & -0.034 \end{pmatrix}$$

y enviar δ_K hacia atrás a los nodos de la capa inmediatamente anterior (capa oculta)

Paso 7. Cada unidad oculta ($j = 1, 2..., m$) suma sus entradas delta (provenientes de las unidades de la capa oculta) y calcular el término de error asociado (22). Debe tenerse en cuenta que el elemento de umbral en la capa oculta no tiene conexión directa con la capa de entrada por lo que no participa en la retropropagación del error en la capa oculta, igualmente el peso que conecta este PE con la capa de salida se actualizó en el paso 6.

$$\begin{aligned}
\delta_J &= (1-o_k)\, o_k \sum_{k=1}^{n} w_{kj}\,\delta_k = D_{J1}\, W_K\, \delta_K \\
\delta_{J1} &= D_{J1} \begin{pmatrix} -0.5493 \\ 0.3705 \end{pmatrix} \begin{pmatrix} -0.1465 & -0.1454 & -0.1481 \end{pmatrix} \\
\delta_{J1} &= \begin{pmatrix} 0.0004 & 0.0054 & 0.0151 \\ -0.013 & -0.007 & -0.013 \end{pmatrix} \\
\delta_{J2} &= D_{J1} \begin{pmatrix} -0.5493 \\ 0.3705 \end{pmatrix} \begin{pmatrix} -0.1465 & -0.1454 & -0.1481 \end{pmatrix} \\
\delta_{J2} &= \begin{pmatrix} -0.0087 & -0.0011 & -0.012 \\ 0.0056 & 0.0059 & 0.0070 \end{pmatrix}
\end{aligned}$$

calcular el término de corrección de los pesos en la capa oculta (24) (usado para actualizar después w_{ji})

$$\begin{aligned}
\Delta W_J &= \eta\, \delta_J\, X_q^t \\
\Delta W_{J1} &= 0.2 \begin{pmatrix} 0.0004 & 0.0054 & 0.0151 \\ -0.013 & -0.007 & -0.013 \end{pmatrix} \begin{pmatrix} 1 & -4 & 3 \\ 1 & -4 & -3 \\ 1 & -1 & 0 \end{pmatrix} \\
\Delta W_{J1} &= \begin{pmatrix} 0.0042 & -0.0077 & -0.0030 \\ -0.0068 & 0.0192 & -0.003 \end{pmatrix} \\
\Delta W_{J2} &= 0.2 \begin{pmatrix} -0.0087 & -0.0011 & -0.012 \\ 0.0056 & 0.0059 & 0.0070 \end{pmatrix} \begin{pmatrix} 1 & -2 & -3 \\ 1 & -2 & 3 \\ 1 & 1 & 0 \end{pmatrix} \\
\Delta W_{J2} &= \begin{pmatrix} -0.004 & 0.0016 & 0.0045 \\ 0.0037 & -0.0032 & 0.001838 \end{pmatrix}
\end{aligned}$$

y el incremento finalmente en los pesos debido a los dos patrones de entrenamiento

$$\begin{aligned}
\Delta W_J &= \Delta W_{J1} + \Delta W_{J2} \\
\Delta W_J &= \begin{pmatrix} 0.000108 & -0.0061 & 0.0015 \\ -0.0031 & 0.0160 & -0.0032 \end{pmatrix}
\end{aligned}$$

Actualización de los pesos y los umbrales

Paso 8. Cada neurona de salida ($k = 1, 2, ..., n$) actualiza sus pesos y umbrales

$$\begin{aligned}
W_K(t+1) &= W_K(t) + \Delta W_K \\
W_K(1) &= \begin{pmatrix} 0.6656 & -0.5493 & 0.3705 \end{pmatrix} + \begin{pmatrix} -0.0411 & 0.0088 & -0.034 \end{pmatrix} \\
W_K(1) &= \begin{pmatrix} 0.6245 & -0.541 & 0.3361 \end{pmatrix}
\end{aligned}$$

