Comparación de la Aplicación de Métodos de Primer y Segundo Orden para el Aprendizaje de Redes Neuronales.

Ph. Dr. Ricardo Ribeiro Gudwin, Ing. Lizet Liñero Suárez, Lic. José Antonio Sánchez Guerrero

Resumen: Este trabajo presenta una comparación de la aplicación de métodos de primer (Algoritmos Genéticos) y segundo (“*Back-Propagation*”) orden para el aprendizaje de las Redes Neuronales, mostrando de esta manera un método donde se combinan estas técnicas clásicas de la Inteligencia Artificial para la solución de problemas de aproximación de funciones y de otra índole que requieran de la aplicación de estas técnicas. Además se presenta la aplicación de las Redes de Objetos que no son más que una nueva herramienta para modelar sistemas que exhiben un carácter de modificación en su propia estructura y que no pueden ser modelados por herramientas clásicas como Automatas Finitos y Redes de Petri.

1. Introducción

La utilización hoy en día de los métodos tradicionales de inteligencia artificial Algoritmo Genético (GAs)[2,3,4], Redes Neuronales (RN) [1,8,11], Lógica Fuzzy (FL)[1] y otras, son cada vez mas aplicadas a la solución de diversos problemas de la vida práctica. En la mayoría de los casos los problemas son enfrentados con la combinación de estos métodos

Existen diversas aplicaciones donde se utilizan los diferentes métodos tradicionales combinados [5,7,9,12,13], este trabajo consiste en un estudio de la aplicación de los Algoritmos Genéticos para el aprendizaje de las Redes Neuronales y una comparación de esta técnica con el tradicional algoritmo del *Back-Propagation* [8,11] para el aprendizaje de algunas Redes Neuronales.

El trabajo está organizado de la siguiente manera. En la sección 2 y 3 se presentan de manera general los Algoritmos Genéticos y las Redes Neuronales respectivamente. La aplicación de las técnicas de GAs y *Back-Propagation* para el aprendizaje de la Red Neuronal es descrita en la sección 4. Por último se dan a conocer las conclusiones de este trabajo en la sección 5.

2. Algoritmos Genéticos

Los Algoritmos Genéticos (GAs) son métodos adaptables que pueden ser utilizados para buscar la solución de problemas de optimización y búsqueda. Ellos están basados en los procesos genéticos de los organismos biológicos. En

todas las generaciones la población evoluciona de acuerdo con los principios de la selección natural y supervivencia del mejor individuo.

En la naturaleza los individuos de una población compiten con los otros por los recursos naturales, como la comida, agua y atraer a su pareja, aquellos individuos más aptos para la supervivencia y la acción de parear, obtendrán relativamente más número de descendientes. Esto significa que los genes más adaptados y aquellos individuos más convenientes pueden incrementar el número de individuos para las próximas generaciones. La combinación de las buenas características de diferentes antepasados pueden generar descendientes más capaces de adaptarse al medio ambiente

Los GAs emplean una analogía directa al comportamiento natural. Ellos trabajan con una población de "individuos", cada uno representa una posible solución para el problema a tratar. A cada individuo le es asignado un "valor apropiado" en relación con la solución que el representa para el problema.

La mayoría de las nuevas soluciones posibles se obtuvieron por la selección del mejor individuo en la generación actual y su combinación con otro individuo produciendo un nuevo conjunto de individuos. Esta nueva generación posee una alta proporción de las características de los mejores individuos de la generación anterior. Si el GA es bien diseñado entonces la población converge para la solución óptima del problema.

El potencial de los GAs consiste en el hecho de que es una técnica robusta y puede ser utilizada en diversos problemas, incluyendo aquellos que tienen cierta dificultad para su solución por otros métodos. Los GAs no garantizan encontrar una solución óptima del problema, pero procuran en general, una buena solución para el problema y de una manera rápida.

Los GAs no son los únicos algoritmos que están basados en analogía con la naturaleza. Las Redes Neuronales son técnicas basadas

en la conducta del neuronio en el cerebro. Ellas son utilizadas para dar solución a una variedad de tareas de clasificación, por ejemplo, reconocimiento de patrones, procesamiento de imagenes y sistemas expertos. El uso de GAs para diseñar Redes Neuronales son áreas de investigaciones recientes.

2.1 Principios básicos

Un algoritmo genético simple puede ser representado por la siguiente figura:

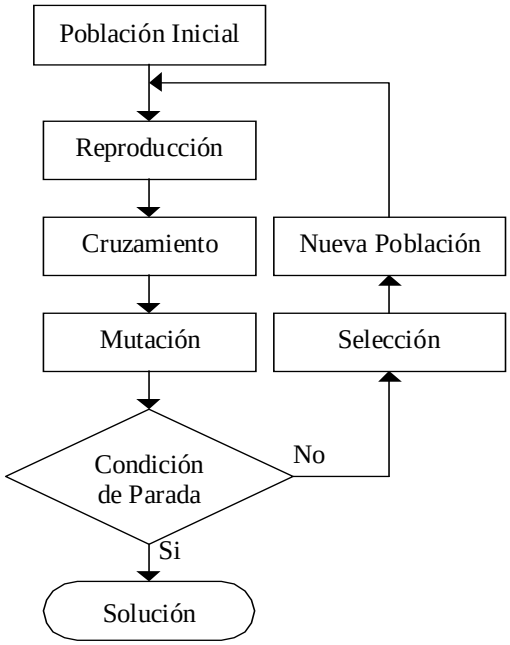


Figura 1. Principio de funcionamento de un GA.

Antes de especificar cada uno de los operadores genéticos, reproducción, cruzamiento y mutación, explicaremos dos aspectos importantes para los GAs:

Codificación: Este paso es importante, aquí se realiza una codificación para el conjunto de parámetros del problema. Existen varios tipos de codificación para estos parámetros, la más utilizada es la codificación de *Gray*, algunos estudios muestran que esta codificación es ligeramente más efectiva que la codificación binaria para aquellos problemas de optimización con dos variables, para otros casos se emplea la codificación no binaria, otros estudios demuestran que este tipo de codificación es más rápida, más consistente y se obtienen resultados más precisos. Este es un aspecto que depende del tipo de problema a solucionar.

Función de Aptitud (*fitness function*): La Función de Aptitud es ideada para cada problema a ser resuelto. Esta función para cada cromosoma particular retorna un valor númerico que representa la “aptitud” de este individuo para su supervivencia en ese

medio, este valor se supone sea proporcional a la “utilidad” o “habilidad” del individuo representado por el cromosoma.

