REDES NEURONALES Y ALGORITMOS GENETICOS

Guillermo Choque Aspiazu
gchoque@inf.umsanet.edu.bo

Resumen
Uno de los problemas grandes con los que tropieza el diseño de las redes neuronales es la asignación de los pesos iniciales para los procesos de aprendizaje de la red neuronal, esta tarea normalmente se encarga a un experto del dominio, quien es el encargado de asignar de manera aproximada estos valores. En este trabajo se propone la revisión de los paradigmas de las redes neuronales y los algoritmos genéticos, con la perspectiva de lograr que la característica evolucionaría de los algoritmos genéticos puedan colaborar al ajuste adecuado y casi automático de los pesos iniciales de las redes neuronales.
Palabras clave: *red neuronal, algoritmo genético, propagación de errores, evolución neuronal, factor de momento.*

1. INTRODUCCION

El estudio de los sistemas neuronales artificiales, o redes neuronales, deriva del deseo de entender la función y estructura del cerebro humano. Muchos modelos de redes neuronales exhiben la habilidad para aprender una propiedad particular del cerebro. Desafortunadamente, dichos modelos contienen muchas consideraciones técnicas de arquitectura para su implementación. En los seres humanos, por otra parte, el cerebro está pre-cableado, gracias a muchos millones de años de evolución. Tal como la arquitectura del cerebro fue establecida por la selección natural y la genética, un objetivo actual constituye el desafío de establecer la posible arquitectura de un modelo de red neuronal con la ayuda de los algoritmos genéticos [Wasserman, 1993].

En una red neuronal, los arcos son etiquetados con valores denominados pesos, los cuales proporcionan una medida de la fuerza de la conexión entre las neuronas. Los pesos son ajustados a medida que el modelo aprende nuevas asociaciones. Los nodos son etiquetados con valores denominados activaciones, los cuales son una función de la entrada a la neurona en un punto particular en el tiempo. Mientras que es posible ajustar los valores de los pesos utilizando algoritmos genéticos, este enfoque generalmente converge de manera mucho mas lenta que cualquiera de los paradigmas de aprendizaje modernos. Una excepción es que los algoritmos genéticos pueden ser utilizados para implementar una porción razonable de la técnica denominada temple simulado[1] [Goldberg, 1989].

Uno de los problemas grandes con los que tropieza el diseño de las redes neuronales es la asignación de los pesos iniciales para los procesos de aprendizaje de la red neuronal, esta tarea normalmente se encarga a un experto del dominio, quien es el encargado de asignar de manera aproximada estos valores. En este trabajo se propone por una parte la revisión de los paradigmas de las redes neuronales y los algoritmos genéticos, y por la otra la perspectiva de lograr que estos últimos puedan colaborar a los primeros en el ajuste adecuado y casi automático de los pesos a través de las características evolucionarías asociadas a los algoritmos genéticos.

2. REDES NEURONALES BIOLÓGICAS

El sistema nervioso y especialmente el cerebro constan de una cantidad increíble de células nerviosas denominadas neuronas, conexiones entre las neuronas denominadas sinapsis y soportan células denominadas células glia. Las neuronas son las células más importantes del cerebro, debido a que realizan todo el proceso, mientras que el rol de las células glia es soportar las neuronas mediante el control de las

[1] Simulated annealing.

concentraciones iónicas al interior de las neuronas. La habilidad del cerebro para almacenar información es distribuida sobre todas las neuronas y el proceso de recordar la información almacenada es un proceso colectivo entre todas las neuronas de la red neuronal.

El modelo de red neuronal biológica es bastante común en los libros de texto sobre neuro fisiología. Una neurona consta de tres partes básicas:

1. ***Dendritas***: Un árbol de fibras de entrada que lleva los potenciales de acción de las neuronas transmisoras en la neurona.
2. ***Soma***: El cuerpo principal de la célula con un núcleo. Es en este lugar donde los potenciales de acción son construidos antes que la neurona se active.
3. ***Axón***: Es una fibra de salida simple que bifurca a otras neuronas y transmite los potenciales de acción generados por la neurona.

