Revista Facultad de Ingeniería N.º 38. pp. 146-162. Septiembre, 2006

Generación dinámica de la topología de una red neuronal artificial del tipo perceptron multicapa

*Héctor Tabares[a, *], John Branch[b], Jaime Valencia[a]*

[a]Departamento de Ingeniería Eléctrica. Facultad de Ingeniería. Universidad de Antioquia. Apartado Aéreo 1226 Medellín, Colombia.

[b]Escuela de Sistemas. Facultad de Minas. Universidad Nacional de Colombia. Medellín, Colombia

(Recibido el 24 de agosto de 2004. Aceptado el 26 de enero de 2006)

Resumen

En este trabajo se aplica un método constructivo aproximado para encontrar arquitecturas de redes neuronales artificiales (RNA) de tipo perceptrón multicapa (PMC). El método se complementa con la técnica de la *búsqueda forzada de mejores mínimos locales*. El entrenamiento de la red se lleva a cabo a través del algoritmo gradiente descendente básico (GDB); se aplican técnicas como la repetición del entrenamiento y la detención temprana (validación cruzada), para mejorar los resultados. El criterio de evaluación se basa en las habilidades de aprendizaje y de generalización de las arquitecturas generadas específicas de un dominio. Se presentan resultados experimentales con los cuales se demuestra la efectividad del método propuesto y comparan con las arquitecturas halladas por otros métodos.

---------- *Palabras clave:* redes neuronales artificiales, perceptron multicapa, topología, arquitectura.

Dynamic topology generation of an artificial neural network of the multilayer perceptron type

Abstract

This paper deals with an approximate constructive method to find architectures of artificial neuronal network (ANN) of the type Multi-Layer Percetron (MLP) which solves a particular problem. This method is supplemented with the technique of the *Forced search of better local minima*. The training of the net uses an

* Autor de correspondencia. Teléfono: + 574 + 250 57 57, fax +574 263 82 82, correo electrónico: htabares@udea.edu.co (Héctor Tabares).

algorithm basic descending gradient (BDG). Techniques such as repetition of the training and the early stopping (cross validation) are used to improve the results. The evaluation approach is based not only on the learning abilities but also on the generalization of the specific generated architectures of a domain. Experimental results are presented in order to prove the effectiveness of the proposed method. These are compared with architectures found by other methods.

---------- *Key words:* Artificial Neural Networks, Multi-layer Perceptron, Topology, Architecture.

Introducción

Con el algoritmo gradiente descendente básico (GDB) [1] la estimación teórica del número exacto de neuronas ocultas en una RNA del tipo PMC para resolver un problema de aproximación en particular es difícil, pero ello no significa que no se pueda emplear alguna técnica de optimización numérica [2]. Yau [3] presenta un resumen de las metodologías utilizadas con mayor frecuencia. Así por ejemplo, Ergeziner [4], Coetzee [5], Reyneri [6], Park [7] y Sperduti [8] elaboraron algoritmos para determinar el valor de los pesos utilizando técnicas de optimización no lineal. Otros autores como: Ash [9], Hirose [10], Aylward [11], Tauel [12], han incorporado estructuras de redes neuronales variables durante el proceso de aprendizaje.

Ash, por ejemplo, desarrolló un algoritmo con un criterio dinámico para generar la topología. En su propuesta, un nuevo nodo es generado en una capa oculta cuando el error está por debajo de un valor estimado. *Hirose* adoptó el método de *Ash* para la creación de un nodo, y lo completó con una técnica para borrar un nodo cuando su valor es cero o muy cercano a cero. *Aylward y Anderson* [11] propusieron una serie de reglas basadas en el error cuadrático medio (ECM) que alcanza la red durante el proceso de aprendizaje. En este caso, se adiciona un nuevo nodo cuando las reglas no son satisfechas de manera continua. *Masters* [13] propuso la regla de la pirámide geométrica para la creación de la topología. Aquí, la cantidad de neuronas en la capa oculta se calcula como $\sqrt{n*m}$, donde n es el número de entradas y m el número de salidas que tiene la red. *Yao* [14] y *Fiesler* [15] investigaron la aplicación de algoritmos evolutivos para optimizar el número de unidades ocultas y el valor de los pesos en un PMC. En este caso, la programación evolutiva es una técnica estocástica que puede lograr una optimización global. En este último sentido, *Murata* [16] propone métodos estadísticos.

No obstante, la mayor limitante de cualquiera de las anteriores metodologías de tipo constructivo se encuentra en el algoritmo de entrenamiento GDB, el cual puede quedar atrapado en un mal punto mínimo local de la superficie de error, medido en función del ECM, finalizando así el aprendizaje. Por lo anterior, en este artículo se propone complementar el método de la generación dinámica de la topología con la técnica de la *Búsqueda forzada de mejores mínimos locales*. Estos puntos son llamados por *Martín del Brío* [17], *mínimos más profundos*. Con este sistema híbrido, un PMC puede obtener topologías de redes con las cuales resolver problemas de aproximación particulares.

2. Metodología propuesta

Se han presentado diferentes propuestas para determinar topologías de RNA de tipo PMC con las cuales resolver un problema de aproximación específico. De todas ellas, se considera que los métodos propuestos por *Ash* e *Hirose* son los mejores debido a que requieren el menor costo computacional.

En este trabajo se propone una nueva metodología que combina dos técnicas y que tiene como fundamento los métodos elaborados por los anteriores autores: la primera, consiste en generar la topología de una manera dinámica realizando expansiones de tipo serial, paralelo o mixto [9], o podando los nodos que no estén participando activamente en el proceso de aprendizaje [10], mientras se ejecuta el algoritmo de aprendizaje GDB. La segunda, complemento de la anterior, se basa en hacer una *búsqueda forzada de mejores mínimos locales* de la superficie de error, durante la ejecución del algoritmo de aprendizaje GDB.

