Diseño de Redes Neuronales Artificiales Mediante Algoritmos Evolutivos

P.A. Castillo, J.G. Castellano, J.J. Merelo y A. Prieto
Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada
Campus de Fuentenueva s/n
E. 18071 Granada (España)
pedro@geneura.ugr.es

Resumen

A pesar de que se han ideado una gran cantidad de algoritmos para entrenar los pesos de una red neuronal a través de la presentación de ejemplos para una topología fija, estos algoritmos tienen el problema de que suelen caer en óptimos locales. La obtención de buenos resultados depende en gran medida de los parámetros de aprendizaje y de los pesos iniciales, así como de la topología de la red.
Los algoritmos evolutivos han demostrado ser una clase de métodos de búsqueda muy efectivos y robustos para localizar zonas del espacio de búsqueda donde hay alta probabilidad de encontrar buenas soluciones, aunque dicho espacio sea grande y contenga múltiples óptimos locales. La aplicación de los algoritmos evolutivos en la optimización de redes neuronales artificiales se justifica por el menor coste computacional, en comparación con los métodos de prueba y error, y la mayor robustez frente a los métodos constructivos/destructivos.
Las técnicas de búsqueda global, tales como los algoritmos evolutivos, analizan amplias zonas del espacio para determinar dónde se encuentran buenas soluciones. En general son menos eficientes que las técnicas de búsqueda local encontrando óptimos locales, por lo que lo más adecuado es dejar que el algoritmo evolutivo seleccione soluciones iniciales en buenas áreas del espacio de búsqueda, para posteriormente localizar los óptimos locales en dichas áreas.
Este trabajo lleva a cabo una revisión de las diferentes formas en que se han combinado algoritmos evolutivos y redes neuronales artificiales para optimizar los diferentes parámetros de diseño de estas últimas, al tiempo que justifica los operadores genéticos específicos utilizados en dichos métodos.

Keywords: Métodos Híbridos, Algoritmos Evolutivos, Redes Neuronales Artificiales, Optimización

1 Introduction

El algoritmo backpropagation (BP) y sus variantes han sido aplicados con éxito en gran cantidad de aplicaciones, dando buenos resultados [Lang et al., 1990] [Fels and Hinton, 1993] [Knerr et al., 1992] [Prechelt, 1994] [Grönroos, 1998]. Sin embargo

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.14 (2001), pp. 2-32. (ISSN: 1137-3601). (c) AEPIA (http://aepia.dsic.upv.es/)

el diseño de las redes a ser entrenadas con dichos algoritmos plantea diversos problemas [Sutton, 1986] [Whitley et al., 1990]. Como sucede con cualquier red neuronal artificial, es necesario establecer de antemano una serie de parámetros. En el caso de perceptrones multicapa (MLP) entrenados usando el algoritmo BP o alguna de sus variantes, hay que establecer la tasa de aprendizaje, la inicialización de los pesos (que influye en la rapidez del aprendizaje y que se alcance o no el mínimo global de la función de error), y el número de capas intermedias y de neuronas en cada una de ellas (decisivo en la capacidad de clasificación/aproximación de la red).

Estos problemas se suelen afrontar utilizando algún método que optimice dichos parámetros que determinan la arquitectura. En general estos métodos de optimización se encuadran en dos grupos: métodos constructivos/destructivos y métodos que hacen evolucionar redes neuronales artificiales (RNA).

Los *métodos constructivos* intentan adaptar el tamaño de la red al problema, comenzando con una red pequeña y añadiendo las capas y unidades necesarias hasta que se encuentre la solución. La principal ventaja es que no se necesita hacer una estimación a priori del tamaño de la red.

Los *métodos de poda* se basan en entrenar una red de mayor tamaño del necesario para ir eliminando posteriormente las partes (unidades o pesos) innecesarios. Una ventaja de estos métodos (en general) es que las redes que consiguen son pequeñas y por lo tanto más fáciles de implementar y rápidas [Reed, 1993] [Reed and Marks, 1999] [Yao, 1999].

El principal problema que presentan tanto los algoritmos constructivos como de poda radica en que, al ser métodos de descenso del gradiente, pueden llegar a una solución que no es la mejor (mínimo local) en la cual quedará estancado el proceso de aprendizaje. Además, los criterios para añadir y eliminar nodos depende de la arquitectura de red y del problema a resolver.

En aquellos *métodos que hacen evolucionar RNA* (métodos híbridos, ya que combinan las metodologías de RNA y de algoritmos evolutivos -AE-), la evolución es un factor fundamental, tal y como lo es el aprendizaje [Balakrishnan and Honavar, 1995] [Yao, 1991] [Yao, 1993b] [Yao, 1993a] [Yao, 1994] [Yao, 1995] [Yao, 1999] [Marín, 1997] [Marín and Sandoval, 1995].

Los AE recorren de forma más o menos aleatoria el espacio de soluciones, centrándose mediante la selección en aquellas zonas en las que el valor de la función de evaluación es máximo (mayores prestaciones de la RNA), mientras que los algoritmos de entrenamiento de RNA clásicos son algoritmos de descenso de gradiente, que iterativamente van haciendo disminuir el error hasta alcanzar un mínimo. Esto hace que, en este último caso, la búsqueda se concentre en una parte pequeña del espacio de soluciones, alrededor de la solución inicial de la que partió el método. En los AE, por el contrario, la búsqueda se lleva a cabo por todo el espacio, siendo posible además que en una población se encuentren soluciones que se sitúen cerca de varios óptimos.

La característica relevante del AE es que puede llevar a cabo una búsqueda global, situando la RNA cerca de un óptimo (global), a partir del cual, mediante el algoritmo de entrenamiento de la red (búsqueda local), se llegue rápida y eficazmente al óptimo [Belew, 1990] [Renders and Flasse, 1996].