Cada neurona oculta ($j = 1, 2, ..., m$) actualiza sus pesos y umbrales

$$\begin{aligned}
W_J(t+1) &= W_J(t) + \Delta W_J \\
W_J(1) &= \begin{pmatrix} -0.1875 & 0.9320 & -0.4577 \\ -0.6111 & -0.3974 & -0.2099 \end{pmatrix} + \begin{pmatrix} 0.000108 & -0.0061 & 0.0015 \\ -0.0031 & 0.0160 & -0.0032 \end{pmatrix} \\
W_J(1) &= \begin{pmatrix} -0.1876 & 0.9259 & -0.4562 \\ -0.6145 & -0.3814 & -0.2131 \end{pmatrix}
\end{aligned}$$

Paso 9. Evaluación de la condición parada

Siguiendo el paso 1 se calcula nuevamente el error global con los nuevos pesos $W(1)$

$E = 1.79278$

No cumple la condición de parada, por lo que hay que regresar nuevamente a 2 - 9

Debe observarse que en la inicialización de la red no participa el término de momento, puesto que es necesario conocer los incrementos anteriores de los pesos. Sin embargo, para saber como procede esta variación, se han obtenido los incrementos en la iteración 2 y nuevamente se ajustan los pesos, pero en este caso teniendo en cuenta el término de momento

El ajuste a realizar en los pesos que enlazan la capa de salida K

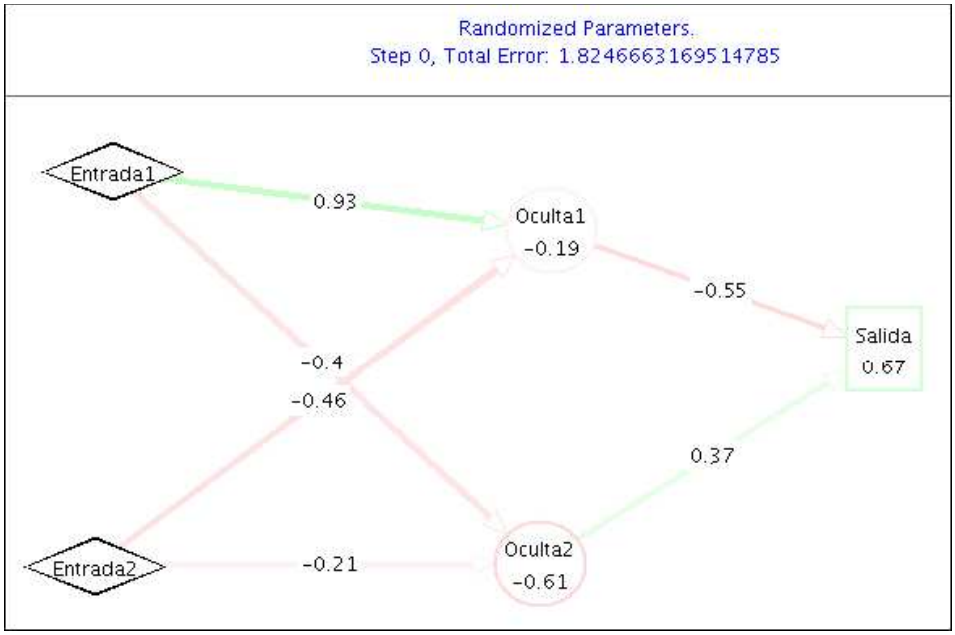
$$
\begin{aligned}
W_K(2) &= W_K(1) + \Delta W_K + \alpha\, \Delta W_K(1) \\
W_K(2) &= \begin{pmatrix} 0.6245 & -0.541 & 0.3361 \end{pmatrix} + \begin{pmatrix} -0.0391 & 0.0095 & -0.0328 \end{pmatrix} + 0.9 \begin{pmatrix} -0.0411 & 0.0088 & -0.034 \end{pmatrix} \\
W_K(2) &= \begin{pmatrix} 0.5485 & -0.5232 & 0.2722 \end{pmatrix}
\end{aligned}
$$