Técnica de generación dinámica de la topología

Expansión dinámica de la topología [9]

Esta técnica consiste en generar la topología del PMC de manera dinámica, mientras se ejecuta el algoritmo de aprendizaje GDB, utilizando para ello un algoritmo de tipo constructivo. Al aplicar el método, experimentalmente se ha observado que cuando se genera una expansión se produce

una respuesta de la totalidad de la red, que se puede llamar transitoria. Período durante el cual, la red se acomoda a la nueva topología y se distingue por el aumento y rapidez en su capacidad de aprendizaje. Después del período transitorio, sigue el estacionario, caracterizado por la suavidad y la lentitud en el aprendizaje.

La expansión dinámica comienza con una topología de un PMC cualquiera con *n* pesos, *l* neuronas de entrada, *m* neuronas de salida, (figura 1).

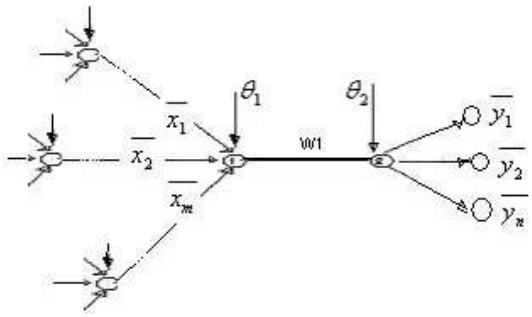


Figura 1 PMC con *m* neuronas de entrada y *n* neuronas de salida

Donde:

$\overline{x_i}$: vector de datos de entrada de la neurona *i.*

$\overline{y_i}$: vector de datos de salida de la neurona *i.*

θ_i: pesos de neuronas adaptativas.

w_1: peso entre neuronas 1 y 2.

Se ejecuta el algoritmo de entrenamiento GDB utilizando las expresiones que se ilustran en la tabla 1, hasta que el PMC ha alcanzado el error mínimo local, es decir

$$\left(\frac{dw_j}{d_{iteraciones}} = 0\right).$$

Tabla 1 Expresiones que operan en la figura 1

Neurona	*Entrada*	*Salida*
1	$-\theta_1 + \sum \overline{x_i}$	$-\theta_1 + \sum \overline{x_i}$
2	$-\theta_2 + w_1\left(-\theta_1 + \sum \overline{x_i}\right)$	$\frac{1}{1+e^{-\left(-\theta_2 + w_1\left(-\theta_1 + \sum \overline{x_i}\right)\right)}}$

(1)

Como consecuencia inmediata, el ECM de la red se convierte en constante. (figura 2).

Con el propósito de aumentar la capacidad de aprendizaje del PMC, se expande la red colocando una nueva neurona en serie (figura 3).

En el instante de la expansión se debe buscar el valor del nuevo peso de la neurona insertada, de manera tal que entre a ser parte de la red de forma controlada (continuidad de la función de error) y constructiva (aumento en la capacidad

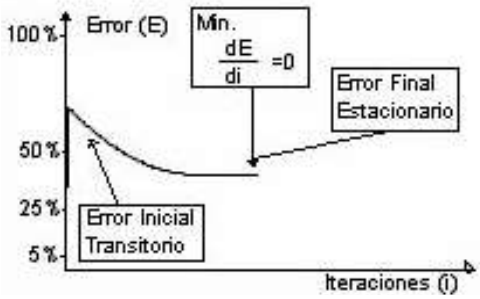


Figura 2 Error mínimo local

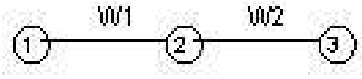


Figura 3 PMC con expansión serial

de aprendizaje de la red), es decir, que este valor no distorsione la información almacenada previamente en la red. Para lograrlo, se supondrá que el valor del nuevo peso debe ser tal, que el error final de la red antes de la expansión sea el mismo error inicial después de la expansión. Esta operación, garantizará la continuidad en la función de error (figura 4).

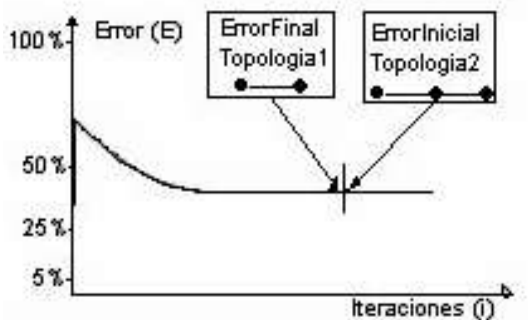


Figura 4 Respuesta deseada del error cuando se hace una expansión serial

Entonces, en el instante de la expansión serial, las nuevas expresiones que entran a ser parte de la red son los que se muestran en tabla 2.

Igualando (1) con (2) y despejando w_2 se encuentra que:

Tabla 2 Expresiones que operan en la figura 3

Neurona	***3***
Entrada	$-\theta_3+\left(\frac{w_2}{1+e^{-\left(-\theta_2+w_1\left(-\theta_1+\sum\overline{x_i}\right)\right)}}\right)$
Salida	$\frac{1}{1+e^{-\left(-\theta_3+\frac{w_2}{1+e^{-\left(-\theta_2+w_1\left(-\theta_1+\sum\overline{x_i}\right)\right)}}\right)}}$ (2)
Igualando (1) con (2) y despejando se encuentra que:	

$$w_2=\left[\theta_3-\theta_2+w_1\left(-\theta_1+\sum\overline{x_i}\right)\right]\left(1+e^{-\left(-\theta_2+w_1\left(-\theta_1+\sum\overline{x_i}\right)\right)}\right) \quad (3)$$

Es decir, el nuevo peso w_2 no es un escalar, como se hubiera querido, sino un vector que depende del peso w_1, de las neuronas adaptativas con pesos θ_i y de los vectores de datos de entradas $\overline{x_i}$ Por lo tanto, no existe ningún valor w_2 tal que garantice la continuidad de la función de error. De hecho, cualquier valor de w_2 que se elija aumentará el error inicial de la nueva topología, cuya consecuencia será también la disminución en la capacidad de aprendizaje de la red. (figura 5).