El resto de este trabajo está organizado de la siguiente forma: en la sección (2) se lleva a cabo una revisión completa de los diversos enfoques con los que se suelen hacer evolucionar las RNA y de las decisiones de diseño que se deben tomar al realizar el sistema evolutivo. Entre estas decisiones de diseño se comentará el *tipo de codificación* de las redes en los individuos de la población y el método para asignar valor a los *pesos iniciales*. Entre los enfoques para hacer evolucinar RNA que se analizarán figura *la evolución de los pesos de conexión*, *la evolución de la arquitectura* de red, *la evolución de la regla de aprendizaje* y la *evolución del número de entradas*. En la sección 3 se estudian los operadores genéticos más importantes presentes en la bibliografía, analizando las características que aportan al algoritmo híbrido el usarlos. La sección 4 estudia los efectos de los métodos de búsqueda local en los algoritmos híbridos, centrándose en el llamado efecto Baldwin [Baldwin, 1896]

[Hinton and Nowlan, 1987] [Belew, 1989] [Harvey, 1993]. Por último, en la sección 5 se estudian los AE que utilizan cromosomas de longitud variable, centrándonos en aquellos métodos cuyos individuos tienen *segmentos de código genético no utilizados para codificar características* ("intrones").

2 Diseño de RNA con Algoritmos Evolutivos

El proceso de diseño que se va a considerar consta de las siguientes etapas: en primer lugar se debe hacer un preprocesamiento de los datos o patrones que debe aprender la red; en segundo lugar se debe elegir la topología de la red (elegir las entradas y salidas, escoger la codificación, determinar el número de capas ocultas y sus unidades y el tipo de conectividad), y por último, escoger el método para realizar el entrenamiento de la red y sus parámetros.

Se puede hablar de tres enfoques principales para hacer evolucionar una RNA: evolución de los pesos de conexión, de la arquitectura de red, y de las reglas de aprendizaje. La *evolución de los pesos de conexión* supone una aproximación adaptativa y global para realizar el entrenamiento, especialmente en aquellos casos en los que los algoritmos de entrenamiento basados en descenso del gradiente experimentan grandes dificultades. La *evolución de la arquitectura de red* hace que la propia red pueda adaptar su topología al problema sin intervención humana, lo cual representa una aproximación al diseño automático de RNA en tanto que los pesos y la arquitectura pueden hacerse evolucionar. La *evolución de las reglas de aprendizaje* significa una adaptación de las reglas de aprendizaje mediante evolución, en el sentido de buscar los parámetros de la regla de adaptación de pesos e incluso buscar nuevas reglas de adaptación de pesos.

En el siguiente apartado (subsección 2.1) se hace una revisión de los principales métodos utilizados en la bibliografía para codificar los pesos de conexión (codificaciones directa e indirecta) y las representaciones binaria y real de las redes en los individuos de la población. La subsección 2.2 presenta diversos métodos utilizados para inicializar los pesos de la red, ya estén basados en AE o en algún tipo de heurística. Dentro de este apartado, la subsección 2.2.1 analiza aquellos algoritmos híbridos que hacen evolucionar los pesos de conexión buscando un conjunto casi-óptimo de estos para una arquitectura de red fija. La subsección 2.3 presenta aquellos métodos que siguen el enfoque evolutivo para hacer evolucionar la arquitectura de red, buscando una arquitectura adecuada para resolver cierto problema. Ya que la arquitectura de la RNA determina la capacidad de proceso de información de ésta, el diseño de la arquitectura es una de las tareas más importante al utilizar RNA. En la subsección 2.4 se hace una revisión de los distintos métodos que llevan a cabo la evolución de la regla de aprendizaje, buscando los parámetros del algoritmo de aprendizaje o bien la regla de actualización de pesos adecuados para una arquitectura determinada. La subsección 2.5 presenta aquellos métodos híbridos que llevan a cabo la evolución del conjunto de entradas a la red, eliminando las entradas redundantes.

2.1 Elección de la codificación

Tanto en la evolución de los pesos de conexión como de la arquitectura de red, los operadores genéticos son fundamentales en el funcionamiento del AE. No obstante, la forma en que se aplicarán depende estrechamente de la *forma en que se representarán las redes* (**representación binaria o real**) y de la *cantidad de información que cada individuo de la población codificará* (**codificación directa o indirecta**).

2.1.1 Representación binaria vs. real

La evolución de los pesos de conexión conlleva dos fases: la primera consiste en decidir la representación de los pesos en el individuo (*cadenas binarias o números reales*). La segunda tiene que ver con los operadores genéticos de búsqueda que se usarán (ver sección 3). Estas dos decisiones de diseño son muy importantes, ya que el uso de diferentes representaciones y operadores genéticos puede conseguir muy diferentes resultados.

Los algoritmos genéticos [Holland, 1975]

[Goldberg, 1989] suelen utilizar ***cadenas binarias*** para codificar las soluciones candidatas. Las primeras investigaciones en evolución de los pesos de conexión de una RNA [Whitley et al., 1990] [Whitley, 1989b] [Caudell and Dolan, 1989] [Srivinas and Patnaik, 1991] [de Garis, 1991] [Wieland, 1991] [Janson and Frenzel, 1992] [Janson and Frenzel, 1993] utilizan un esquema de representación tal que cada peso se representa mediante una serie de bits, y la RNA completa se representa concatenando todos esos pesos en el individuo. Los pesos de una misma unidad oculta se suelen colocar juntos para que el operador de cruce intercambie unidades ocultas completas, y no sólo pesos individuales [Merelo et al., 1993b] [Merelo and Prieto, 1995] [Merelo et al., 1998].

Las ventajas de la representación binaria son su simplicidad y generalidad, de forma que aplicar los operadores de cruce y de mutación es muy sencillo, sin necesidad de diseñar nuevos operadores específicos. Por otro lado, la implementación física es sencilla ya que los pesos se pueden representar en términos de bits en el soporte físico.

En contrapartida, hay que hacer un balance entre la precisión y la longitud del individuo. Si se utilizan pocos bits para representar cada peso, el entrenamiento podría llegar a fallar debido a que ciertas combinaciones de pesos reales no podrían ser aproximadas por valores discretos. Un detallado estudio sobre este problema se encuentra en [Bernier et al., 2000a] [Bernier et al., 1999] [Bernier et al., 2000c] [Bernier et al., 2000b]. Utilizar muchos bits para aumentar la precisión hace que los individuos sean extremadamente grandes, haciendo la evolución ineficiente (lo cual no tiene excesiva importancia cuando se usan algoritmos híbridos).