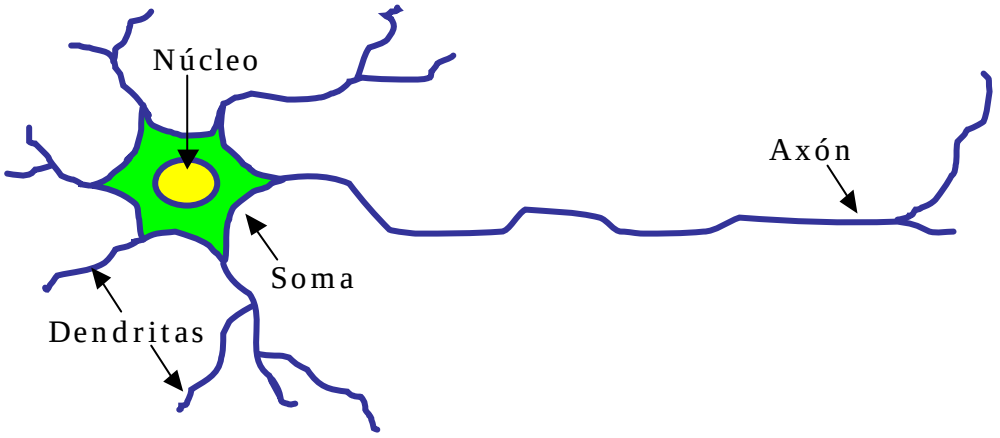

Fig. 1. Neurona biológica
Fuente: [Hilera y Martínez, 1995]

Las dendritas y los axones pueden ser vistos como cadenas de comunicación entre las neuronas, y no se conocen los detalles exactos acerca de cómo funcionan actualmente. Las partes más interesantes son el soma y la sinapsis los cuales constituyen los puntos de conexión entre los axones y las dendritas.

Las neuronas reciben señales (inputs) de otras neuronas vía conexiones sinápticas que pueden ser activadoras o inhibidoras. En función de las señales recibidas, una neurona envía a su vez una señal a otras neuronas por medio del axón.

Una neurona contiene un potencial interno continuo denominado potencial de membrana. Cuando éste excede un cierto valor umbral, la neurona puede transmitir todo su potencial por medio del axón. Se estima que el cerebro humano contiene más de cien mil millones (10^{11}) de neuronas y que hay más de 1000 sinápsis a la entrada y a la salida de cada neurona.

4. REDES NEURONALES ARTIFICIALES

Una red neuronal artificial consiste de un grafo dirigido de arcos y nodos. Los nodos representan células nerviosas idealizadas o neuronas, y los arcos representan las conexiones entre las neuronas. Los modelos de redes neuronales aceptan datos de entrada de un subconjunto de nodos que son referenciados como neuronas de entrada, produciendo como datos de salida un subconjunto de nodos que son referenciados como neuronas de salida. Los nodos que no son neuronas de entrada ni de salida son referenciados como neuronas ocultas.

Los arcos son etiquetados con valores denominados pesos, los cuales proporcionan una medida de la fuerza de la conexión entre las neuronas. Los pesos son ajustados a medida que el modelo aprende nuevas asociaciones. Los nodos son etiquetados con valores denominados activaciones, los cuales son una función de la entrada a la neurona en un punto particular en el tiempo. Existe una plétora de referencias acerca de las redes neuronales, un conjunto que cubre gran parte de dichas referencias puede ser consultada en [Hertz et al., 1991] [Haykin, 1994] y [Wasserman, 1993].

Una red neuronal, o de manera mucho mas adecuada una red neuronal artificial, puede ser considerada, desde un punto de vista aplicativo, como un dispositivo computacional que supera las deficiencias de las computadoras digitales seriales[2]. Se puede decir que una red neuronal artificial (RNA) imita al cerebro, en un intento demasiado simplista de explicarla. Ninguna RNA ha sido construida, hasta el momento, que sea capaz de lograr el poder de procesamiento equivalente a 10^{11} neuronas del cerebro humano. Además el modelo de una neurona artificial simple es una simplificación de la neurona real. Sin embargo, una RNA soporta una cierta similitud a una red neuronal real en el siguiente sentido:

1. Antes que ser programada se le enseña una tarea
2. La robustez es una de sus principales ventajas. Su rendimiento se degrada elegantemente como también se produce el fallo en sus neuronas.
3. Puede conducirse con datos contradictorios, inciertos o incompletos.
4. Es capaz de conducirse con una situación que no ha visto anteriormente a través de la generalización.
5. Es masivamente paralela y por consiguiente bastante rápida.