En términos generales, se puede concluir que para cualquier PMC, cuando se hace una expansión serial, el nuevo peso es función de todos los pesos de la red y de los vectores de entradas, es decir,

$$w_{nuevo}=f\left(w_{ij},\theta,\overline{x_i}\right) \quad (4)$$

Como w_{nuevo} puede ser cualquier valor, una solución práctica es asignarle un valor igual al centroide del conjunto de datos.

$$Expansion\_Serial \Rightarrow w_{nuevo_{inicial}}=\overline{\overline{f\left(w_{ij},\theta,\overline{x_i}\right)}} \quad (5)$$

Se continúa ejecutando el algoritmo de aprendizaje GDB hasta que el PMC ha alcanzado el nuevo error mínimo local o error estacionario ($\frac{dw_j}{d_{iteraciones}}=0$) (figura 6).
Entonces, se realiza una nueva expansión, paralela en este caso (figura 7).

Se debe encontrar el valor de los nuevos pesos w_{n+1} y w_{n+2} de manera que el error final o estacionario antes de la expansión paralela, sea el mismo error inicial o transitorio después de la expansión. En el instante de la expansión paralela, las nuevas expresiones que entran a ser parte de la red se listan en la tabla 3.

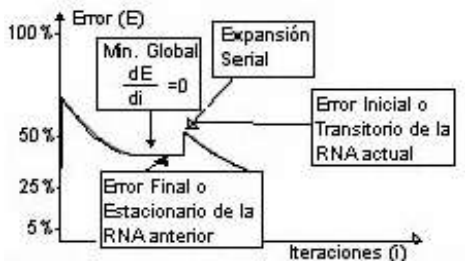


Figura 5 Respuesta encontrada del error cuando se hace una expansión serial

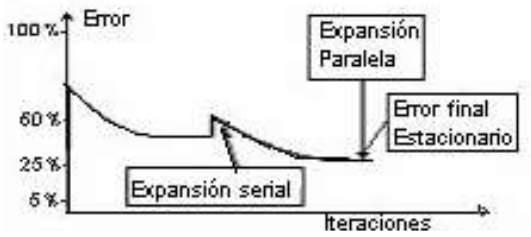


Figura 6 Error estacionario después de realizar la expansión serial

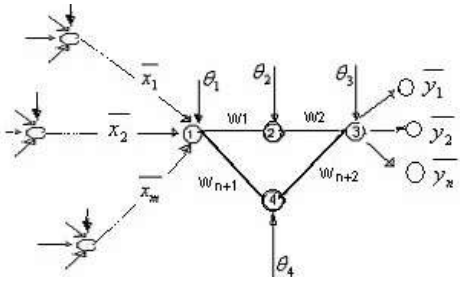


Figura 7 PMC con expansión paralela

Tabla 3 Expresiones de la figura 7

Neurona 4

Entrada

$$-\theta_4 + w_{n+1}\left(-\theta_1 + \Sigma \overline{x_i}\right)$$

Salida

$$\frac{1}{1+e^{-\left(-\theta_4 + w_{n+1}\left(-\theta_1 + \Sigma \overline{x_i}\right)\right)}}$$

Neurona 3

Entrada

$$-\theta_3 + \frac{w_2}{1+e^{-\left(-\theta_2 + w_1\left(-\theta_1 + \Sigma \overline{x_i}\right)\right)}} + \frac{w_{n+2}}{1+e^{-\left(-\theta_4 + w_{n+1}\left(-\theta_1 + \Sigma \overline{x_i}\right)\right)}}$$

Salida

$$\frac{1}{1+e^{-\left(-\theta_3 + \frac{w_2}{1+e^{-\left(-\theta_2 + w_1\left(-\theta_1 + \Sigma \overline{x_i}\right)\right)}} + \frac{w_{n+2}}{1+e^{-\left(-\theta_4 + w_{n+1}\left(-\theta_1 + \Sigma \overline{x_i}\right)\right)}}\right)}} \quad (6)$$

Haciendo (2) = (6) y despejando w_3 y w_4 se obtiene:

$$\frac{w_{n+2}}{1+e^{-\left(-\theta_4 + w_{n+1}\left(-\theta_1 + \sum \overline{x_i}\right)\right)}} = 0 \qquad (7)$$

Para que el cociente sea igual a cero, los nuevos pesos: w_{n+2} deben ser iguales a cero (0) y w_{n+1} puede ser cualquier valor. De esta manera se garantiza que, en el instante de la expansión paralela, no se romperá la continuidad de la función de error.

En este caso, los efectos sobre la función de error de una expansión paralela son completamente contrarios a los efectos que produce una expansión serial (figura 8).

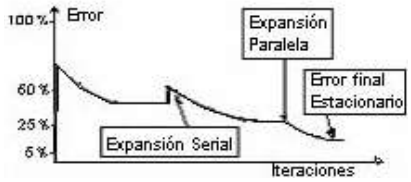


Figura 8 Error estacionario después de la expansión paralela

Se puede concluir que para cualquier PMC con *n* pesos, cuando se hace una expansión paralela, los nuevos pesos de la red son:

$$w_{n+2} = 0 \qquad (8)$$

$$w_{n+1} = Cualquier\ valor \qquad (9)$$

Este resultado confirma el teorema de *Kolmogorov* [18], que dice que un PMC de una sola capa oculta, en donde los nodos están conectados de forma paralela, puede aproximar hasta el nivel deseado cualquier función continua en un intervalo, por lo tanto, las redes neuronales multicapa unidireccionales son aproximadores universales de funciones. No obstante, este teorema no informa sobre el número de nodos ocultos, necesarios para aproximar una función determinada.

En la práctica se ha encontrado que aun cuando las expansiones paralelas entran a ser parte de la red de manera constructiva y controlada, la ganancia en la disminución del error, en algunos casos, es tan baja, del orden de las milésimas y aun menores, que se prefiere analizar otro tipo de expansión, por ejemplo mixta, con la cual aumentar la capacidad de aprendizaje de la red. Para obtener esto primero se expande de forma serial el PMC utilizando la ecuación (5) y luego se expande de forma paralela utilizando las ecuaciones (8) y (9).