Otros investigadores han propuesto una ***representación de reales***, donde cada peso está representado por un número real [Montana and Davis, 1989] [Fogel et al., 1990] [Bartlett and Downs, 1990] [Porto et al., 1995] [Fogel et al., 1997] [Fogel et al., 1995] [Saravanan and Fogel, 1995] [Tang et al., 1995].

Ya que cada individuo estará formado por un vector de números reales, el cruce y la mutación clásicos no se podrán utilizar, por lo que habrá que diseñar nuevos operadores. Montana y Davis [Montana and Davis, 1989] definieron una serie de operadores de búsqueda que construyen detectores de características durante la evolución.

Una manera directa de hacer evolución de vectores de números reales es utilizar programación evolutiva (PE) o estrategias evolutivas (EE), que están pensadas para hacer optimización continua. Hay numerosos trabajos en los que se utilizan estas aproximaciones evolutivas para hacer evolución de RNA [Fogel et al., 1990] [Porto et al., 1995] [Greenwood, 1997] [Topchy and Lebedko, 1997] [Saravanan and Fogel, 1995] [Sarkar and Yegnanarayana, 1997], en los que el operador de búsqueda principal es la mutación (gaussiana o de Cauchy [Yao and Liu, 1996d] [Yao et al., 1997]).

A pesar de que la representación real es más precisa, un inconveniente que presenta (derivado de ese incremento en la precisión) es que el espacio de búsqueda es extremadamente más grande [Michalewicz, 1992] [Michalewicz, 1996].

Ambos tipos de representación presentan el problema de que la aplicación entre entradas y salidas puede ser implementada por diversas redes, simplemente permutando alguna de las neuronas de las capas ocultas [Radcliffe, 1991b] [Hancock, 1992] [Thierens et al., 1993] (*"problema de las permutaciones"* o *"competing conventions problem"*, ver sección 3.6), es decir, se trata de un problema con una alta degeneración.

El problema se presenta en aquellos métodos híbridos en los que el AE trabaja sobre la representación genotípica de la red, ya que redes estructuralmente diferentes son soluciones candidatas diferentes, a pesar de que su funcionamiento sea idéntico. De esta forma, si el cruce se aplica a dos redes funcionalmente idénticas, pero estructuralmente diferentes, se obtendrán descendientes no útiles.

Para evitar dicho problema, algunos trabajos recientes presentan métodos híbridos en los

cuales se hace evolución de RNA ***sin necesidad de codificar*** la red como un individuo de la población. En dichos trabajos, el AE está implementado de tal forma que no necesita hacer uso de una codificación de la red, sino que, la propia red es un individuo de la población que se hace evolucionar a través de ciertos operadores genéticos específicos. Un ejemplo de estos métodos es el propuesto por Castillo et al. [Castillo et al., 1999c], en el que se utiliza un método híbrido que combina enfriamiento simulado y MLP para buscar los pesos iniciales y el parámetro de aprendizaje de BP del MLP.

Así mismo, Rivas et al. [Rivas et al., 1999] proponen un método con el que se optimizan los pesos y centros de una red de base radial (RBF), utilizando un AE que hace uso de operadores genéticos específicos para realizar evolución de las redes, sin necesidad de codificación alguna.

Por último, el método G-Prop [Castillo et al., 1999a] [Castillo et al., 1999b] [Castillo et al., 2000a] [Castillo et al., 2000b] [Castillo et al., 2000c] [Castillo et al., 2001a] [Castillo et al., 2001b] [Castillo et al., 2001c] [Valderrábano et al., 2001], que lleva a cabo la optimización de MLP mediante un AE, busca los parámetros de aprendizaje para el algoritmo de entrenamiento, los pesos iniciales de la red y la arquitectura adecuada para alcanzar una buena solución. La evolución se lleva a cabo a través de seis operadores genéticos específicos (mutación, cruce, añadir, eliminar o sustituir neuronas ocultas, y un operador de entrenamiento) que acceden al MLP tratando cada neurona intermedia y todos sus pesos como un gen, de forma que cuando por ejemplo se cruzan dos MLP, realmente se intercambian neuronas de las capas ocultas completas (la neurona intermedia y sus pesos de entrada y salida se tratan conjuntamente), tal y como se propone en [Thierens et al., 1993].

2.1.2 Codificación directa vs. indirecta

La evolución de la arquitectura de red conlleva dos fases: la primera consiste en decidir la codificación de la red en el genotipo del individuo. La segunda tiene que ver con el AE utilizado (los operadores genéticos de búsqueda que se usarán, ver sección 3). La clave al decidir la codificación de una arquitectura está en la cantidad de información que codificará cada individuo de la población. Se puede optar por el máximo detalle, si cada conexión y nodo de la arquitectura se codifica (*codificación directa*), o bien por codificar sólo los parámetros más importantes de la arquitectura, tales como el número de nodos en cada capa (*codificación indirecta*).

En la ***codificación directa*** cada conexión está especificada mediante su representación binaria [Whitley et al., 1990] [Marín and Sandoval, 1993] [Alba et al., 1993] [Schiffmann et al., 1992] [Miller et al., 1989] [Schaffer et al., 1990] [Grönroos, 1998] [Roberts and Turega, 1995]. En general, para representar una red con N neuronas, se forma una matriz de $N \times N$ donde cada elemento c_{ij} indica la presencia o ausencia de conexión entre el nodo i y el j. Esta aproximación se puede ampliar de forma que cada c_{ij} sea un número real que represente la conexión entre el nodo i y el j y así se haga evolucionar la arquitectura y los pesos de conexión simultáneamente [Srivinas and Patnaik, 1991] [Koza and Rice, 1991] [Martí, 1992] [Marín and Sandoval, 1993] [Alba et al., 1993] [White and Ligomenides, 1993] [Grönroos, 1998] [Roberts and Turega, 1995].

Cada matriz tiene una correspondencia única con la arquitectura de red correspondiente. La cadena binaria que representa una arquitectura es la concatenación de las filas (o columnas) de la matriz. Una ventaja de esta codificación es que podemos imponer restricciones a las arquitecturas a explorar, simplemente imponiendo ciertas restricciones a la matriz. Por ejemplo, para codificar redes *hacia adelante* (feed-forward) simplemente debemos hacer que la matriz sea triangular inferior.