2.1.1 Cruzamiento

Este operador genético básicamente consiste en tomar dos individuos de la población y seleccionar aleatoriamente una posición para comenzar a realizar el intercambio de los genes de cada individuo. Esta técnica es conocida como cruzamiento en un punto simple (Figura 2).

Existen otras técnicas para realizar el cruzamiento de los individuos, una es cruzamiento con dos puntos de cruzamiento, esta consiste en marcar con dos puntos el segmento del cromosoma para llevar a cabo el intercambio. (Figura 3).

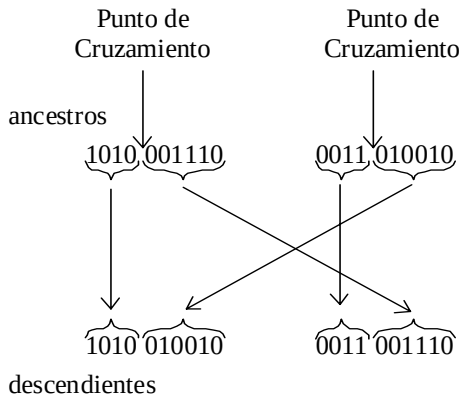


Figura 2. Cruzamiento con punto simple.

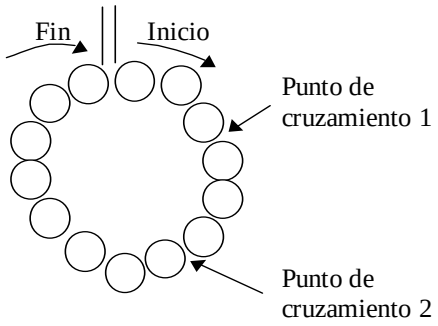


Figura 3. Cruzamiento con dos puntos.

Otra de las técnicas empleadas para realizar el cruzamiento, es el cruzamiento uniforme, este consiste en generar una máscara de cruzamiento aleatoria, donde 1 significa que ese gene será copiado en el descendiente del padre número 1 y 0 del padre número 2 (Figura 4), este proceso es repetido con una nueva máscara de cruzamiento aleatoria para generar el otro descendiente con estos mismos padres.

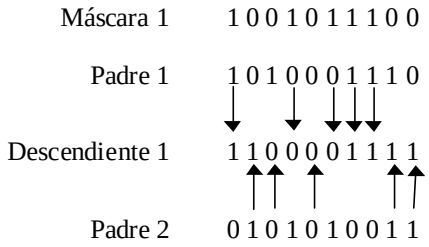


Figura 4. Cruzamiento uniforme.

Se pueden utilizar otras técnicas de cruzamiento conociendo el dominio del problema, pero estas serán muy específicas para esos problemas en cuestión.

2.1.2 Mutación

El operador genético Mutación es aplicado a los descendientes después de efectuado el cruzamiento. Este operador consiste básicamente en alterar de manera aleatoria cada gene del cromosoma (Figura 5) con una probabilidad pequeña (en general 0.001).

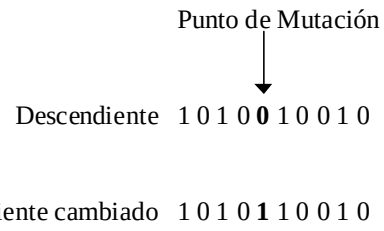


Figura 5. Mutación simple.

Este operador es tradicionalmente visto como un operador de fondo, responsable de la introdución inadvertida de los valores perdidos de un gene, insertando un pequeño elemento a la búsqueda aleatoria en la vecindad de la población que está convergiendo.

2.1.3 Reproducción o Selección de los individuos.

Este es un operador muy importante para el desempeño de los GAs. Este operador es el encargado de realizar la selección de aquellos individuos de la población para formar la nueva población que podemos llamar de población de trabajo ("*maping pool*"), donde tienen lugar los otros operadores genéticos (mutación y cruzamiento), formando después la nueva generación.

3. Redes Neuronales

Una Red Neuronal Artificial es un sistema de procesamiento de la información que ejecuta características en común con las Redes Neuronales Biológicas. Las Redes Neuronales Artificiales fueron desarrolladas como generalización de modelos matemáticos del conocimiento humano o de los neuronios biológicos.

Una Red Neuronal consiste en un largo número de elementos simples del procesamiento llamado neuronios, unidades o nodos, cada neuronio es conectado a otro neuronio por medio de un eslabón de comunicación directo; a cada uno se le asocia un peso. Los pesos representan la información usada por la red para resolver el problema. Las Redes Neuronales pueden ser aplicadas a diversos problemas, como

reconocimento de patrones y clasificación de patrones. Una Red Neuronal simple puede estar representada como se muestra:

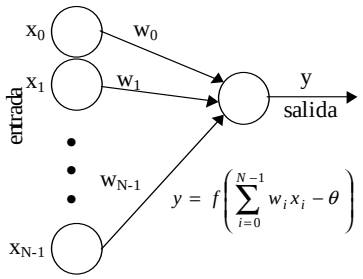


Figura 6. Rede Neural Simples.

El nodo es caracterizado por un *Threshold* interno o un desplazamiento θ y por un tipo de no-linearidad. Las no-linearidades más usadas son:

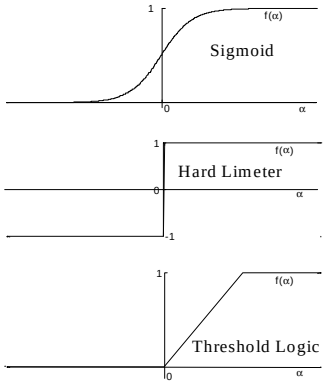
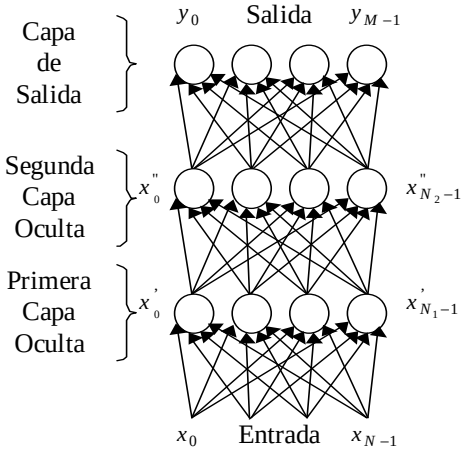


Figura 7. No-linearidades más utilizadas en Redes Neuronales.