Por estas razones las redes neuronales son atractivas como maquinas computacionales. Las redes neuronales son esencialmente aproximadores de funciones [Vogiatzis, 1995]. Existe una fase de aprendizaje y una fase de generalización[3]. Durante la formación aprende a aproximar una función *f* de un conjunto de los datos de entrenamiento. Que significa, la red neuronal entrenada crea la función *f'* la cual se aproxima a la función real *f*. La fase de generalización consiste en evaluar *f'* para un vector de entrada.

4.1. Aprendizaje

El procedimiento de aprendizaje ampliamente conocido para una topología[4] de red neuronal es la propagación de errores hacia atrás [Brison and Ho, 1969][5]. Este procedimiento es empleado en el aprendizaje supervisado; es decir que una entidad conoce el dominio y está tratando de enseñar a la red neuronal. La red proporciona un par (I, D) donde I es el vector de entrada y D es el vector de salida deseado. La red inicialmente cuenta con un conjunto de pesos de valores aleatorios. Luego, el vector de entrada I es presentado como entrada a la red, la cual produce O como salida. La diferencia entre D y O es la medida de error, la cual es propagada hacia atrás para corregir los pesos de las conexiones. Los pesos son alterados para reducir la función de error. La propagación de errores hacia atrás es actualmente un método de gradiente descendente hacia un mínimo global. La ecuación 1 resume la regla de actualización de los pesos.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \qquad \textit{ecuación (1)}$$

[2] Quizá no lo que McCulloch y Pitts tienen en mente en [McCulloch and Pitts, 1943] cuando proponen un modelo de neurona, puesto que las computadoras digitales eran virtualmente inexistentes en ese tiempo.
[3] Esto se refiere al aprendizaje supervisado, que es una de las formas posibles de aprendizaje.
[4] El termino topología hace referencia al numero de unidades y su conectividad.
[5] Existe un lote de mejoras introducidas en el algoritmo clásico de propagación de errores hacia atrás: el Momento es introducido en [Rumelhart et al. 1986] y Quick-propagation es presentada en [Fahlman 1988]

Donde w_{ij} es el peso de la unidad i a la unidad j, n es la proporción de aprendizaje y E es la función de error global.

La elección de la proporción de aprendizaje es crucial para la propagación de errores hacia atrás. Idealmente, pasos infinitesimales conducirán el gradiente descendente a un mínimo local. Los resultados empíricos sugieren que debe también existir un mínimo global, o algo bastante cerrado al mismo. Sin embargo, en la vida real se está interesado en el aprendizaje rápido el cual implica una proporción de aprendizaje bastante grande. Por otra parte un valor grande de n podría hacer que en la búsqueda se extravíe el mínimo. Algunas mejoras a la situación descrita pueden ser realizadas si se consideran derivadas de orden superior para la superficie de error [Hilera y Martínez, 1995]

Como resumen de los requerimientos de la propagación de errores hacia atrás se puede decir que demanda una función de error continua donde la información del gradiente se encuentre presente. Actualmente, para lograrlo puede ser necesario, contar con derivadas de orden superior. Además, la topología de la red es crucial al tiempo que toma una red para aprender. Es también crucial a su habilidad de generalización. Una red que es bastante pequeña puede no aprender la función en la que uno está interesado, por otra parte una red grande puede no adaptar los datos del entrenamiento, y por consiguiente ser incapaz de generalizar. Lo que es más significativo, un gran numero de unidades o capas puede desacelerar severamente el tiempo de aprendizaje.

4.2. Diseño

Mientras que existe la propagación de errores hacia atrás para la definición de los pesos, no existe nada general para la definición de la topología. Por consiguiente, uno puede preguntarse ¿Cuantas unidades se deben utilizar? ¿Cómo será realizada la conectividad? o ¿Cuantas capas ocultas son necesarias?. Las preguntas formuladas no han recibido las respuestas adecuadas y completas. Sin embargo a continuación se realizan algunas consideraciones que resultan útiles a momento de comenzar con el diseño de las redes neuronales artificiales.