Dependiendo del tipo de expansión elegida se pueden obtener diferentes tipos de topologías que resuelven un problema determinado. La finalización del algoritmo se determina usando el criterio de la validación cruzada. Este procedimiento consiste en entrenar y validar a la vez el PMC para detenerse en el punto óptimo *(early stopping)* [13, 19].

Contracción dinámica de la topología

Hirose desarrolló un método automático para podar las neuronas innecesarias de una RNA. Explica el autor que es posible eliminar unidades ocultas que resulten superfluas examinando periódicamente los valores de los pesos de las neuronas. A medida que se entrena la red, se observa que los pesos de ciertas neuronas cambian muy poco con respecto a sus valores finales. Por lo tanto, estos nodos pueden no estar participando en el proceso de aprendizaje y se podan.

Criterios para determinar el tipo de expansión

Para determinar la cantidad máxima de expansiones paralelas, se puede utilizar el criterio planteado por *Hush* [20], quien recomienda como máximo cuatro (4) neuronas ocultas en cada capa.

Una expansión de tipo serial no es recomendable pues está comprobado que en la mayoría de los casos el PMC no aumenta su capacidad de aprendizaje. Por lo tanto, se aconseja complementar este tipo de expansión con una paralela; ambas expansiones conforman una conexión mixta.

Técnica de búsqueda forzada de mejores mínimos locales

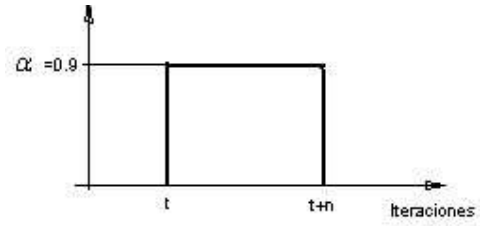
Generar la topología de un PMC de manera dinámica es una propuesta formulada por *Ash* desde el año 1989. No obstante, ésta no se encuentra implementada como una función de la librería ToolBox Neural Network de MATLAB. En Internet se pueden conseguir muchos simuladores de redes neuronales artificiales (NeurDS, GENESIS, Mactivation, SNNS 4.1, etc.) con diversas técnicas de entrenamiento implementadas (Traingd, Traingda, Traingdm, Traingdx, Traingc, Tarinlm, etc.). En contraste, ninguno de ellos tiene implementadas las diferentes metodologías formuladas para generar las topologías de las redes neuronales artificiales con las cuales resolver un problema de aproximación particular. Esto se explica porque las metodologías propuestas por *Ash* e *Hirose*, que son de tipo constructivo, generan topologías que resuelven sólo algunos problemas. La dificultad radica en que al aumentar la topología, la superficie de error que se forma puede ser bastante compleja, así como el proceso de minimización del error.

El algoritmo GDB converge, dependiendo del valor de los pesos, al punto de mínima más próximo. Aun cuando las expansiones paralelas mantienen la continuidad de la función de error haciendo que los nuevos pesos de la red entren a ser parte de la misma de manera controlada, no necesariamente aumentan de forma significativa la capacidad de aprendizaje, es decir, no siempre estos pesos entran de manera constructiva. Menos aún cuando se realizan expansiones de tipo serial, los cuales siempre entran a ser parte de la red de forma descontrolada (no hay continuidad en la función de error) y destructiva (disminuyen la capacidad de aprendizaje del PMC).

Ahora bien, durante la ejecución del algoritmo constructivo, cuando la red se asienta en un mal mínimo local, medido en función del ECM obtenido, en vez de comenzar de nuevo el proceso de aprendizaje de la red con otros pesos inicializados aleatoriamente, se propone aumentar el valor del coeficiente de aprendizaje transitoriamente durante la ejecución del algoritmo de entrenamiento. Su efecto es hacer que el algoritmo salga de la silleta de la superficie de error en la que se encuentra atrapado. Esto significa encontrar el valor óptimo del coeficiente de aprendizaje α con el cual realizar esta operación. Infortunadamente no hay ningún criterio matemático que permita hacerlo, aunque algunos autores sugieren de forma empírica métodos para la determinación inicial de este.

A través de la experimentación se ha encontrado que al aumentar manualmente el coeficiente de aprendizaje α a su máximo valor (p. e. 0,9) durante un número determinado de iteraciones (denominado período transitorio), el algoritmo sale de la silleta de error en la que estaba atrapado para acentuarse en el mínimo local más próximo, mejor que el anterior, medido en función del ECM final obtenido. Por lo tanto, esta operación es completamente forzada. Si se quisiera representar matemáticamente, podría aproximarse a la función pulso, como se ilustra a continuación.

Figura 9 Función pulso



El coeficiente de aprendizaje α se mantiene en su máximo valor, hasta que aplicando el método de la observación estructurada, se observa que la red, durante la ejecución del algoritmo de aprendizaje, se asienta en el mínimo local más próximo. En ese momento se vuelve a disminuir el coeficiente de aprendizaje α a su valor inicial (p. e. 0,01) con el objeto de que la red converja suavemente a la nueva solución.

El anterior procedimiento es completamente manual y empírico. Lo que también significa que el número de iteraciones transitorias utilizadas para realizar la búsqueda de los otros mejores puntos, candidatos a ser mínimos locales sobre la superfi-

cie de error, dependerá de la habilidad que tenga el usuario para manejar el sistema informático en el que se programe esta solución.