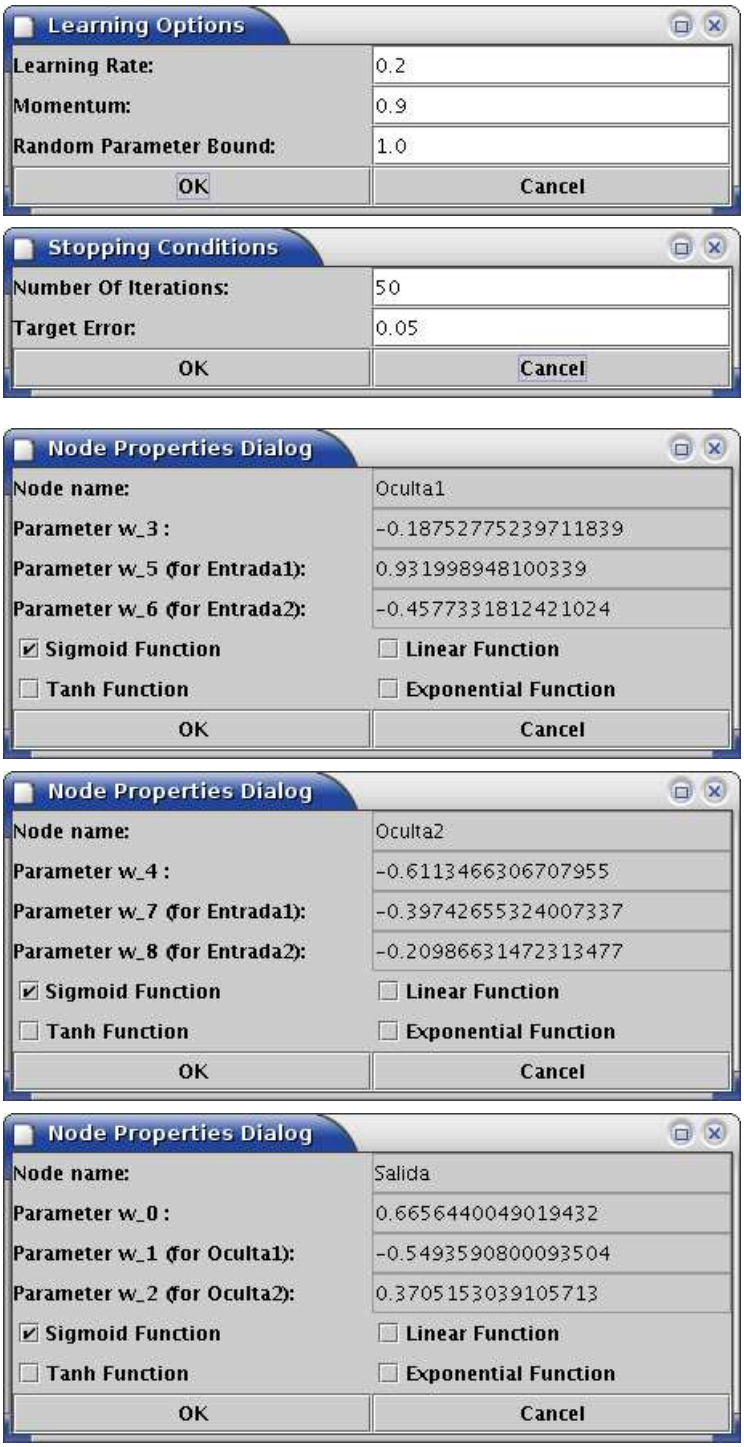
y el ajuste en los pesos de la capa oculta J