1. **Sobre el número de capas**. Ha sido probado que a lo más dos capas de unidades ocultas son suficientes para aproximar cualquier función, y a lo más una capa de unidades ocultas es suficiente para aproximar cualquier función continua [Cybenko, 1988]. Sin embargo, se puede o no conocer si la función es capaz de ser aprendida en un monto razonable de tiempo.
2. **Sobre el número de unidades**. Los algoritmos utilizados para determinar el número de unidades ocultas pueden ser clasificados en dos categorías: las clases destructivas y las constructivas. Los algoritmos destructivos empiezan de una gran red y luego se procede a la poda, removiendo enlaces y/o neuronas. Los algoritmos constructivos comienzan con una red bastante simple y hacen que la misma se transforme en una red más compleja en el curso del proceso. El algoritmo Upstart [Frean, 1990] es para unidades de salida binaria y termina con una arquitectura jerárquica. El algoritmo Tiling [Mezard and Nadal, 1989] construye redes con salidas opuestas, es decir ?1. Existe también el enfoque de Correlación-Cascada [Fahlman and Lebiere, 1990] el cual emplea unidades con salida sigmoidal.

Los métodos mencionados en el anterior párrafo son monotónicos en el sentido que en cada paso del procedimiento de creación la red adiciona o resta unidades ocultas. Este es actualmente un caso del ascenso de colinas con todas las deficiencias asociadas con este método; por ejemplo es bastante sencillo ser atrapado en un mínimo local. En ese caso la arquitectura explorada es abandonada y se adopta una nueva. Además solo pueden ser desarrolladas arquitecturas especificas. Las cuales de acuerdo con [Angeline et al., 1994] "*Crean una situación en la cual la tarea es forzada en la arquitectura antes que la arquitectura llegue a adecuarse a la tarea*"

5. RED DE PROPAGACION DE ERRORES

De manera simplificada, el funcionamiento de una red de propagación de errores consiste en el aprendizaje de un conjunto predefinido de pares de entrada salida dados como ejemplo, empleando un ciclo de propagación-adaptación de dos fases: primero se aplica un patrón de entrada a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada; es decir, el error disminuya [Hilera y Martínez, 1995].

La importancia de la red de propagación de errores radica en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes. Para aplicar esa misma relación, después del entrenamiento, a nuevos vectores de entrada con ruido o incompletas, generando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje. Esta característica importante, que se exige a los sistemas de aprendizaje, es la capacidad de generalización, entendida como la facilidad de generar salidas satisfactorias para entradas que el sistema no ha visto anteriormente en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le proporcionan las entradas de entrenamiento, y que pueda aplicar, además a entradas no presentadas durante la etapa de aprendizaje para clasificarlas según las características que compartan con los ejemplos de entrenamiento [Wasserman, 1993].

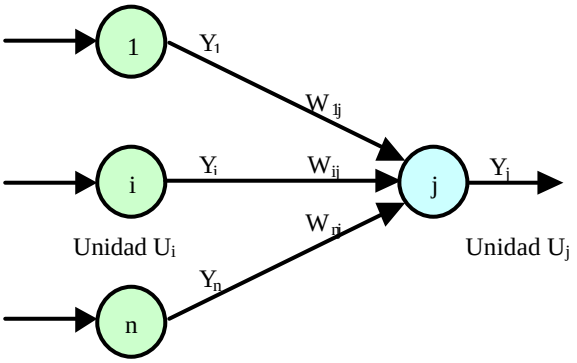

Fig. 2. Red neuronal simple
Fuente: [Hilera y Martínez, 1995]

El factor de momento (momentum factor), es una nueva característica que se adiciona a una red neuronal de propagación de errores hacia atrás. Este parámetro se encarga fundamentalmente de

acelerar el proceso de entrenamiento de la red neuronal. Un cambio simple en la regla de entrenamiento resulta precisamente en un entrenamiento mucho mas rápido a través de la adición de un factor de momento. La regla básica de entrenamiento para la red de propagación de errores hacia atrás es aproximadamente la siguiente: Cambio peso = Beta * error_salida * entrada. Adicionando el factor de momento la ecuación cambia de la siguiente manera: Cambio peso = Beta * error_salida * entrada + Alfa * cambio_previo_peso.

El segundo término en esta ecuación es el término momento. El cambio de peso, en la ausencia de error, será una constante multiplicada por el cambio en el peso previo. En otras palabras, el cambio en los pesos sigue en la dirección establecida hacia la meta. El termino momento es un intento para acumular el cambio de pesos en los procesos de movimiento y propagación de errores con el objetivo central de evitar su entrampamiento en los mínimos locales. De acuerdo a las observaciones realizadas en las redes neuronales de propagación de errores se recomienda que el factor de momento sea un número real que se encuentre en el rango de 0 a 0.5.