Implementación del *software* del método propuesto

Con el objeto de validar la metodología propuesta y realizar los experimentos numéricos, las pruebas de ensayo y las mediciones hechas en este artículo, se desarrolló el programa de computadora RNA_UdeA.exe, con el cual se pueden generar topologías de PMC dinámicamente, es decir durante la ejecución del algoritmo de entrenamiento. Es sabido que esta operación no se puede realizar con los programas convencionales utilizados para entrenar PMC, por ejemplo MATLAB. En este caso, la generación de la topología se determina de forma estática en tiempo de edición o programación. Luego, esta característica lo convierte en un desarrollo en tecnología informática único en su género, y con el cual se pueden mapear funciones con arquitecturas de PMC de una entrada, una salida y máximo cinco (5) capas ocultas con cinco (5) neuronas en cada una de ellas.

La implementación *software* del método propuesto se encuentra en la siguiente dirección electrónica:

http://ingenieria.udea.edu.co/inicial.html

Siga la siguiente ruta de acceso:

- Menú principal: producciones.
- Menú emergente: páginas web académicas.
- Opción: temas de apoyo a cursos.

Héctor Tabares O.

- Redes neuronales artificiales (RNA_UdeA.exe).

A continuación se ilustra la forma como se maneja el programa. Suponiendo que se quiere entrenar un PMC para que se aproxime, por ejemplo, a la función seno, siga las siguientes instrucciones:

- **Primero.** Entre a la forma Topología presionando el botón de la interfaz del sistema. Llene la forma como se ilustra en la figura 10.

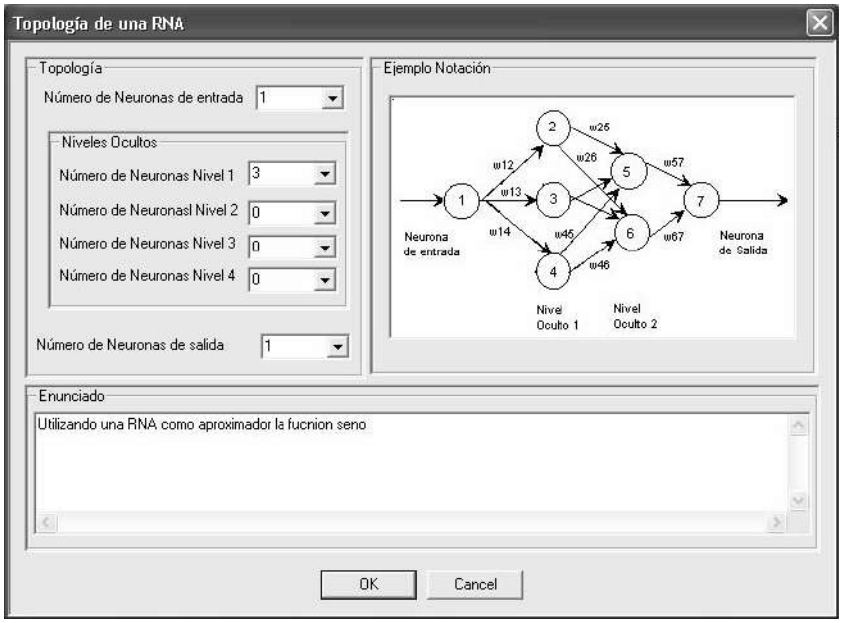


Figura 10 Interfaz del sistema topología

- **Segundo.** Entre a la forma Inicialización Pesos presionando el botón de la interfaz del sistema. Presione el botón Inicialización pesos ALEATORIAMENTE.

Figura 11 Interfaz lectura pesos del PMC

- **Tercero.** Entre a la forma Lectura Tuplas presionando el botón de la interfaz del sistema. Llene la forma como se ilustra a continuación.

Figura 12 Interfaz lectura Tuplas de Entrenamiento

- **Cuarto.** Entre a la forma Algoritmo de Aprendizaje presionando el botón de la interfaz del sistema. Presione el botón

para ejecutar el algoritmo de aprendizaje GDB. Cuando desee detener su ejecución presione el botón

para finalizar.

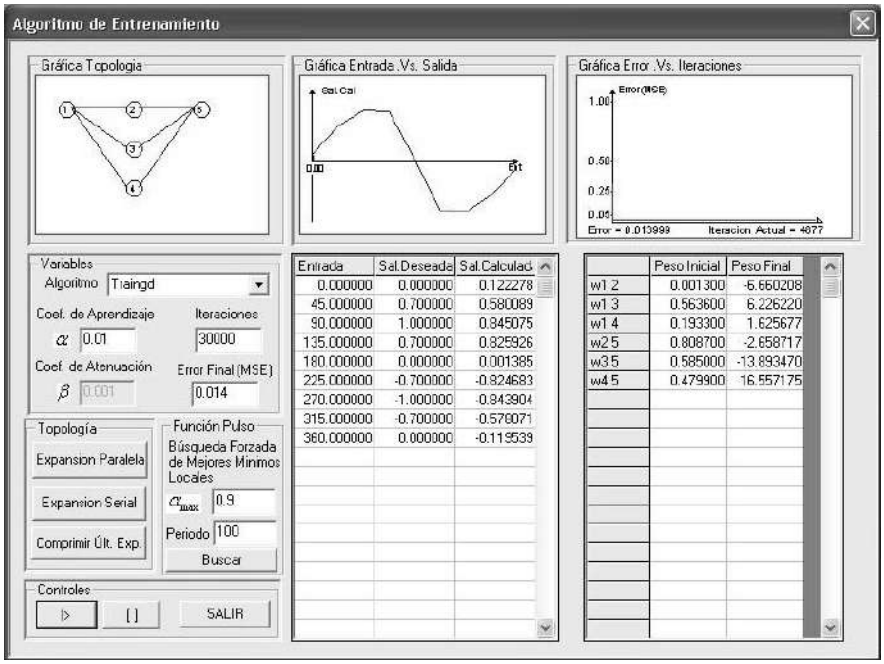


Figura 13 Interfaz Algoritmo de Aprendizaje

- **Quinto.** Entre a la forma Validación del Entrenamiento presionando el botón de la interfaz del sistema. Presione el botón "Calcular" del grupo Validar Grupo Valores. El sistema muestra la forma que se ilustra en la figura 15.