3.1 *Red Multi-Layer Perceptron*

Las redes *Multi-Layer Perceptron* son del tipo *Feed-Forward* con una o más capas de nodos entre los nodos de la entrada y de la salida. Las capas adicionales contienen unidades ocultas, los nodos que no son directamente conectados a uno u otro nodo de la entrada y de la salida (Figura 8). La misma aprovecha el algoritmo de *Back-Propagation* que es un algoritmo de gradiente iterativo, utilizado para disminuir el error cuadrático medio entre la salida real de la red y la salida deseada.



$$y_l = f\left(\sum_{k=0}^{N_2-1} W''_{kl}\, x''_k - \theta''_l\right) \quad 0 \le l \le M-1$$

$$x''_k = f\left(\sum_{j=0}^{N_1-1} W'_{jk}\, x'_j - \theta'_k\right) \quad 0 \le k \le N_2 - 1$$

$$x'_j = f\left(\sum_{i=0}^{N-1} W_{ij}\, x_i - \theta_i\right) \quad 0 \le j \le N_1 - 1$$

Figura 8. Red *Multi- Layer Perceptron.*

El algoritmo *Back-Propagation* se muestra a continuación:

Paso 1:

Inicialización de pesos y desplazamientos

Paso 2:

Presentación de la entrada y salida deseada

Se presentan valores contínuos del vector de entrada $x_e, x_{1,\ldots} x_{n-1}$ y se especifica la salida deseada $d_0, d_1, \ldots d_{m-1}$.

Passo 3:

Cálculo de las salidas actuales:

Se utiliza no-linearidad Sigmoide. Se calculan las salidas por las ecuaciones de la figura 8.

Paso 4:

Adaptación de los pesos:

Se emplea un algoritmo recursivo en los nodo de salida y trabaje en dirección a la primera capa oculta.

Ajustando los pesos por:

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta x_i'$$

donde:

W_{ij} : peso para nodo oculto *i* o para la entrada del nodo *j* en el tiempo.

x_j' : salida del nodo *i*.

η : término de ganancia

δ_j : término de error para nodo *j*

Sí el nodo *j* es un nodo de salida entonces:

$$\delta_j = y_j(1-y_j)(d_j - y_j)$$

onde:

y_j : salida real

d_j : salida deseada

Sí el nodo *j* es un nodo oculto entonces:

$$\delta_j = x'_j(1-x'_j)\sum_k \delta_k W_{jk}$$

Para lograr la convergencia rápida, se utiliza un término de momento en la ecuación de ajuste de pesos:

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j x'_i + \alpha(W_{ij}(t) - W_{ij}(t-1))$$

donde:

α : término de momento

que debe cumplir la siguiente condición:

$$0 < \alpha < 1$$

Paso 6:

Repetir a partir del *Paso2*.

4. Aplicación de un GA y el algoritmo *Back-Propagation* para el aprendizaje de una Red Neuronal

En esta sección presentaremos un GA y el algoritmo Back-Propagation como método

de entrenamiento para el aprendizaje de una red reuronal, para este objetivo tomamos una red neuronal del tipo Multi-layer Perceptron, que es una red del tipo Feed-Forward con una o dos capas entre los reunonios de entrada y de salida, para la aproximación de funciones, en este caso se empleó como función para ser aproximada la siguiente:

$$f(x, y, z) = e^{-z} \cdot (sen(x + y))^2$$

en el intervalo

$$-10\pi \le x \le 10\pi$$
$$-10\pi \le y \le 10\pi$$
$$-1 \le z \le 1$$

4.1 Estructura de la Red Neuronal empleada.

Para la aproximación de esta función, se empleó una Red que tiene 3 neuronios en la entrada, dos capas ocultas con 5 neuronios cada una y un neuronio en la salida. (Figura 9).

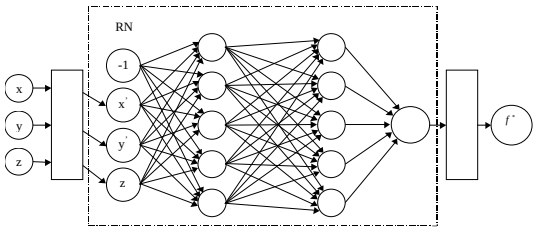


Figura 9. Estructura de la Red Neuronal empleada.

Las variables de los neuronios de entrada *x* e *y*, se les aplicó una normalización para que todos los valores de la entrada estuvieran en la misma región [-1,1]. Debido a esta normalización, se precisó de aplicar otra normalización al neuronio de salida, para poder obtener un resultado verdadero. Para la normalización de los valores de entrada se dividió cada valor por 10π (31.4) y para los valores de salida se utilizó la siguiente ecuación de la recta:

$$y = \frac{2.72 \cdot x - 0.272}{0.8}$$

4.2 Entrenamiento de la Red Neuronal

Para el entrenamiento de la Red Neuronal se escogió un conjunto de entrenamiento de 500 muestras, para el caso de la aplicación del GA y 1000 muestras para el caso del algoritmo *Back-Propagation*, distribuidas de forma uniforme. Se aplicó el método de entrenamiento por lotes, obteniendo los siguientes errores:

1. Error Cuadrático Medio

$$Eqm = \frac{\sum_{i=1}^{n} (Erro_i)^2}{n}$$

2. Error Medio

$$Em = \frac{\sum_{i=1}^{n} Erro_i}{n}$$

3. Error Máximo

$$Emáx = \max_{i=1}^{n} (Erro_i)$$

4.3 Aprendizaje de la Red Neuronal empleando un GA.

Para el aprendizado de la red neuronal se utilizó un GA para actualizar los pesos de la red. El GA empleado tiene como condición de parada 40 iteraciones, este valor fue escogido después de algunos ensayos.

4.3.1 Codificación empleada

Para la codificación de los genes a utilizar en el GA, se realizó un análisis de los diferentes métodos de codificación empleados en los GAs, escogiendo la codificación no binaria, debido a la particularidad de este problema, donde los valores posibles de los genes se encuentran en el intervalo real [-1,1], representando los pesos de la red neuronal.

4.3.2 Función de aptitud empleada

La función de aptitud (*fitness function*) en esta aplicación, fue una función que representara una composición entre el error cuadrático medio y el error máximo dada por la siguiente ecuación:

$$ff = \alpha \cdot Eqm + (1-\alpha) \cdot Emáx$$

Ahora la función objetivo para el GA es minimizar la función ff donde existe una relación de los errores a ser minimizados por el entrenamiento de la Red Neuronal.