Para el factor proporción de aprendizaje (learning rate), el aprendizaje por corrección de error consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red, es decir, en función del error cometido en la salida. Una regla o algoritmo simple de aprendizaje por corrección de error es el siguiente:

$$?w_{ji} = ? y_i(d_j - y_j)$$

donde la ecuación representa de manera atómica:

$?w_{ji}$: Variación en el peso de la conexión entre las neuronas i y j.
y_j : Valor de salida de la neurona i.
d_j : Valor de salida deseado para la neurona j.
y_j : Valor de salida obtenido en la neurona j.
$?$: Factor de aprendizaje.

En el factor función de salida o transferencia, existen cuatro funciones de transferencia típicas que determinan distintos tipos de neuronas:

1. Función escalón.
2. Función lineal y mixta.
3. Sigmoidal.
4. Función gausiana.

La función escalón o umbral se utiliza únicamente cuando las salidas de la red son binarias. La salida de una neurona se activa sólo cuando el estado de activación es mayor o igual que cierto valor umbral (la función puede estar desplazada sobre los ejes). La función lineal o identidad equivale a no aplicar ninguna función de salida, se utiliza muy poco. Las funciones mixta y sigmoidal son las más apropiadas cuando se quiere como salida información analógica. Con la función sigmoidal, para la mayoría de los valores del estimulo de entrada (variable independiente), el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal (o cualquier otra función similar) es que su derivada es siempre positiva y cercana a cero para valores grandes positivos o negativos; además, toma su valor máximo cuando x es 0. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para las funciones escalón, con la ventaja respecto a esta función, de que la derivada está definida en todo el intervalo. La función escalón no podía definir la derivada en el punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se utilizan derivadas. Se adiciona a la propuesta la ganancia de la función sigmoidal como un parámetro que varia en el intervalo de los números reales 0.5 a 1 [Hilera y Martínez, 1995].

6. ENFOQUE DE ALGORITMOS GENÉTICOS

De manera básica el interés del presente trabajo radica en la necesidad para definir los pesos de conexión y la topología de una red, con tan pocos supuestos como sea posible. Los algoritmos genéticos[6] son un método de búsqueda débil, especialmente establecidos para problemas “dificultosos”; problemas para los cuales no existe un método analítico formal. A primera vista las propiedades de los algoritmos genéticos

[6] Existen numerosas fuentes para entender los algoritmos genéticos, sin embargo se sugiere las siguientes referencias: [Goldberg, 1989] y [Michalwiecz, 1994], como las más importantes.

(AGs) parecen conformar las demandas del problema. Las siguientes propiedades constituyen el comportamiento superior de los AGs cuando el espacio de búsqueda es grande se conoce poco acerca del mismo.

1. Ejecutan búsqueda paralela en el espacio de búsqueda, mientras que el gradiente descendente es un método de búsqueda local
2. Pueden ser utilizados para definir la topología y los pesos de conexión al mismo tiempo, sin ninguna restricción.
3. No necesitan ninguna información del gradiente de la superficie de error, ni necesitan una función de error continua.
4. Proporcionan una buena ventaja entre la búsqueda exhaustiva y aleatoria.

Existe una prueba que los algoritmos genéticos son una buena compensación entre la explotación y la exploración del espacio de búsqueda. La explotación en este contexto se refiere al uso del dominio de conocimiento para guiar la búsqueda, mientras que la exploración consiste en la búsqueda de nuevos estados [Goldberg, 1989].

Se debe tener cuidado que una buena compensación entre exploración y explotación del espacio de búsqueda pueda no ser deseada. Considere por ejemplo, un caso extremo admitido, donde la solución es conocida. Cualquier técnica que se pueda emplear incluso la búsqueda más insignificante, será inferior que aquella que apunta a la solución. Entre este extremo y la compensación ofrecida por los algoritmos genéticos radica una área vasta a la cual se invoca para un análisis cuidadoso. De otra forma este puede ser el caso de los AGs sea ejecutados por un método más analítico.

7. CROMOSOMAS A REDES NEURONALES

El enfoque evolucionario que se plantea consiste en codificar una red neuronal por cromosoma. Existen implementaciones donde tanto la topología de las unidades y sus pesos de conexión forman el cromosoma; algunas otras implementaciones codifican solamente la topología y conducen la definición de los pesos hacia un algoritmo de aprendizaje.