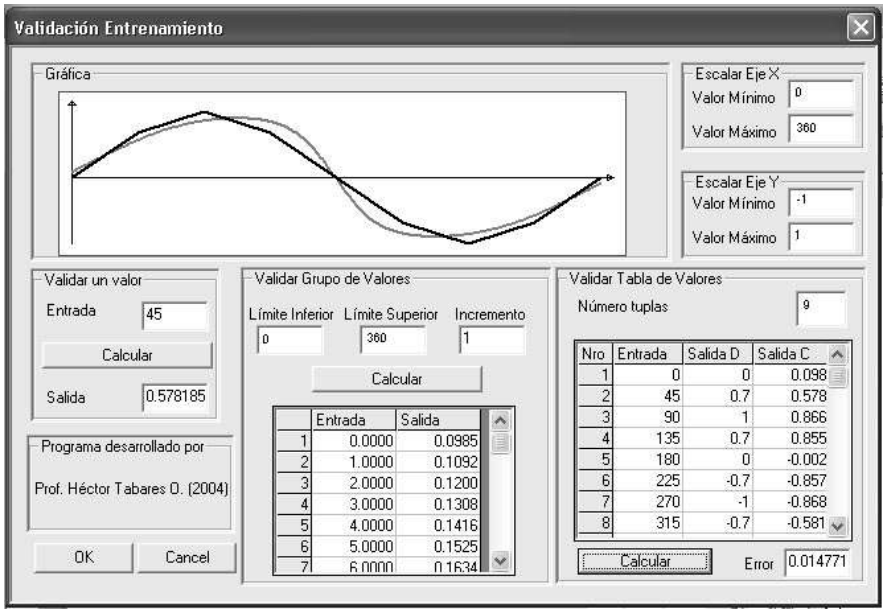


Figura 14 Interfaz Validación del PMC

- **Sexto.** El programa RNA_UdeA también posee un módulo para hacer prácticas sobre el reconocimiento de imágenes. Este módulo es capaz de reconocer imágenes en blanco y negro representadas por matrices de 5 x 5 = 25 pixeles. Esta red estará compuesta por 25 neuronas de entrada y 25 neuronas de salida. La cantidad de neuronas en la capa oculta puede ser determinada por el usuario del sistema. Se trabajará con valores binarios (píxel negro = 1, píxel blanco = 0).

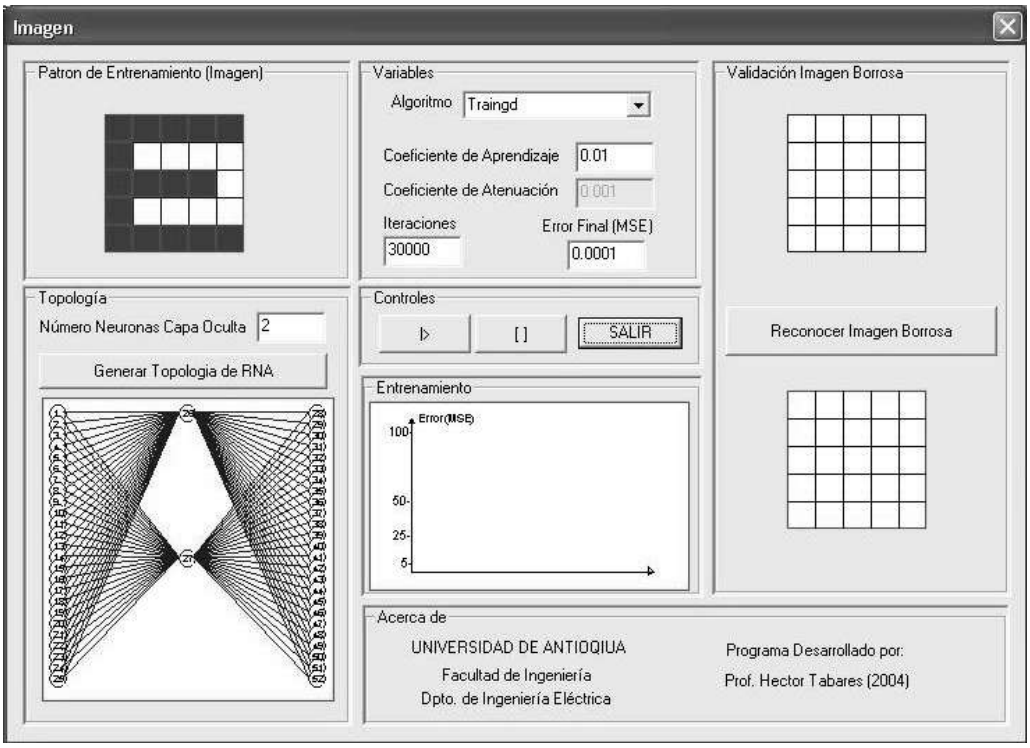


Figura 15 Interfaz Reconocimiento de Imágenes

Validación del método propuesto

El criterio de evaluación se basa en la habilidad de aprendizaje de las arquitecturas específicas de un dominio con la complejidad de la topología generada. Se utilizarán las funciones de iteración simple, radial e iteraciones complicadas [21] para validar la metodología propuesta. Éstas son consideradas por los especialistas en el tema, como problemas referencia para algoritmos de entrenamiento y selección de la topología. Para hacer más sencillas las comparaciones, se confinarán las pruebas de validación en dos dimensiones. De esta manera, los modelos pueden ser visualizados gráficamente como una función $y = g(x1, x2)$.

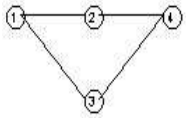
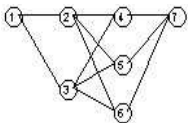
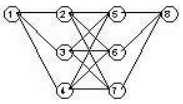
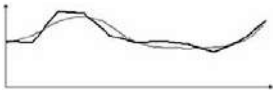
Las neuronas adaptativas son opcionales [17], por lo cual las simulaciones se harán sin estas conexiones. El algoritmo de entrenamiento empleado es el GDB con un coeficiente de aprendizaje α y de atenuación β, constante en todas las pruebas de validación e igual a 0,01 y 0,001 respectivamente. La versión equivalente de este algoritmo en MATLAB (versión 6.5) es "Traingd".