$$
\begin{aligned}
W_J(2) &= W_J(1) + \Delta W_J + \alpha\, \Delta W_J(1) \\
W_J(2) &= \begin{pmatrix} -0.1876 & 0.9259 & -0.4562 \\ -0.6145 & -0.3814 & -0.2131 \end{pmatrix} + \begin{pmatrix} -0.0002889 & -0.0059 & 0.0017 \\ -0.0027 & 0.0148 & -0.0029 \end{pmatrix} + \\
& \quad 0.9 \begin{pmatrix} -0.1876 & 0.9259 & -0.4562 \\ -0.6145 & -0.3814 & -0.2131 \end{pmatrix} \\
W_J(2) &= \begin{pmatrix} -0.1880 & 0.9146 & -0.4531 \\ -0.620 & -0.352 & -0.2189 \end{pmatrix}
\end{aligned}
$$

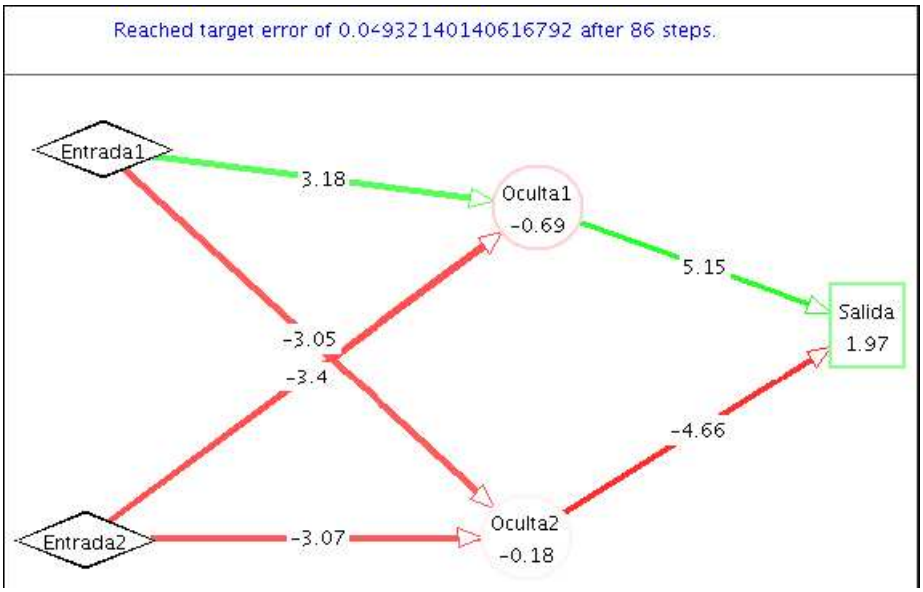
Haciendo uso del simulador de redes neuronales artificiales ***neural*** se obtuvo una solución satisfactoria en la iteración 86. A continuación se muestra algunas de la figuras obtenidas del programa, presentándose los ajustes de cada neurona y el proceso de entrenamiento a través de las iteraciones

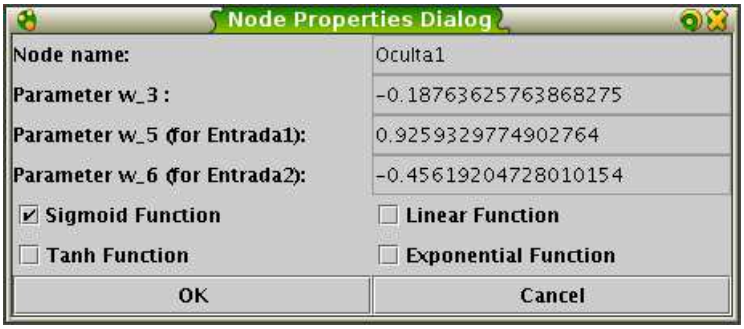
- **Inicialización de los pesos ($t := 0$)**