La principal ventaja de este tipo de codificación es su facilidad de implementación y manejo por los operadores genéticos, de forma que una conexión puede ser añadida o eliminada muy fácilmente.

Un problema que presenta es la falta de escalabilidad. Una RNA grande necesita una matriz muy grande, lo cual hace que se incremente el tiempo de computación.

Para reducir la longitud de la repre-

sentación de la arquitectura, muchos autores han utilizado la ***codificación indirecta*** [Harp et al., 1990] [Harp et al., 1989] [Grönroos, 1998] [Roberts and Turega, 1995], en la cual sólo algunas de las características de la arquitectura se codifican en el individuo. La manera en que cada conexión se codifica está predefinida de acuerdo a cierto conocimiento previo sobre el dominio del problema, o bien mediante ciertas reglas determinísticas. La codificación indirecta puede generar representaciones más compactas (en cuanto a la longitud del genotipo) de la arquitectura de red, aunque ello no quiere indicar que pueda encontrar redes más compactas.

En la bibliografía podemos encontrar diversos métodos de codificación indirecta, aunque la mayoría se basa en realizar una ***representación paramétrica***. Las arquitecturas de red se pueden especificar con ciertos parámetros, como el número de capas ocultas, el número de nodos por capa y las conexiones entre éstas. Harp et al. [Harp et al., 1990] [Harp et al., 1989] propusieron una aproximación consistente en representar las capas y las conexiones salientes, de forma que en el individuo, para cada capa se guarda información sobre el número de nodos en esa capa y el tipo de conectividad con la siguiente (no la especificación de todas las conexiones). Para utilizar esta aproximación hay que desarrollar operadores genéticos específicos, ya que individuos funcionalmente correctos pueden generar descendientes no funcionales.

Este método de codificación puede reducir la longitud del individuo, aunque presenta la desventaja de que el AE sólo puede buscar en un subconjunto limitado del espacio de la arquitectura. Es un buen método a utilizar cuando conocemos el tipo de arquitectura que estamos buscando.

Cuando se desconoce el tipo de arquitectura buscada, se suele recurrir a un tipo de codificación indirecta basada en ***reglas de construcción*** [Kitano, 1990b] [Yao and Shi, 1995] [Vonk et al., 1995] [Gruau, 1992b] [Gruau, 1992a] [Gruau and Whitley, 1993a] [Grönroos, 1998] [Mjolsness et al., 1989]. En esta aproximación, en lugar de codificar parámetros de la arquitectura, lo que se hace es codificar en el individuo las reglas de construcción que se usarán para formar arquitecturas. Una regla de construcción suele ser una ecuación recursiva [Mjolsness et al., 1989] o una regla de generación [Kitano, 1990b] similar a las reglas de producción usadas en los sistemas de producción. La generación de la matriz que especifica la conectividad se efectúa partiendo de un símbolo base, y aplicando repetidamente las distintas reglas de generación a los elementos no terminales de la matriz hasta que sólo contenga símbolos terminales (un 1 significará la existencia de conexión y un 0 la ausencia).

El método propuesto por Gruau [Gruau, 1992a] comienza con una neurona y hace crecer la red, pudiendo expresar redes recursivas. Según el autor, dicho método genera redes muy compactas y escalables. La ventaja de este método es que se consigue una representación más compacta, sin embargo, requiere que se le especifique el número de pasos de aplicación de reglas y no permite el uso de reglas recursivas. Desde nuestro punto de vista, el principal interés es académico, porque las redes generadas no se pueden entrenar mediante algoritmos clásicos.

2.2 Elección de los pesos iniciales

La inicialización de los pesos de la red es clave para obtener una convergencia rápida en MLP, ya que, dependiendo del punto del espacio de búsqueda del que se parta (y ese punto está determinado por el conjunto de pesos) se obtendrán mejores o peores soluciones al llevar a cabo el entrenamiento de la red [Thimm and Fiesler, 1995].

En la bibliografía se pueden encontrar diversos métodos de inicialización de pesos [Thimm and Fiesler, 1995]. El más simple de todos se basa en hacer una inicialización aleatoria, que resulta sencillo y generalmente produce múltiples soluciones [Kolen and Pollack, 1990]. Otros métodos más elaborados requieren un análisis estadístico de los datos de entrenamiento, lo cual los hace poco eficientes.

Fahlman [Fahlman, 1988] propuso, tras un estudio experimental, usar un rango inicial que variaba entre $[-4.0, 4.0]$ y $[-0.5, 0.5]$ dependiendo del problema. Otros autores [Bottou, 1988] [Boers and H.Kuiper, 1992]

[Smieja, 1991] [Wessels and Barnard, 1992] [Nguyen and Widrow, 1990] propusieron rangos de inicialización particulares que dan buenos resultados en ciertos problemas, aunque sin justificación matemática y sin llevar a cabo una comparación de resultados con otros métodos.

Lee et al. [Lee et al., 1993] demostraron teóricamente que los nodos con pesos con valores altos son más propensos a sufrir *saturación* (los cambios que se realicen a dichos pesos no afectarán practicamente a la salida del nodo). En dicho trabajo se propone utilizar un rango pequeño de inicialización, que puede hacer que la velocidad de aprendizaje aumente.

Otros autores proponen métodos no aleatorios de inicialización [Chen and Nutter, 1991] [Denoeux and Lengellé, 1993] [Kim and Choi, 1997]. Estos métodos necesitan hacer un preprocesamiento, de los patrones de entrada o de todos los pesos de la red, previo a la aplicación del algoritmo de aprendizaje, lo cual incrementa el coste computacional. En [Kim and Choi, 1997] se propone un método alternativo a la inicialización aleatoria, en el que se inicializan los pesos de sesgo ("bias") de forma que se minimice la función objetivo, mientras que el resto de los pesos se inicializan aleatoriamente. Lo que se pretende es eliminar el periodo inicial de decrecimiento de pesos en el entrenamiento, fenómeno que hace que el MLP produzca malos resultados.