En cada prueba, los pesos del PMC se inicializan aleatoriamente una sola vez. Esto se explica porque el método propuesto en este trabajo comienza con la topología de red más elemental (sin capas ocultas). De esta manera, con la primera topología se puede demostrar analíticamente que

siempre se converge al mismo punto de mínima sobre la superficie de error. El valor de los pesos finales obtenidos con la topología de red inicial, se utiliza como semilla para generar los demás pesos de la red.

Aplicando el método de la observación estructurada en la tabla 4, columna Gráfica generalización, se puede concluir que con el método propuesto se obtuvo una aceptable habilidad en la generalización de la red.

Tabla 4 Habilidad de aprendizaje

Función	*Topología generada con el método propuesto*	*ECM*	*Gráfica Generalización*
Iteración simple $g^{(1)}(x_1,x_2)=10{,}391((x_1-0{,}4)(x_2-0{,}6)+0{,}36)$		0,0031	
Radial $g^{(2)}(x_1,x_2)=24{,}234\left(r^2(0{,}75-r^2)\right)$, $r^2=(x_1-0{,}5)^2+(x_2-0{,}5)^2$		0,0100	
Iteraciones Complicadas $g^{(5)}(x_1,x_2)=1{,}9\left(1{,}35+e^{x_1}\sin\left(13(x_1-0{,}6)^2\right)e^{-x_2}\sin(7x_2)\right)$		0,1100	

Finalmente, los resultados de la clasificación se compararán con las arquitecturas halladas por los siguientes métodos:

- Método Ash-Hirose (MAH). [9, 10]: generación dinámica de la topología. Es el mismo método expuesto en este artículo pero sin la técnica de la búsqueda forzada de los mejores mínimos locales.
- Método Random (MR): elección de la topología aleatoriamente. En este caso se selecciona la misma topología que se obtuvo con el método propuesto, pero a diferencia de éste, se inicializan nuevamente y de forma aleatoria los pesos de la topología o arquitectura de la red. Ésta última permanece estática durante la ejecución del algoritmo de entrenamiento GDB.
- Método Algoritmos Genéticos (MAG) [14]: generación de la topología usando Algoritmos Genéticos.
- Método Masters (MM) [13]: generación de la topología aplicando la regla de la pirámide geométrica.

Como se observa en la tabla 5, el método propuesto fue el único que pudo resolver la función radial. El histograma generación de la topología es como se ilustra en la tabla 6.

Ventajas y limitaciones del método propuesto

Como se puede observar en la tabla 5, el método masters (MM) no solucionó ninguna de las pruebas de validación. Por el contrario, el único método que pudo aproximarse a la función radial, fue el propuesto en este artículo. Esto se explica porque la limitante de las metodologías convencionales para generar la topología de una red (constructivos [9], destructivos [10], heurísticos [20], [13], técnicas evolutivas [14]) se encuentra

Tabla 5 Complejidad de la topología generada. Arquitecturas halladas por otros métodos

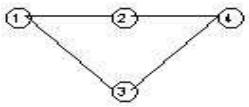
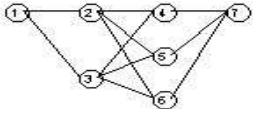
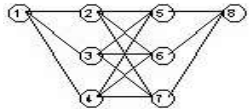
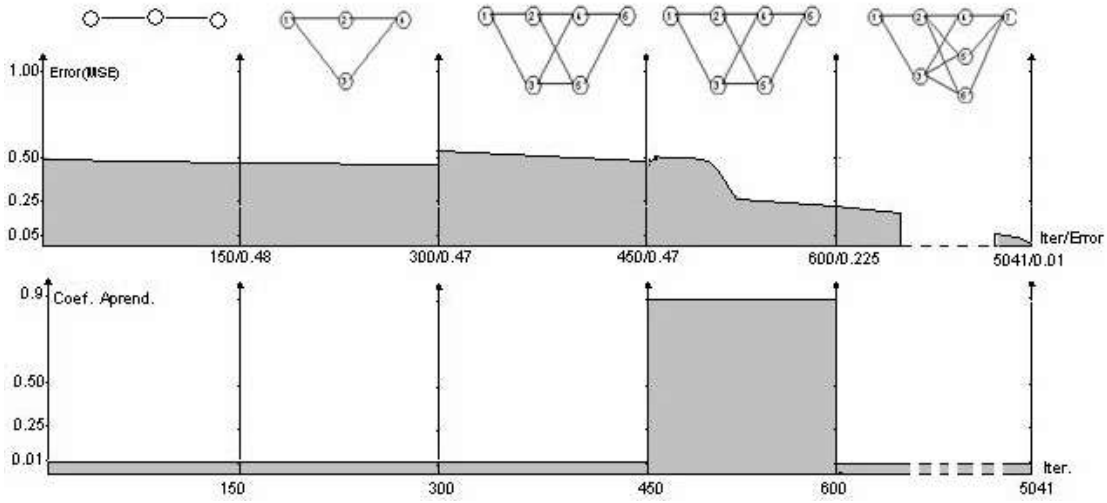
		ECM/iteraciones				
Función	***Topología final obtenida con los diferentes métodos***	***M. Propuesto***	***MAH***	***MR***	***MAG***	***MM***
Iteración simple		0,0031/3491	0,0031/3491	0,0031/2439	0,0031/NA	0,1770/NA
Radial		0,0100/5041	0,4890/NA	0,4890/NA	0,4890/NA	0,4890/NA
Iteraciones complicadas		0,1100/6640	0,1100/6640	0,1570/6640	0,1100/NA	0,.3900/NA

Tabla 6 Histograma generación topología con el método propuesto. Función radial



en el algoritmo de entrenamiento de primer orden GDB, el cual quedó atrapado, en este caso, en un punto de la superficie de error, y es mal candidato a ser mínimo local. Como se observa en la tabla 6, el método propuesto realizó una *búsqueda forzada*, entre el rango de iteraciones (450, 600), de los mejores puntos próximos de mínima local. Esto se hizo aumentando transitoriamente el valor del coeficiente de aprendizaje de 0,01 a 0,9, encontrándose una mejor solución que la obtenida con cualquiera de los otros métodos utilizados.