4.3.2 Tamaño de la población empleada

Para definir el tamaño de la población a ser empleada para el trabajo del GA se realizó un estudio en el laboratorio con diferentes muestras de tamaño de poblaciones con el fin de encontrar el tamaño de la población para obtener los mejores resultados en el entrenamiento de la Red Neuronal.

Como se puede observar en las gráficas de la figura 10, la repercusión de la población es muy importante para el desempeño del GA, debido a, cuanto mayor sea el tamaño de la

población, mayor será el espacio de búsqueda de la mejor solución.

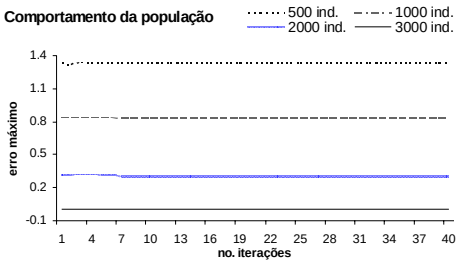
4.3.2 Operadores genéticos empleados en el GA.

Para el GA se emplearon diferentes operadores genéticos; se utilizó un operador de reproducción que sólo hace una copia de la población, para siempre preservar la población original.

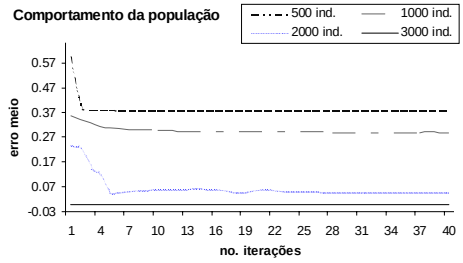
Otro de los operadores genéticos utilizados fueron el operador cruzamiento, en este caso utilizando la técnica de cruzamiento uniforme debido a su robustez, escogiendo los individuos aleatoriamente aplicando el método del *RouletteWheel* [4].

Para el operador mutación se hizo un análisis de la aplicación de diferentes mutaciones (figura 11). Para este análisis se realizaron 3 pares de mutaciones, en la leyenda las dos mutaciones se refieren a las mutaciones efectuadas a el 10% de los genes (aleatoriamente) del mejor individuo de la población, una a el ±1% del valor que puede aceptar el gene y otra a el ±30% del valor que puede aceptar el gene, las cuatro mutaciones se refieren a las dos mutaciones (±1% e ±30% del valor que puede aceptar el gene) efectuadas al 10% de los genes del mejor individuo de la población y al 50% de los genes del mejor individuo de la población (aleatoriamente), las seis mutaciones se refieren a las dos mutaciones (±1% e ±30% del valor que puede aceptar el gene) realizadas al 10%, 50% y 100% de los genes del mejor individuo de la población.

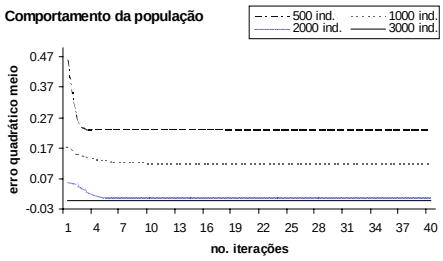
Como se puede observar en los diagramas de la figura 11, los mejores resultados se obtienen con la aplicación de las 6 mutaciones efectuadas sobre el mejor individuo de la población, esto se debe a la importancia que tienen los mecanismo de exploración del espacio de búsqueda (cruzamiento y mutación) para obtener los mejores resultados en la solución del problema.



(a)



(b)

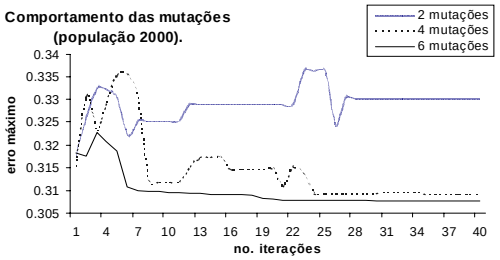


(c)

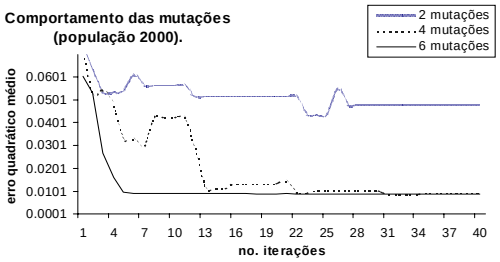
Figura 10. Comportamiento de la población respecto a los errores.

4.3.3 Función de Seleción Natural

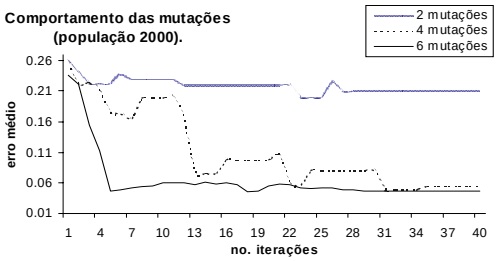
La selección de la función de selección natural es importante en la ejecución del algoritmo genético para obtener una solución óptima, debido a que esta selección es la que permite la utilización del espacio de búsqueda y obtener de esta manera la mejor solución.



(a)



(b)



(c)

Figura 11. Comportamiento de las mutaciones respecto a los errores.

Para este estudio se utilizaron cuatros funciones de selección natural (figura 12), una escoge los mejores individuos de la población del trabajo para generar la nueva población, otra escoge el 10% de los individuos más adaptados y el 90% de los menos adaptados, la tercera escoge el 30% de los mejores individuos y el 70% de los peores individuos y por último escoge el

50% de los mejores y peores individuos. En la figura 12 se muestra como el mejor comportamiento en sentido general, es el desempeñado por la función que escoge el 50% de los individuos más adaptados y los menos adaptados, observándose en la rápidez de conver-gencia y obtención de la mejor solución.