2.2.1 Evolución de los pesos de conexión

El entrenamiento de una RNA se suele formular como la minimización de una función de error entre las salidas esperadas y las que ofrece la red, al tiempo que se van ajustando dichos pesos. BP y sus variantes se han aplicado con éxito en varias áreas [Lang et al., 1990] [Fels and Hinton, 1993] [Knerr et al., 1992], aunque presenta ciertos problemas debido a que se basan en descenso del gradiente [Sutton, 1986] [Whitley et al., 1990].

Una forma de resolver los problemas que presentan estos métodos es hacer evolucionar directamente los pesos de conexión. Así, se puede usar un AE para buscar un conjunto de pesos casi óptimo, de forma global, sin utilizar información sobre el gradiente [Whitley, 1989a] [Montana and Davis, 1989] [Fogel et al., 1990] [Belew et al., 1991] [Koza and Rice, 1991] [Porto et al., 1995] [Topchy et al., 1996] [Topchy and Lebedko, 1997] [Kinnebrock, 1994] [Osmera, 1995] [Yoon et al., 1994] [Fogel et al., 1995] [Koeppen et al., 1997] [Merelo et al., 1993a] [de Falco et al., 1998b] [Whitley et al., 1990]. De esta forma se evita tener que ajustar los parámetros del algoritmo de aprendizaje (tasa de aprendizaje, momento, etc.) y, además, no se ha de imponer ninguna restricción a la topología de la red (función de activación continua y derivable) ya que en la búsqueda no se necesita información del gradiente.

La función de evaluación debe definirse teniendo en cuenta dos factores principales: el error entre las salidas deseadas y las obtenidas, y la complejidad de la red.

La evolución de los pesos de conexión conlleva dos fases: la primera consiste en decidir la representación de los pesos, esto es, usar, por ejemplo cadenas binarias (ver subsección 2.1). La segunda tiene que ver con los operadores genéticos de búsqueda que se usarán (ver sección 3). Estas dos decisiones de diseño son muy importantes, ya que el uso de diferentes representaciones y operadores genéticos puede conseguir muy diferentes resultados.

El uso de un AE puede extenderse al entrenamiento de diferentes tipos de RNA, independientemente de que sean *hacia adelante*, recurrentes o de otro tipo. La aproximación evolutiva puede ahorrar mucho esfuerzo en el desarrollo de diferentes algoritmos de entrenamiento para diferentes tipos de RNA.

Al mismo tiempo, el uso del enfoque evolutivo puede facilitar la implementación y el uso de ciertas características en las redes. Por ejemplo, se puede incrementar la capacidad de generalización o decrementar la complejidad de la red diseñando adecuadamente la función de evaluación (incorporar la "desintegración de pesos").

El entrenamiento de MLP usando AE puede resultar lento si se compara con algunas variantes del algoritmo BP. Sin embargo, los AE son mucho menos sensibles a las condiciones iniciales del entrenamiento, y además buscan una solución global y óptima, mientras que un algorit-

mo basado en descenso del gradiente sólo podrá encontrar un óptimo local en la vecindad del punto desde el que inició su búsqueda.

En la bibliografía podemos encontrar trabajos en los que, para ciertos problemas, la aproximación evolutiva resulta ser más rápida que BP [Bartlett and Downs, 1990] [Prados, 1992b] [Prados, 1992a] [Porto et al., 1995] [Hansen and Meservy, 1996] [Sexton et al., 1998]. Otros autores, por el contrario, presentan resultados en los que alguna variación de BP llega a ser más ventajosa para resolver cierto tipo de problema [Kitano, 1990b]. Estos resultados contradictorios pueden deberse a que se estén comparando diferentes AE y diferentes versiones de BP. Es evidente que no hay un algoritmo de entrenamiento que resulte mejor que el resto. El mejor siempre depende del problema a resolver, lo cual está en consonancia con el teorema *no free lunch* ("no existe comida gratis") [Wolpert and Macready, 1997].

2.2.2 Algoritmos híbridos

En general, los AE son ineficientes en la tarea de realizar una búsqueda fina local, mientras que son muy eficientes para realizar búsquedas globales. Por ello, el entrenamiento mediante AE puede mejorarse si se incorpora un método de búsqueda local al de búsqueda global proporcionado por el AE. Así, éste buscará una región adecuada en el espacio de búsqueda y posteriormente el método de búsqueda local afinará la solución encontrada por el AE para encontrar una solución más cercana al óptimo en dicha región.

En la bibliografía podemos encontrar multitud de trabajos [Lee, 1996] [Belew et al., 1991] [Omatu and Deris, 1996] [Omatu and Yoshioka, 1997] [Erkmen and Ozdogan, 1997] en los que se proponen métodos híbridos para buscar el conjunto inicial de pesos más cercano al óptimo, y después usar BP para llevar a cabo la búsqueda local a partir de dichos pesos.

Si tenemos en cuenta que BP debe utilizarse varias veces para encontrar unos pesos iniciales adecuados, debido a su dependencia respecto a las condiciones iniciales, este tipo de métodos híbridos puede resultar muy adecuado.

2.3 Evolución de la arquitectura de red

La arquitectura de red incluye la *topología* de la red, la *conectividad* y la *función de transferencia* de cada neurona de la red.

Como ya se comentó, el diseño de la topología de la RNA es de suma importancia en cuanto a que marcará su capacidad de aprendizaje y generalización. Una red con muy pocas neuronas puede tener dificultad en aprender la tarea requerida, mientras que una red con demasiadas neuronas puede ajustar excesivamente su salida a los patrones de entrenamiento, disminuyendo su capacidad de generalización ante nuevos patrones.

Hasta ahora, el diseño de la arquitectura se había realizado manualmente, por un experto con la suficiente experiencia, mediante un proceso de prueba y error. Las aproximaciones automáticas más estudiadas han sido los algoritmos incrementales y de poda [Gallant, 1993] [Gallant, 1990] [Frean, 1990] [Lee et al., 1990] [Fahlman and Lebière, 1990] [Mézard and Nadal, 1989] [Cun et al., 1990] [Hassibi et al., 1994] [Buntine and Weigend, 1994].