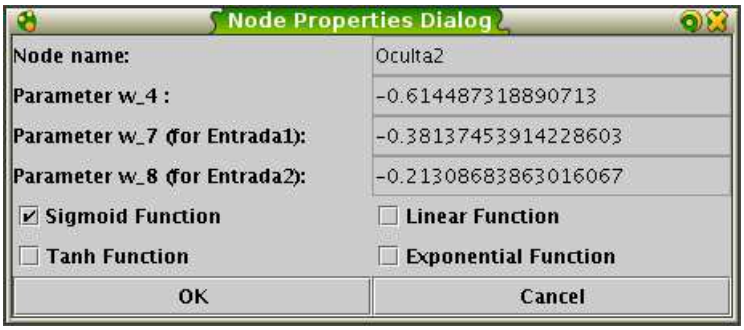


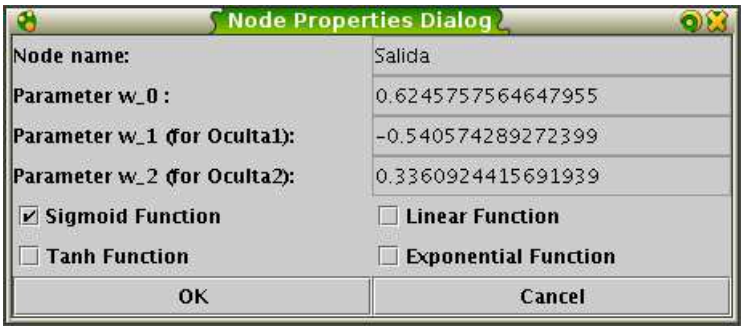

Learning Options
Learning Rate: 0.2
Momentum: 0.9
Random Parameter Bound: 1.0
OK
Cancel
Stopping Conditions
Number Of Iterations: 50
Target Error: 0.05
OK
Cancel
Node Properties Dialog
Node name: Oculta1
Parameter w_3 : -0.18752775239711839
Parameter w_5 (for Entrada1): 0.931998948100339
Parameter w_6 (for Entrada2): -0.4577331812421024
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel
Node Properties Dialog
Node name: Oculta2
Parameter w_4 : -0.6113466306707955
Parameter w_7 (for Entrada1): -0.39742655324007337
Parameter w_8 (for Entrada2): -0.20986631472313477
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel
Node Properties Dialog
Node name: Salida
Parameter w_0 : 0.6656440049019432
Parameter w_1 (for Oculta1): -0.5493590800093504
Parameter w_2 (for Oculta2): 0.3705153039105713
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel