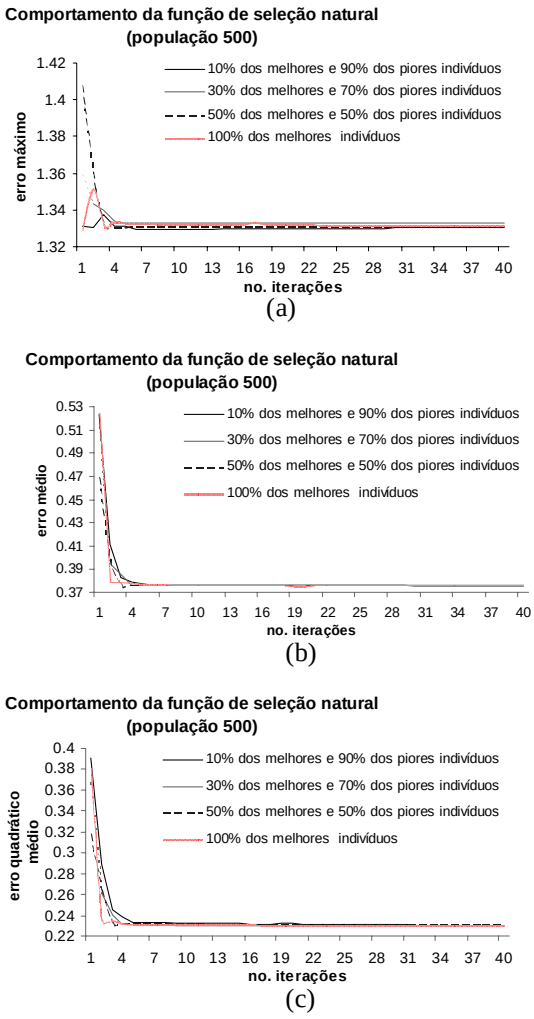


Figura 12. Comportamiento de las funciones de selección natural respecto a los errores.

4.4 Aprendizaje de la Red Neuronal por el algoritmo Back-Propagation

Para el aprendizaje de la Red Neuronal, se utilizó el algoritmo *Back-Propagation*, presentado en la sección 3.1; para actualizar los pesos de la red, el algoritmo tiene como condición de parada 1000 iteraciones, este valor fue escogido después de algunos ensayos e ir observando que a partir de este valor la red mantiene un valor fijo en los errores obtenidos.

4.4.1 Algunos ensayos en el entrenamiento variando parámetros del algoritmo *Back-Propagation*

El primer ensayo se realizó variando en la ecuación de ajuste de los pesos los valores del término de ganancia y sin utilizar el término de momento. Se utilizaron términos de ganancia de 0.01, 0.1 y 0.8, observándose el mejor comportamiento de la red para el valor del término de ganancia de 0.1. (figura 13). El segundo ensayo se efectuó tomando el mejor término de ganancia η=0.1 y utilizando la ecuación del ajuste de pesos con el

término de momento. Este término de momento se toma inicialmente con un valor fijo, bien pequeño, $\alpha = 0.00001$.

Los resultados obtenidos para este ensayo son comparados con el mejor comportamiento del ensayo anterior, sin término de momento, observando que al introducir el término de momento la red converge más rápido. (figura 14).

El tercer ensayo se realizó utilizando en la ecuación de ajuste de pesos el mejor término de ganancia η=0.1 y el término de momento para este caso va a ir diminuyendo con las iteraciones a través de la ecuación:

$$y = k\ell^{-ct}$$

onde:

k: constante

t: es el número de iteraciones

Primeramente se toman los siguientes valores:

k=0.1 y *c*=0.9

Luego se cambia el valor de la constante, *k*=0.5 y los resultados se comparan con el caso de *k*=0.1, observándose que el mejor comportamiento es para el valor de *k*=0.5. Los resultados obtenidos se pueden observar en el gráfico de la figura 15.

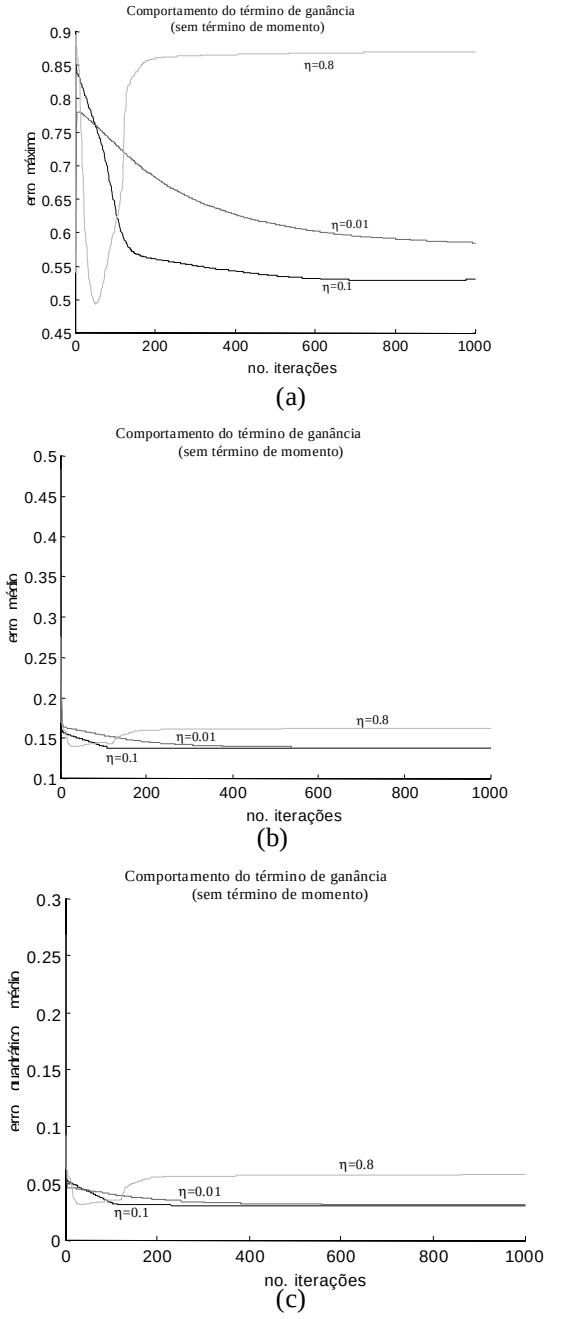


Figura 13. Comportamiento del término de ganancia, sin término de momento, respecto a los errores.

El último sería una comparación entre el comportamiento de la red sin la utilización del término de momento, utilizando este último fijo y variable con las iteraciones. Llegando a la conclusión de que la red converge más rápido con término de momento variable (figura 16).