Un algoritmo incremental comienza con una red pequeña y va añadiendo capas, neuronas o conexiones conforme son necesarias durante el entrenamiento, mientras que un algoritmo de poda hace lo contrario, esto es, comienza con una red de gran tamaño y va eliminando elementos durante el entrenamiento. Estos métodos siguen adoleciendo del problema principal de los métodos de escalada: son susceptibles de caer en mínimos locales, además de que sólo pueden llevar a cabo la búsqueda en subconjuntos reducidos del espacio de búsqueda de las posibles topologías.

El diseño óptimo de la arquitectura de la red puede formularse como un problema de búsqueda en el espacio de las arquitecturas, donde cada punto representa una posible arquitectura de red. Dicho problema de optimización puede ser resuelto, debido a las siguientes características apuntadas por Miller et

al. [Miller et al., 1989], con más facilidad por un AE que por los métodos incrementales o de poda ya referenciados:

- El espacio de búsqueda es infinitamente grande, ya que el número de nodos y conexiones no está fijado de antemano.
- La superficie no es diferenciable, ya que los cambios en el número de nodos o conexiones son discretos.
- La superficie es compleja y ruidosa, ya que la aplicación entre la arquitectura y su capacidad de representación es indirecta y depende del método de evaluación usado.
- Arquitecturas similares pueden tener una capacidad diferente.
- La superficie es multimodal, ya que diferentes arquitecturas pueden tener una capacidad parecida.

Al igual que en el enfoque de la evolución de los pesos de conexión, en este otro hay dos fases fundamentales, que son la representación de la red en el genotipo del individuo (codificación directa o indirecta, ver subsección 2.1), y el AE utilizado (los operadores genéticos de búsqueda que se usarán, ver sección 3).

En la bibliografía podemos encontrar diversas investigaciones acerca del enfoque evolutivo de la arquitectura de red [Koza and Rice, 1991] [Miller et al., 1989] [Erkmen and Ozdogan, 1997] [White and Ligomenides, 1993] [Alba et al., 1993] [Kitano, 1990a] [Yao and Shi, 1995] [Liu and Yao, 1996b] [Yao and Liu, 1996b] [Yao and Liu, 1996a] [Liu and Yao, 1996a] [Yao and Liu, 1996c] [Yao and Liu, 1997b] [Yao and Liu, 1997a] [Yao and Liu, 1998b] [Yao and Liu, 1998a] [Ribert et al., 1994] [de Falco et al., 1998a] [Ragg et al., 1995].

Casi todas las investigaciones se han centrado en el aspecto de hacer evolucionar la topología de red, dejando fija la función de transferencia de cada nodo, que está predefinida [Ragg et al., 1995]. Diversos trabajos demuestran la importancia de dichas funciones en la capacidad de las RNA [Mani, 1990] [Lovell and Tsoi, 1992] [DasGupta and Schnitger, 1992], lo cual nos lleva a plantear una optimización de los parámetros que la definen, o cuando menos, el poder definir, dinámicamente, qué función utilizar en cada uno de los nodos de la red.

En una primera aproximación, Stork et al. [Stork et al., 1990] aplicaron un AE para hacer evolucionar la estructura topológica y la función de transferencia de RNA simples. Tanto la topología como la función de transferencia estaban codificadas en el genotipo de cada individuo de la población del AE.

White y Ligomenides [White and Ligomenides, 1993] propusieron una aproximación más simple que la anterior. En ésta, cada RNA en la población tenía el 80% de sus pesos con funciones sigmoidales, y el 20% restante, funciones gaussianas. La investigación se centraba en decidir qué mezcla era la óptima, de hecho, ninguno de los parámetros de dichas funciones se hacía evolucionar.

Liu y Yao [Liu and Yao, 1996a] aplicaron programación evolutiva (PE) [Fogel et al., 1991] para hacer evolucionar RNA con una mezcla de funciones gaussianas y sigmoidales de forma que el porcentaje de cada una no estaba prefijado. El algoritmo añadía o eliminaba unidades, eligiendo el tipo de función a usar en dicha unidad de forma aleatoria.

Las mejoras presentadas en estos trabajos debidas a la evolución de la función de transferencia no son significativas.

2.3.1 Evolución simultánea de la arquitectura de red y de los pesos de conexión

El enfoque estudiado hasta ahora se limita a buscar una arquitectura adecuada, sin preocuparse de los pesos de conexión, tras lo cual hay que realizar el entrenamiento de dichos pesos.

Esto presenta el problema de que diferentes conjuntos de pesos, para una arquitectura dada, pueden producir diferentes resultados; y que algoritmos de entrenamiento diferentes pueden producir diferentes resultados a partir del mismo conjunto de pesos inicial.

Una forma de evitar este problema es hacer evolucionar tanto los pesos como la arquitectura simultáneamente [Koza and Rice, 1991] [White and Ligomenides, 1993] [Gruau, 1992b] [Yao and Shi, 1995] [Liu and Yao, 1996b] [Yao and Liu, 1996b] [Yao and Liu, 1996a] [Liu and Yao, 1996a] [Yao and Liu, 1997b] [Yao and Liu, 1997a] [Yao and Liu, 1998b] [Yao and Liu, 1998a]. De esta forma, cada individuo de la población especifica tanto la arquitectura como los pesos de conexión.

Yao y Liu [Yao and Liu, 1997b] [Yao and Liu, 1998a] presentaron un método basado en PE para hacer evolucionar tanto la arquitectura como los pesos de conexión. Dicho método hace uso intensivo de varios operadores de mutación para modificar los individuos. Dichas mutaciones están orientadas, más que a hacer una evolución genética, a una evolución funcional. No se hace uso del operador de crossover debido a que el hecho de combinar una parte de una RNA con otra parte de otra RNA posiblemente hará que ambas redes pierdan funcionalidad.

2.4 Evolución de la regla de aprendizaje

Un algoritmo de entrenamiento de RNA puede tener diferente funcionamiento al ser aplicado a diferentes arquitecturas. El diseño de algoritmos de entrenamiento, o reglas de aprendizaje para ajustar los pesos depende del tipo de arquitectura que se vaya a utilizar. Debido al escaso conocimiento acerca de la arquitectura de red que se suele tener, es preferible desarrollar un sistema automático para adaptar la regla de aprendizaje a la arquitectura y el problema a resolver. Es más, el sistema debería hacer que la propia red adaptara su regla de aprendizaje al problema, lo cual nos lleva a estudiar las aproximaciones que hacen evolucionar la regla de aprendizaje.