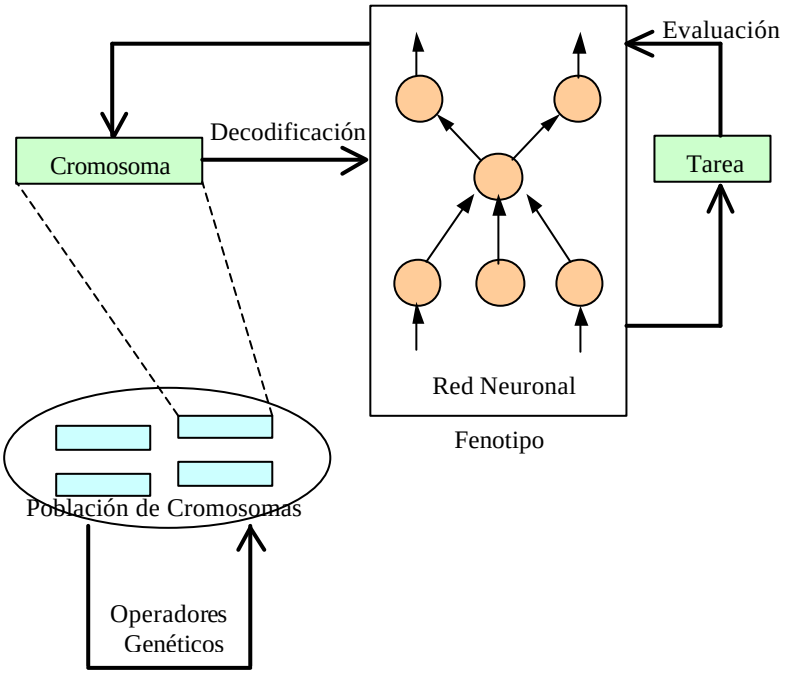

Fig. 3. Evolución Neuronal
Fuente: [Vogiatzis 1995]

Como por ejemplo la propagación de errores hacia atrás. Los operadores genéticos usuales pueden ser aplicados, es decir selección, apareamiento y mutación. Luego los cromosomas son decodificados y evaluados en una tarea especifica. El rendimiento de la red define la adaptabilidad de sus correspondientes cromosomas. La Fig. 3. representa la idea general de la evolución neuronal

8. APLICACION

El comentario en décadas de investigación es que el principio fundamental consiste en ajustar los pesos sobre cada uno de los arcos en el grafo para lograr que los modelos produzcan salidas interesantes. Aunque muchos algoritmos proporcionan métodos convenientes para ajustar los valores de los pesos, ciertos elementos de una arquitectura de red neuronal son difíciles de implementar. En el caso de la propagación hacia atrás estándar esos elementos incluyen el número de capas ocultas, el número de neuronas ocultas por capa, la proporción de aprendizaje, el factor de momento, la función de transferencia, los valores de los pesos iniciales, y los errores permisibles entre la salida observada y deseada del modelo.

De acuerdo al marco teórico revisado, es posible ajustar los valores de los pesos utilizando algoritmos genéticos, este enfoque aunque converge de manera mucho mas lenta que cualquiera de los paradigmas de aprendizaje modernos, puede ser utilizado para implementar una porción razonable de temple simulado.
Una posible aplicación consiste en el desarrollo de un modelo en torno a la construcción de una red neuronal de propagación de errores hacia atrás, encargada de realizar alguna predicción, en la que deben ajustarse principalmente tres factores: el factor de momento, la proporción de aprendizaje y la ganancia de la función sigmoidal.

En un primer experimento sobre la red neuronal, deben escogerse estos factores de manera aleatoria, en un segundo experimento deben ajustarse dichos factores a través de la utilización de un algoritmo genético. Posteriormente puede realizarse un análisis comparativo y una evaluación centrada principalmente en la eficiencia demostrada en el tiempo de entrenamiento de la red neuronal. Una arquitectura general del modelo se presenta de manera detallada en la Fig. 4. En líneas generales este modelo cuenta con tres tipos de parámetros ajustables: el factor de momento, la proporción de aprendizaje y la ganancia de la función sigmoidal.

La arquitectura de la red neuronal elegida puede ser una red de propagación de errores hacia atrás con tres capas ocultas. Los operadores genéticos que se pueden elegir están comprendidos en el conjunto: selección, apareamiento y mutación. Estos operan de acuerdo a una función de adaptabilidad, esta función de adaptación se establece con relación a las características sobresalientes de los tres parámetros ajustables de las redes neuronales.