4.5 Comparación de los métodos de aprendizaje utilizados para la Red Neuronal.

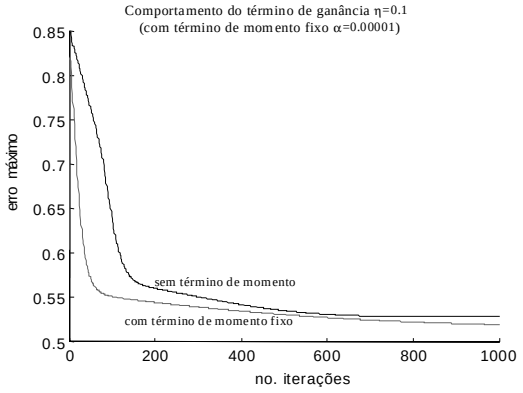
Para el análisis de la efectividad de los métodos empleados para el aprendizaje de la Red Neuronal (Algoritmo *Back-Propagation* y Algoritmo Genético) se realizó una comparación entre los diferentes errores obtenidos por ambos algoritmos y el tiempo de utilización de CPU para obtener los errores.

Como se observa en la figura 17 los mejores resultados obtenidos son los producidos por el GA con un mayor costo computacional, el algoritmo *Back-Propagation* podría alcanzar los mismos niveles de errores que los alcanzados por el GA, lo que sucede aquí con este algoritmo es que el a conseguido encontrar un mínimo local, pero no un mínimo global, escogiendo el conjunto de pesos iniciales de manera más adecuada este algoritmo podría converger a un mínimo global.

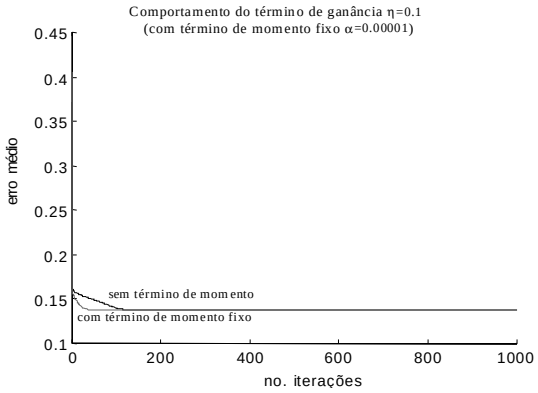
4.6 Aplicación de las Redes de Objetos para modelar el aprendizaje de la Red Neuronal utilizando un GA.

Las Redes de Objetos [10] son un tipo especial de Sistemas de Objetos [10]. Ellas son utilizadas para modelar sistemas que exhiben un caráter de modificación en su propia estructura, los que no puden ser modelados por herramientas clásicas, tales como Automatas Finitos y Redes de Petri.

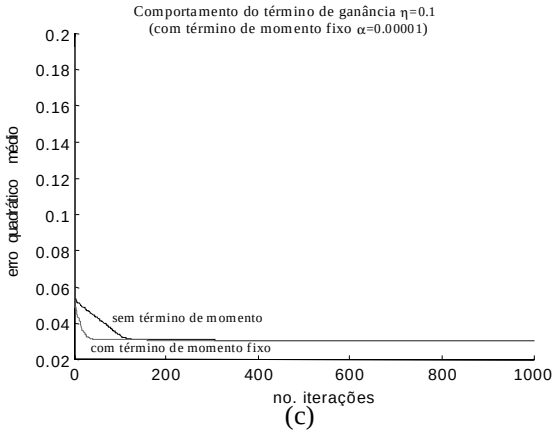
El comportamiento de las Redes de Objetos puede ser analizado de manera a identificar ciertas características de los sistemas por ellas modelados. La figura 18 muestra la Red de Objetos desarrollada para modelar el mecanismo de aprendizaje de la Red Neuronal diseñada para aproximar la función f(x,y,z).



(a)

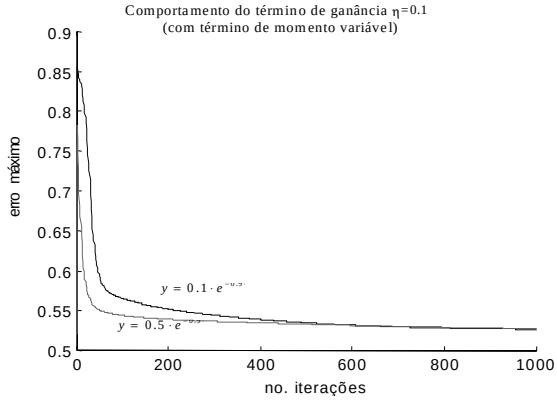


(b)

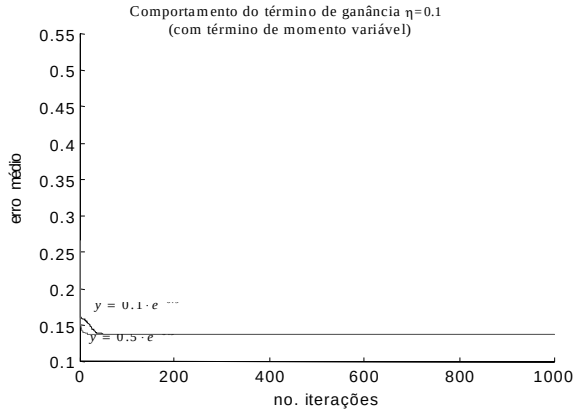


(c)

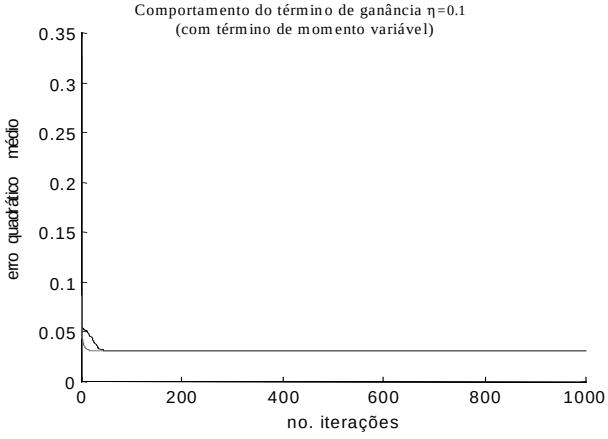
Figura 14. Comportamiento del término de ganancia η=0.1, con término de momento fijo, respecto a los errores.



(a)



(b)



(c)

Figura 15. Comportamiento del término de ganancia η=0.1, con término de momento variable, respecto a los errores.