Se han propuesto varios modelos [Hinton and Nowlan, 1987] [Belew, 1990] [Nolfi et al., 1990] [Muhlenbein, 1988] [Muhlenbein and Kindermann, 1989] [Paredis, 1991] [Chalmers, 1990] [Bengio and Bengio, 1990] [Bengio et al., 1992] [Fontanari and Meir, 1991] [Ackley and Littman, 1991] [Baxter, 1992] [Crosher, 1993] [Turney et al., 1996], aunque la mayoría de ellos se centran en cómo el aprendizaje puede modificar o guiar la evolución [Hinton and Nowlan, 1987] [Belew, 1990] [Nolfi et al., 1990], y en la relación entre la evolución de la arquitectura y de los pesos de conexión [Muhlenbein, 1988] [Muhlenbein and Kindermann, 1989] [Paredis, 1991]. Pocos son los trabajos que se centran en la evolución de las reglas de aprendizaje [Chalmers, 1990] [Bengio and Bengio, 1990] [Bengio et al., 1992] [Fontanari and Meir, 1991] [Baxter, 1992] [Crosher, 1993] [Ribert et al., 1994].

El enfoque que nos ocupa ha sido utilizado de diferentes formas, la primera de las cuales se basa en el ***ajuste de los parámetros del algoritmo BP*** (tasa de aprendizaje y el momento) [Harp et al., 1989] [Belew et al., 1991] [Kim et al., 1996].

Algunos investigadores [Belew et al., 1991] [Patel, 1996] [Kim et al., 1996] proponen métodos en los que se usa un proceso evolutivo para encontrar los parámetros del BP mientras dejan la arquitectura predefinida.

Otros [Harp et al., 1989] [Merelo et al., 1993a] proponen codificar los parámetros del algoritmo BP en los individuos de la población, junto con la arquitectura de red. Dichos métodos posibilitan el llegar a encontrar una combinación casi óptima de los parámetros del BP y de la arquitectura de red para un problema concreto.

Los trabajos comentados hasta ahora se limitan a aplicar una estrategia evolutiva para buscar los parámetros del algoritmo de aprendizaje, sin embargo la regla de actualización de pesos sigue estando predefinida y es fija.

Al contrario que en la evolución de los pesos de conexión y la arquitectura, que trata con objetos estáticos (pesos y topología), ***la evolución de la regla de aprendizaje*** está orientada a proporcionar un comportamiento dinámico a la RNA.

Debido a la complejidad que supone llevar a cabo una representación genérica para todas las posibles reglas de aprendizaje, hay que establecer ciertas restricciones que simplifiquen dicha representación, y por tanto el espacio de

búsqueda. Así pues, habrá que tener en cuenta que la regla de adaptación de pesos dependerá sólo de información local, tal como las activaciones de los nodos de entrada o los pesos actuales, y que la regla de aprendizaje será la misma para todas las conexiones de la RNA.

Chalmers [Chalmers, 1990] definió una regla de aprendizaje como una combinación lineal de cuatro variables y seis términos producto. Cada individuo de la población es una cadena binaria que codifica exponencialmente diez coeficientes más un término de escala. En sus experimentos demostró el potencial de esta aproximación para descubrir nuevas reglas a partir de un conjunto de reglas generadas aleatoriamente.

Otros autores [Bengio and Bengio, 1990] [Bengio et al., 1992] [Fontanari and Meir, 1991] [Baxter, 1992] [Crosher, 1993] han desarrollado métodos basados en la aproximación de Chalmers o han hecho una extensión de ésta, llegando a resultados similares, en los que se aprecia cómo la capacidad de aprendizaje de una RNA puede mejorarse mediante evolución.

2.5 Evolución del conjunto de entradas a la red

En muchos problemas, el conjunto de entradas a una RNA puede llegar a ser muy grande. Sin embargo, en ese conjunto suele haber muchas entradas redundantes que lo único que hacen es incrementar el tamaño de la red y así, aumentar el tiempo necesario para entrenarla y conseguir una buena solución.

Encontrar el conjunto óptimo de entradas a una RNA puede formularse como un problema de búsqueda, en el cual tenemos un conjunto de potenciales entradas y queremos encontrar un subconjunto de éste que tenga el menor número de entradas posible a la vez que la red no produzca resultados peores que los que da la red que utiliza todo el conjunto de potenciales entradas.

La selección de las variables ha sido abordado usando otros métodos tales como el *SOM* de Kohonen [Kohonen, 1982a] [Kohonen, 1982b] [Kohonen, 1990] [Kohonen, 1997] y *escalado multidimensional* ("multidimensional scaling") [Cox and Cox, 1994].

Por otro lado, numerosos autores han abordado este problema mediante un enfoque evolutivo [Guo and Uhrig, 1992] [Brill et al., 1992] [Hsu and Wu, 1992] [Hornyak and Monostori, 1997] [Dellaert and Vandewalle, 1994] [Weller et al., 1995], ofreciendo buenos resultados.

En la evolución de las entradas, cada individuo de la población representa un subconjunto de las posibles entradas. Esto se suele representar como una cadena binaria de la misma longitud que el número total de entradas en la cual un 1 representa la presencia de cierta entrada, mientras que un 0 representa la ausencia. La evaluación se lleva a cabo entrenando una RNA cuya arquitectura suele estar fijada de antemano con esas entradas.

Guo y Uhrig [Guo and Uhrig, 1992] presentan un algoritmo genético (AG) cuyo propósito es reducir la dimensionalidad del espacio de búsqueda seleccionando las variables importantes que se usarán como entrada a RNA modulares para detección de fallos en plantas generadoras de energía. El AG utiliza cadenas binarias para representar el conjunto total de entradas y seleccionar aquellas que formen el subconjunto óptimo para ese problema. Dellaert y Vandewalle [Dellaert and Vandewalle, 1994] proponen un método de detección de características de entrada para entrenar redes que generalicen correctamente en el reconocimiento de caracteres escritos. Dicho método se basa en un AG que codifica las entradas de la red (los pixels de la imagen) en una cadena de forma que busca el subconjunto de entradas óptimo. Hornyak y Monostori [Hornyak and Monostori, 1997] aplican con éxito un AG para obtener un subconjunto de entradas del conjunto de entrenamiento para entrenar RNA para predicción financiera.