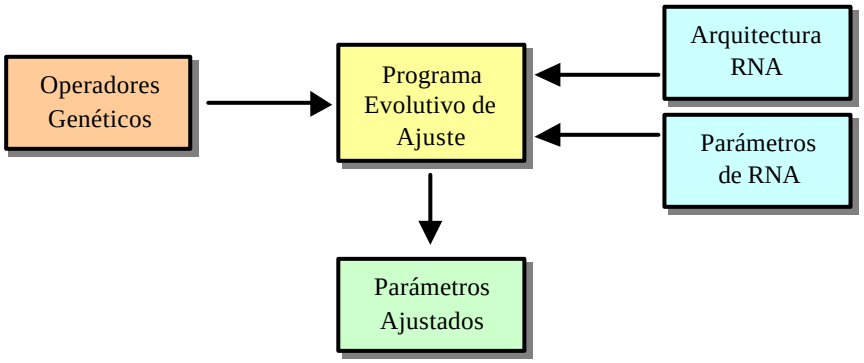

Fig. 4. Arquitectura posible para una aplicación
Fuente: modificado de [Hilera y Martinez, 1995][Goldberg, 1989]

9. CONCLUSIONES

En este trabajo se presentó un posible modelo para afrontar uno de los problemas grandes con los que tropieza el diseño de las redes neuronales: la asignación manual de los pesos iniciales para los procesos de aprendizaje de la red neuronal. En esta propuesta se revisaron los paradigmas de las redes neuronales y los algoritmos genéticos en la perspectiva de lograr que estos últimos puedan colaborar a los primeros en el ajuste adecuado y casi automático de los pesos. Se revisaron también conceptos tan importantes como el aprendizaje en las redes neuronales y la mecánica de la red de propagación de errores. Se amplió el panorama al tratamiento de una red neuronal evolutiva asistida por los operadores de la teoría de los algoritmos genéticos: selección, apareamiento y mutación. Sería recomendable elegir un problema concreto de predicción y aplicar el modelo.

BIBLIOGRAFÍA

1. Angeline, P.J., G.M. Saunders and J. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans on Neural Networks*, January 1994.
2. Brison, A.E. and Y.C. Ho. *Applied Optimal Control*, Blaisdell, New York, 1969.
3. Cybenko, G. Continuous valued Neural networks with two hidden layers are sufficient. *Technical report, Department of Computer Science, Tufts University*, Medford, MA. 1988.
4. Faggin F. VLSI implementation of neural networks. *International Joint Conference on Neural Networks*. Seattle, WA, 1991.
5. Fahlman S.E. and C.Lebiere. The Cascade-Correlation learning architecture. *Technical Report CMU-CS-90-100*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, February 1990.
6. Frean M. The upstart algorithm: A method of constructing and training feedfordward neural networks. *Neural Computation*, pp. 198-209. 1990.
7. Goldberg D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Massachusettes. 1989.
8. Haykin, S. *Neural Networks, A Comprehensive Foundation*, MacMillan College Publishing Company. 1994.
9. Hertz, J. A., A. Krogh and R.G. Palmer. *Introduction to the Theory of Neural Computation*, Volume 1 of Santa Fe Institute Studies in the Sciences of Complexity: Lecture Notes. Addison-Wesley, Redwood City, CA. 1991.
10. Hilera J.R y V.J. Martínez. "Redes Neuronales Artificiales: fundamentos, modelos y aplicaciones". Addison Wesley Iberoamericana. Wilmington, Delaware, EUA. 1995.
11. McCulloch, W. and W. Pitts (1943) "A Logical Calculus of Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, pp. 115-133.
12. Mezard, M. And J.P. Nadal. Learning in feedfordward layered networks: The tiling algorithm. *Journal of Physics*, pp. 2191-2204. 1989.
13. Michalwiecz Z. *Genetic Algorithms + DataStructures = Evolutionary Programs*. Springer-Verlag, 1994.
14. Ramón y Cajal Santiago. *Histología del sistema nervioso del hombre y de los vertebrados*. Consejo Superior de Investigaciones Científicas, Madrid, 1911.
15. Vogiatzis, D. *Symbiotic Evolution of Neural Networks*, Master Thesis, Department of Artificial Intelligence, University of Edinburg. 1995.
16. Wasserman, P.D. *Advanced Methods in Neural Computing*, Van Nostrand Reinhold. 1993.