Por otra parte, también se puede observar en la tabla 5 que el método AG es el que tiene el mayor costo computacional. Se entiende esto porque el algoritmo AG itera tantas veces como la cantidad de topologías presentadas en la población inicial. En cada iteración se realizan los procesos de creación, evaluación, selección y reproducción sexual de topologías. Las resultantes son entrenadas con el algoritmo GDB. Para solucionar las pruebas de validación, el método random (MR) necesitó casi las mismas iteraciones utilizadas por el método propuesto. Esto se entiende porque se eligió una topología de red con la cual se sabía que se podían resolver. No obstante, lo usual es que se desconozca el tipo de arquitectura para tal fin. La búsqueda de la topología apropiada, normalmente teniendo como base una apreciación de carácter empírico, aumenta considerablemente el costo computacional.

Los inconvenientes que presenta la metodología propuesta son:

- La velocidad en la convergencia depende de la habilidad que tenga el usuario para manejar el sistema informático.
- No se garantiza que finalmente se alcance el mínimo absoluto sino un mínimo relativo (local).
- El método funciona dependiendo de la complejidad de la función a deberá ser aprendida.

Conclusiones

La contribución más significativa de la metodología propuesta en este trabajo, y que no se encuentra reportado en la literatura, es el de crear un sistema híbrido entre los métodos constructivos y destructivos para generar topologías, con la técnica de la b*Búsqueda forzada de mejores mínimos locales* sobre la superficie de error.

Si se compara esta propuesta con cualquiera de las demás existentes, es la que menor costo computacional requiere para alcanzar una solución

aproximada que permita resolver un problema en particular.

También es claro que el método propuesto no necesita especular demasiado sobre la estructura y dimensiones que debe adoptar la topología de red, ya que ella sola se va adaptando a las necesidades.

Finalmente, todas las pruebas de validación indican que la topología de un PMC depende de:

- El número de entradas y salidas.
- El número de tuplas de entrenamiento.
- La complejidad de la función que deberá ser aprendida.
- El algoritmo de entrenamiento.

En futuros trabajos se propondrá generar topologías de RNA del tipo PMC con el método propuesto en este artículo, pero utilizando para el entrenamiento algoritmos de segundo orden.

Referencias

1. J, Hilera. *Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones.* Alfaomega, MADRID 2000. p. 132-153.
2. K, Peng. "An algorithm to determine neural network hidden layer size and weight coefficients". *Proceedings of the 15th IEEE International Symposium on Intelligent Control (ISIC 2000).* Rio Patras. Greece. 2000.
3. T. Yau. "Constructive Algorithms for structure learning in feedforward Neural Networks for Regression Problems". *IEEE Transactions on Neural Networks.* Vol. XX. 1999. p. 16.
4. S. Ergeziner, E. Thomsen. "An accelerated learning algorthim for multilayer perceptions: optimization layer by layer". *IEEE Trans Neural Network.* Vol 6. 1995. p 31-42.
5. F. M. Coetzee, V. L. Stonik. "Topology and geometry of single hidden layer network least square weight solution". *Neural Comput.* Vol. 7. 1995. p 672-705.
6. L. M. Reynery. "Modified backpropagation algorithm for fast learning in neural networks". *Electron Left.* Vol. 26. 1990. p. 10-18.
7. D. H. Park. "Novel fast training algorithm for multiplayer feedforward neural networks". *Electron. Left.* Vol 26. 1992. pp. 1-100.
8. A. Sperduti. Staria A. "Speed up learning and network optimization with extended bacpropagation". *Neural Network,* Vol 6. 1993. pp. 365-383.
9. T. Ash. "Dynamic node creation in back propagation networks". *Proceedings of Int. Conf. On Neural Networks.* San Diego. 1989. pp. 365-375.
10. H. Hirose. "Back-propagation algorithm with varies the number of hidden units". *Neural Networks.* Vol. 4, 1991. pp. 20-60.
11. S. Aylward, R. Anders. "An algorithm for neural network architecture generation". *A1AAA Computing in Aerospace VIII.* Baltimore, 1991. p 30-90.
12. R. Tauel. *Neural network with dynamically adaptable neurons.* N94-29756, 1994. pp. 1-12
13. T. Masters, "*Practical Neural Networks recipes in C++*". Ed. Academic Press, Inc. 1993. pp. 173-180.
14. X. Yao, "Evolving Artificial Neural Networks. School of Computer Science". *Proceedings IEEE*. Septiembre, 1999.
15. E. Fiesler. "Comparative Bibliography of Ontogenic Neural Networks". *Proccedings of the International Conference on Artificial Neural Networks*, ICANN 1994.
16. N. Murata, "Network Information Criterion-Determining the number oh hidden units for an Artificial Neural Network Model". *IEEE Trans. on Neural Networks*. Vol. 5. 1994.
17. B. Martín del Brio, *Redes Neuronales y Sistemas Difusos.* AlfaOmega. 2002. pp. 64-66, 69.
18. V. Kurková. "Kolmogorov´s theorem and multiplayer neural networks". *Neural Networks*. Vol. 5. 1992. pp. 501-506.
19. L. Prechelt. "Early Stopping – But When?". *Neural Networks: Tricks of the Trade*. 1998. pp. 55-70.
20. D. Hush, B. Horner. "Progress in supervised neural networks: What´s new since Lipmman?". *IEEE Signal Proc. Mag*., 1993.
21. L. Hwang, M. Maechler, S. Schimert, "Progression modeling in back-propagation and projection pursuit learning". *IEEE Transactions on Neural Networks*. Vol. 5. 1994. pp. 342-353.