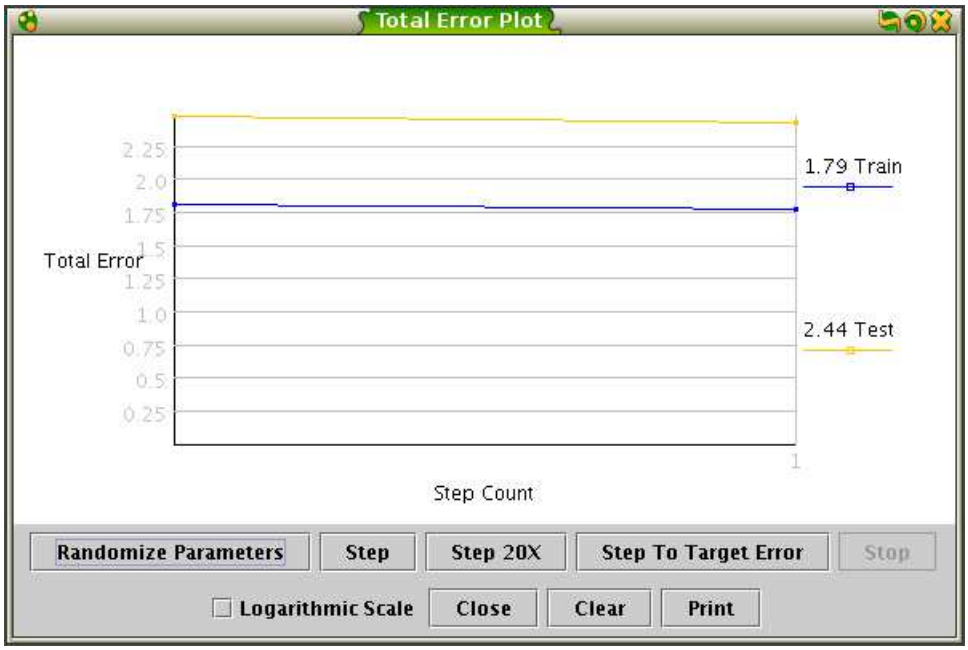
- **Iteración 1**

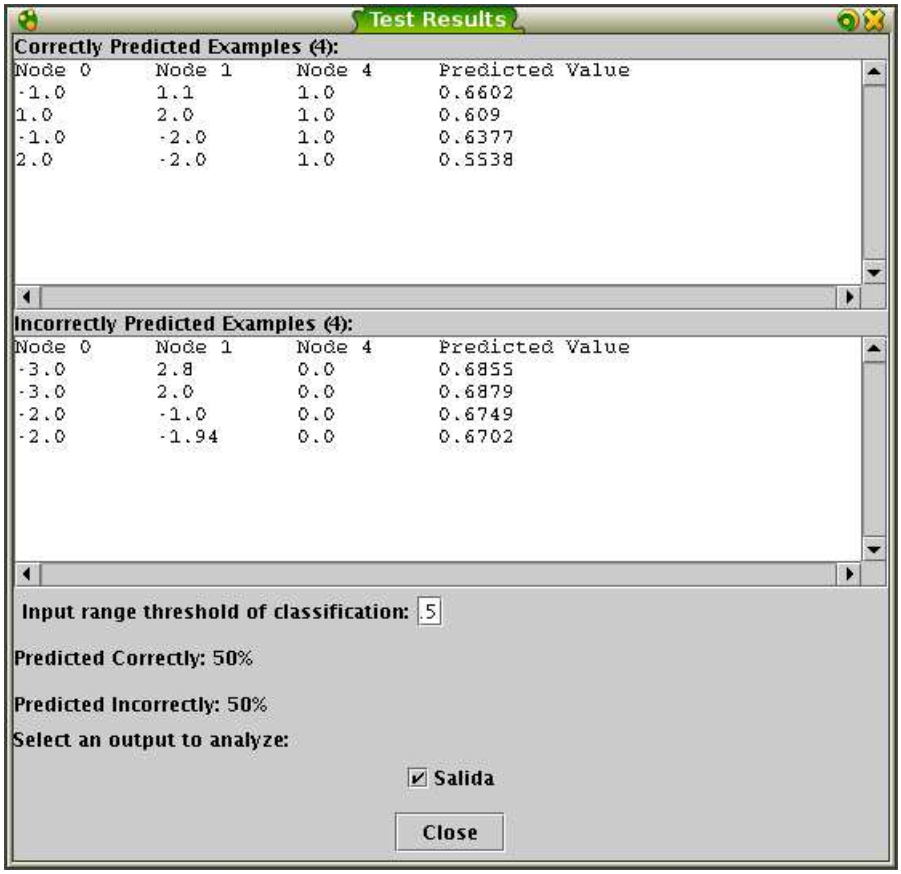

Reached target error of 0.04932140140616792 after 86 steps.
Entrada1
Entrada2
Oculta1
-0.69
Oculta2
-0.18
Salida
1.97
3.18
-3.05
-3.4
-3.07
5.15
-4.66


Node Properties Dialog
Node name: Oculta1
Parameter w_3 : -0.18763625763868275
Parameter w_5 (for Entrada1): 0.9259329774902764
Parameter w_6 (for Entrada2): -0.45619204728010154
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel


Node Properties Dialog
Node name: Oculta2
Parameter w_4 : -0.614487318890713
Parameter w_7 (for Entrada1): -0.38137453914228603
Parameter w_8 (for Entrada2): -0.21308683863016067
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel

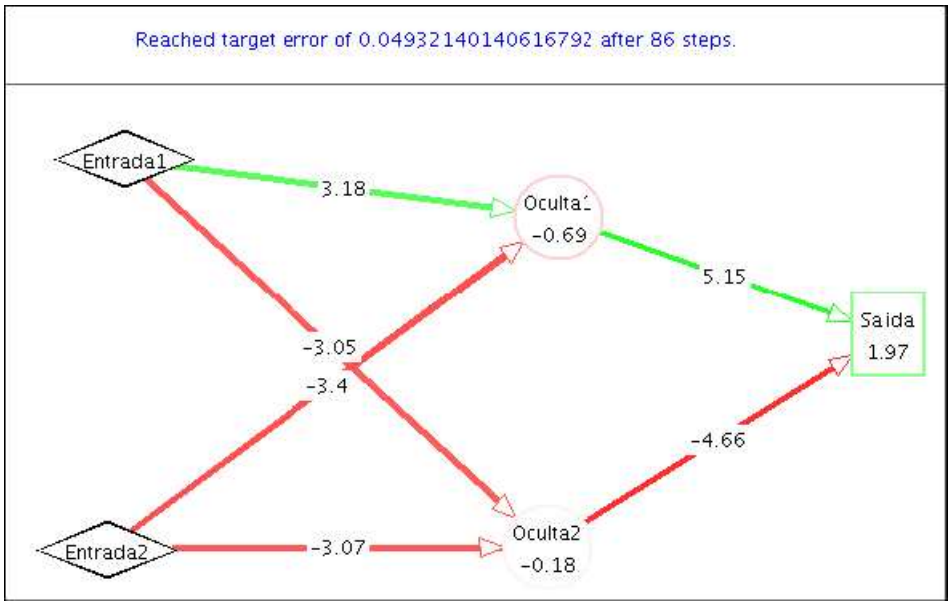

Node Properties Dialog
Node name: Salida
Parameter w_0 : 0.6245757564647955
Parameter w_1 (for Oculta1): -0.540574289272399
Parameter w_2 (for Oculta2): 0.3360924415691939
Sigmoid Function
Linear Function
Tanh Function
Exponential Function
OK
Cancel