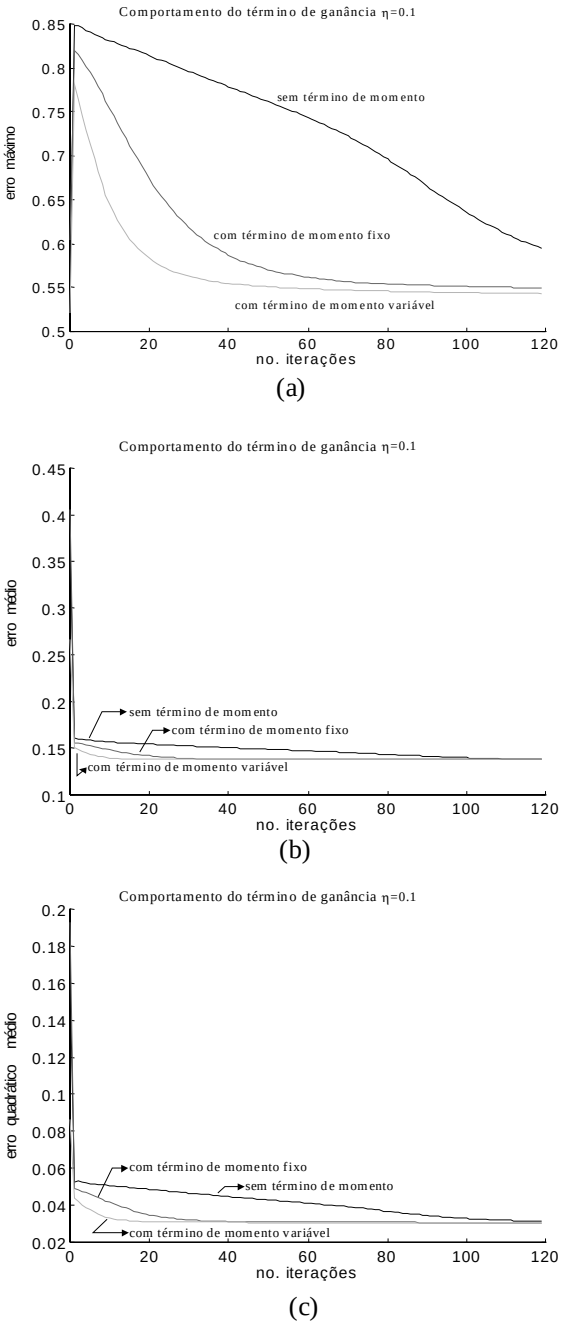


Figura 16. Comportamiento del término de ganancia η=0.1, con los diferentes términos de momentos, respecto a los errores.

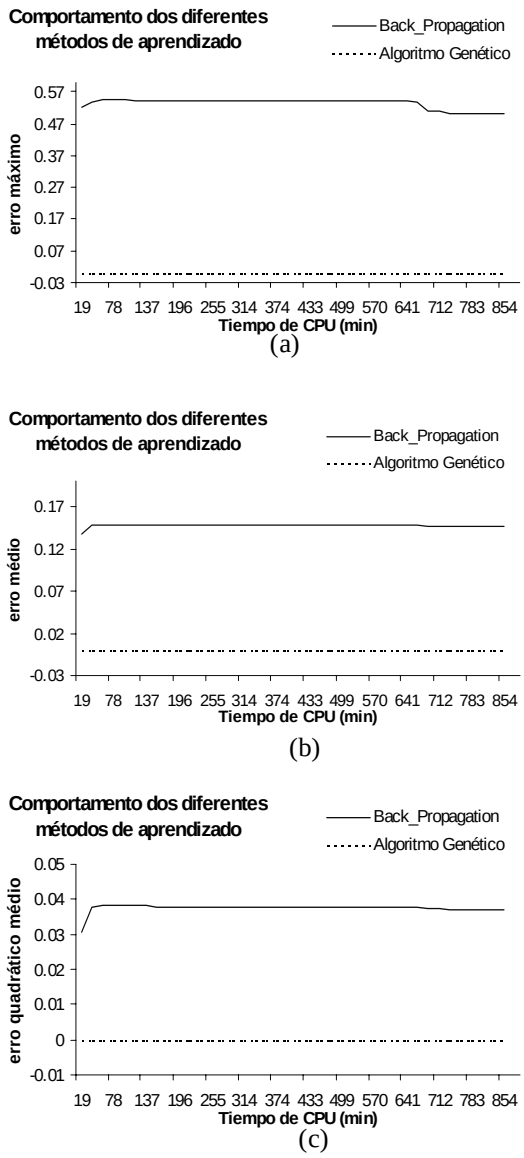


Figura 17. Comportamiento de los algoritmos de aprendizaje.

Algoritmo General de Trabajo de la Red de Objetos definida.

Un algoritmo general de trabajo de la Red de Objetos para el modelado del aprendizaje de la Red Neuronal basada en Algoritmos Genéticos Simples es el siguiente:

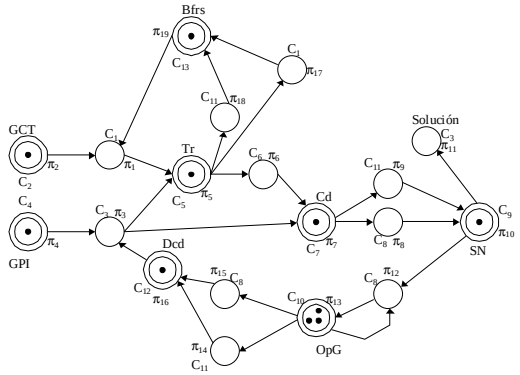


Figura 18. Red de Objetos para modelar el aprendizaje de la Red Neuronal utilizando un GA.

Leyenda	
GCT	Generador de muestras.
GPI	Generador de Redes Neuronales.
Tr	Entrenador de Redes Neuronales.
Cd	Codificador para los cromosomas del GA.
SN	Operador de Selección Natural para el GA
OpG	Operadores Genéticos para el GA.
Dcd	Decodificador de cromosomas del GA en matrices de pesos para el entrenamiento de la RN.
Solución	Representa las matrices de pesos óptima para el trabajo de la RN.
Bfrs	Representa un buffer

El objeto de la clase C_2 genera un conjunto de objetos de la clase C_1 que es colocado en π_1, creando de esta manera el conjunto de entrenamiento de la Rede Neural. El objeto de la clase C_4 genera un objeto de la clase C_3.