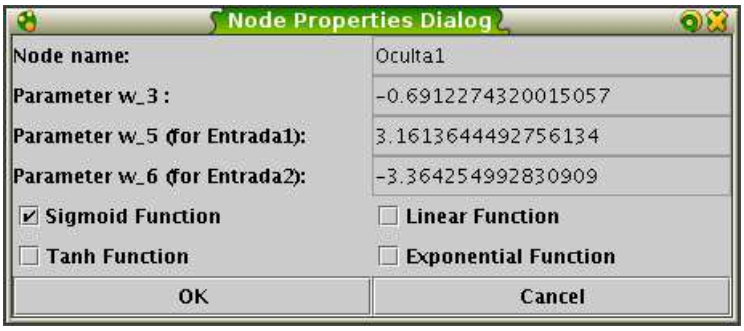


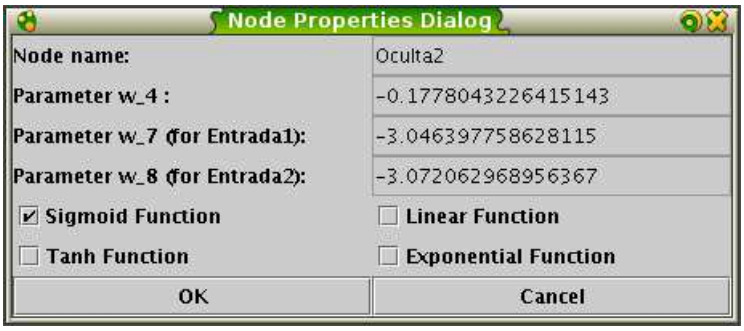


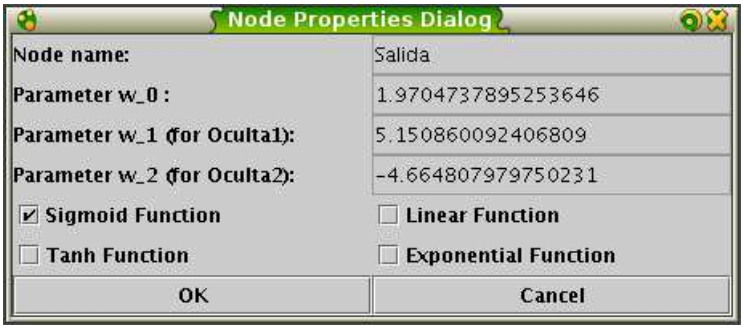
En las dos últimas gráficas se muestra una operación adicional y se refiere al error de generalización, en donde se han incluidos nuevas entradas que no participan en el ajuste de los pesos, solamente son utilizados para saber que tan bien la **ANN** discrimina patrones que no han sido implementados en el entrenamiento de la red. Para esta iteración la **ANN** no es capaz de discriminar 4 de los patrones de prueba adecuadamente

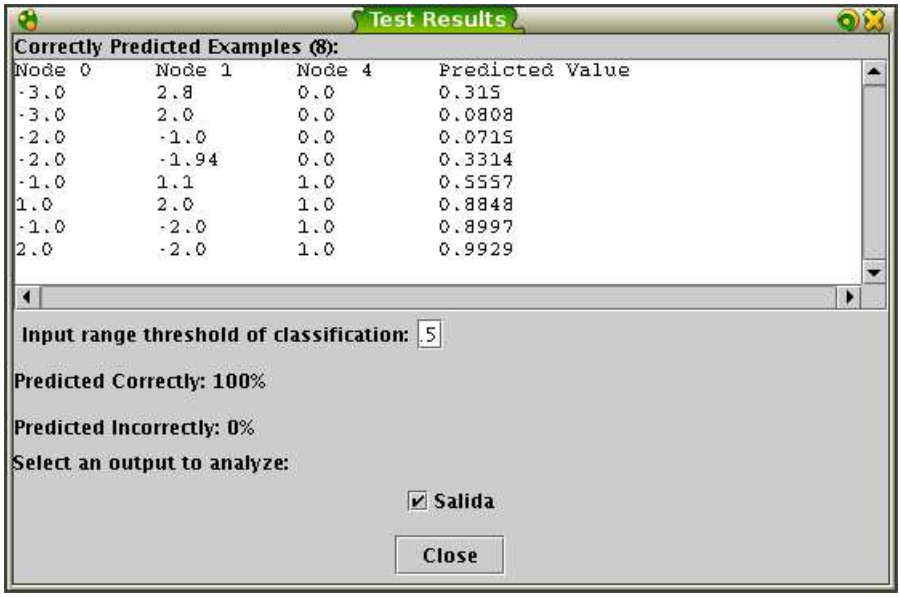
- **Iteración 86**











En esta iteración ya se logra una discriminación adecuada tanto de los patrones de entrenamiento como de prueba, y la red puede ser implementada para clasificar los nuevas entradas.

Bibliografía

[Fyfe, 1997] Fyfe Colin. Artificial Neural Networks. Department of Computing and Information Systems. 1997. The University of Pasley.

[Hristev, 1998] Hristev R. The ANN Book. Disponible en: `ftp://ftp.funet.fi/pub/sci/neural/books/`. 1998. Última visita en Agosto 23 de 2005.

[Martin y Sanz, 2001] Martin Bonifacio y Sanz Alfredo. *Redes Neuronales y Sistemas Borrosos*. RA-MA Editorial, 2da edición. 2001.

[Pao, 1989] Pao H. *Adaptive Pattern Recognition and Neural Networks*. Reading, Addison, Wesley. 1989.

[Richards y Jia, 1999] Richards Jhon y Jia Xiuping. 1999. *Remote Sensing Digital Image Analysis*, páginas 200–220. Springer, 3ra edición.