El objeto de la clase C_5 asimila un objeto de la clase C_3 y va ir asimilando destructivamente los objetos de la clase C_1 y efectua la validación de la RN, actualizando las variables internas y generando una copia del objeto C_1 asimilado destructivamente para π_{17}, cuando no existan más objetos en π_1, el objeto de la clase C_5 genera un objeto de la clase C_{11} para π_{18} y un objeto de la clase C_6 para π_6.

El objeto de la clase C_7 asimila destructivamente los objetos de la clase C_3 y C_6 que se encuentran en π_3 y π_6 respectivamente y que tengan el atributo de identificador *(id)* iguales, generando el objeto de la clase C_8, hasta que no existan más objetos en π_6 y π_3. Entonces, el objeto de la clase C_7 genera un objeto de la clase C_{11} con valor 0 si la cantidad de objetos generados para π_8 es igual al tamaño de la población. Sí la cantidad de objetos es mayor que el tamaño de la población, entonces el objeto de la clase C_7 genera un objeto de la clase C_{11} con valor 1, que es colocado en el lugar π_9.

El objeto de la clase C_9 tiene un comportamiento un tanto complicado, su

régimen de trabajo es el siguiente, sí el contador interno de este objeto es igual al número de iteraciones que es la condición de parada del AG, entonces genera un objeto de la clase C_3 que representa la solución del problema; en caso de no ser igual y existe un objeto de la clase C_{11} en π_9 con valor 0, entonces ejecuta la función de transformación f_7 que realiza una copia del objeto de la clase C_8 que se encuentra en π_8 para π_{12}; sí el objeto de la clase C_{11} tiene como valor 1, entonces la función de transformación que se activa es f_8 que lleva a cabo una selección de los mejores individuos (30%) y peores individuos (70%) y son colocados en π_{12} , en caso de no existir objetos en π_8, entonces el objeto de la clase C_9, sólo asimila al objeto de la clase C_{11} en π_9 e incrementa el contador interno que representa la condición de parada del GA.

Los objetos de la clase C_{10} en π_{13} representan los operadores genéticos (reproducción, cruzamiento y mutación), los mismos son activados de la siguiente manera: un objeto de la clase C_{10} es activado cuando existe un objeto de la clase C_8 en π_{12}, si el *flag* interno es 0, entonces se activa el operador reproducción que hace una copia del objeto C_8 en π_{12} para π_{15} y hace una transportación del objeto de la clase C_8 en π_{12} para π_{12} actualiza el atributo *id* en 1, indicando que ese objeto ya fue copiado, cuando no existan más objetos en π_{12} con atributo 0, entonces el *flag* interno es actualizado con el valor 1. Cuando el *flag* interno tiene valor 1, se activa el operador cruzamiento, el mismo asimila dos individuos basados en la técnica *RouletteWheel* y hace un cruzamiento uniforme de sus genes y colocando estos nuevos individuos en π_{15}, además de esto se hace una transportación de los objetos utilizados de π_{12} para π_{12}, actualizando el atributo *id* con valor 2 y actualiza el *flag* interno con valor 2.

Sí el *flag* interno es 2, se activa el operador mutación, que asimila destructivamente los objetos de la clase C_8 en π_{12} y selecciona el

mejor individuo y hace mutaciones en sus genes aleatoriamente, generando nuevos individuos para π_{15}, hasta que no existan más objetos en π_{12} y genera un objeto de la clase C_{11} que es colocado en π_{14}.

El objeto de la clase C_{12} en π_{16} es activado con la existencia de objetos de la clase C_8 en π_{15} y C_{11} en π_{14}, este objeto convierte al vector que representa el cromosoma (objeto de la clase C_8) en las matrices de pesos (Objetos de la clase C_3) para ser utilizadas por la RN en su nueva validación.

El objeto de la clase C_{13} es activado cuando existe un objeto de la clase C_{11} en π_{18}, asimilando destructivamente a un objeto de la clase C_1 en π_{17} y hace una copia de este para π_1, cuando no existan más objetos de la clase C_1 en π_{17}, asimila destructivamente el objeto de la clase C_{11} en π_{18}.

Conclusiones

A manera de conclusión de este trabajo podemos citar las siguientes:

- Aplicación de métodos de primer (Algoritmos Genéticos) y segundo orden (*Back-Propagation*) para el aprendizado de las Redes Neuronales.
- Combinación de algunos de los métodos clásicos de la Inteligencia Artificial para la solución de problemas.
- Uso de las Redes de Objetos como una herramienta para el modelaje del aprendizaje de Redes Neuronales utilizando Algoritmos Genéticos.

Referencias

[1.] B. Kosko, “Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence”, Prentice-Hall International, 1992.

[2.] D. Beasley, D.R. Bull, R.R. Martin. “An Overview of Genetic Algorithms: Part 1, Fundamentals” – University Computing 15(2), 1993 58-69.

[3.] D. Beasley, D.R. Bull, R.R. Martin. “An Overview of Genetic Algorithms: Part 2, Research Topics” – University Computing 15(4), 1993 170-181.

[4.] D.E. Goldberg. “Genetic Algorithms in search, optimization and machine learning.”, Addison-Wesley, 1989.

[5.] H. Saito, N. Kobayashi. “Evolutionary Computation Approaches to Halftoning Algorithm” – Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 787-791.

[6.] J. Arabas, Z. Michalewics, J. Mulawka. “GAVaPS- a Genetic Algorithm with Varying Population Size” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol I (1994) 73-78.

[7.] J. Renders, S. P. Flasse. “Hybric Methods using Genetic Algoritms for Global Optimization” – IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 26, No. 2, April 1996, 243-258.

[8.] L. Fausett. “Fundamentals of Neural Networks, architectures, algorithms and applications.”, Prentice Hall International, 1994.

[9.] L. Tsinas, B. Dachwald. “A Combined Neural and Genetic Learning Algoritm” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 770-774.

[10.] R.R. Gudwin. “Contribuições ao estudo matemático de sistemas inteligentes”, Tese de Doutorado, UNICAMP, Campinas 1996.

[11.] R. P. Lippmann. “An Introduction to Computing with Neural Nets”, IEEE ASSP Magazine, April, 1987.

[12.] S. Bengio, Y. Bengio, J. Cloutier. “Use of Genetic Programming for the Search of a New Learning Rule for Neural Networks” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol I (1994) 324-327.

[13.] S. G. Romaniuk. “Applying Crossover Operators to Automatic Neural Network Construction” – Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 750-752a.