3 Operadores Genéticos Específicos

Una cuestión clave al llevar a cabo la evolución de RNA es la elección de los operadores genéticos de búsqueda que se usarán en el AE,

ya que la precisión de la búsqueda dependerá de estos.

El papel de los operadores genéticos de búsqueda es generar nuevas soluciones candidatas en el área del espacio de búsqueda en la que se encuentra la solución a partir de la cual se van a generar las nuevas. En ciertos casos es preferible generar dichas soluciones en áreas lejanas a la inicial.

Los operadores más comunes en los métodos encontrados en la bibliografía son el de mutación, el de cruce y algunos operadores que realizan una búsqueda local (BP y sus variantes).

3.1 Mutación

El operador de mutación puede tener dos papeles principales: el de llevar a cabo un refinamiento de las soluciones (un cambio o movimiento pequeño en el espacio de búsqueda) o bien, en el caso de que exista otro operador que las refine (uno que realice una búsqueda local), cambiar de área en el espacio de búsqueda (un cambio o movimiento grande) [Land, 1998]. Es más, realizar pequeñas mutaciones no tiene sentido si ese individuo se refinará posteriormente, ya que los cambios causados por la mutación se verán deshechos por la búsqueda local del operador de refinamiento (dicho operador llevará el individuo de vuelta al óptimo local). En casos así, una mutación que provoque grandes cambios (para mover individuos entre áreas) será más útil.

Algunos autores [Land, 1998] llegan a la conclusión de que en ciertos problemas, para evitar caer y quedar atrapado en óptimos locales, el método debe utilizar grandes mutaciones que lo saque de dichas áreas. Estas grandes mutaciones alejan la población de los óptimos locales, lo cual hace que el valor de la función de evaluación media se vea degradado en las primeras generaciones del AE. En generaciones más tardías, la población se encontrará alrededor del óptimo global, lo cual compensa los efectos negativos iniciales de dichas mutaciones. A partir de ahí, utilizando un método de refinamiento (operador de mutación que realice pequeños cambios o un operador de búsqueda local), se puede alcanzar el óptimo global [Land, 1998].

Montana y Davis [Montana and Davis, 1989] concluyeron que un operador de mutación que actúa simultáneamente sobre todos los pesos de entrada de un nodo es más efectivo que otro que actúe sobre los pesos individuales. Utrecht y Trint [Utrecht and Trint, 1994] propusieron varias heurísticas para utilizar un operador de mutación que generase pequeños cambios en las soluciones. En relación con la decisión de hacer la mutación más o menos drástica, Angeline et al. [Angeline et al., 1994] propuso un operador que hiciese los cambios sobre los pesos según lo cerca de la solución se hayase la red. Schiffmann [Schiffmann et al., 1990] restringe los cambios introducidos por su operador a añadir o eliminar conexiones entre nodos.

Rivas et al. [Rivas et al., 1999] proponen dos operadores de mutación para hacer evolución de RBF: un operador mutador de centros, que modifica el valor de cada componente del punto del espacio en que se centra la funcion de base radial de cada neurona de la capa oculta de una red neuronal, sumando o sustrayendo una pequeña cantidad que sigue una función de probabilidad de una gaussiana; un operador mutador de radios, que modifica el valor de cada componente del radio utilizado en la función de base radial de cada neurona de la capa oculta de una red neuronal, según una función de probabilidad definida por una gaussiana.

Castillo et al. [Castillo et al., 1999a] [Castillo et al., 1999b] [Castillo et al., 2000b] [Castillo et al., 2000c] presentan un operador de mutación basado en las ideas de Montana y Davis [Montana and Davis, 1989] y en el algoritmo presentado por Kinnebrock en [Kinnebrock, 1994], que modifica los pesos del MLP después de cada época de entrenamiento de la red sumándole o restándole un pequeño número aleatorio.

3.2 Cruce

El papel general de un operador de cruce es el recombinar aquellas partes útiles de los distintos individuos para formar nuevas soluciones que tengan las mejores características de ambos padres. La habilidad del operador para hacer esto depende del problema en cuestión y de la forma en que las soluciones se están representando. Una representación binaria de los indi-

viduos de la población hace que un operador genérico de cruce sea adecuado para muchos problemas.

El operador estándar de cruce (un punto de cruce, dos puntos de cruce, o cruce uniforme) simplemente asigna cada bit en el hijo al valor de dicho bit en el padre correspondiente.

Según otros autores, el uso de operadores de cruce puede tener efectos negativos en la construcción de una RNA, ya que dicho operador funciona bien cuando se pueden intercambiar "bloques de construcción", y en RNA no está claro qué se puede definir como un bloque de construcción debido a la naturaleza distribuida de la representación de la información [Rumelhart and McClelland, 1986]. En los MLP la información se encuentra distribuida entre todos los pesos de la red, por lo cual, combinar partes de dos redes puede hacer que las características de ambas se vean degradadas.

Incluso en aquellos casos en que el operador de cruce sea de utilidad, si los miembros de la población son muy parecidos (la población ha convergido en un área concreta), el cruce no tiene efectos destacables ya que los elementos generados serán casi idénticos a los padres [Land, 1998].

En otro tipo de redes, como las RBF, en las cuales la información no se halla distribuida entre todos los pesos de la red, el operador de cruce resulta ser un operador muy útil [Sarkar and Yegnanarayana, 1997] [Angeline, 1997]. Por otro lado, aquellas redes que distribuyen la información entre sus pesos suelen ser más compactas y tienen más capacidad de generalización.

Trabajando con MLP, algunos autores consideran como gen (unidad intercambiable por el cruce) todos los pesos de entrada a un nodo [Montana and Davis, 1989] [Karunanithi et al., 1992] [Miller et al., 1989]. Otros, consideran todos los pesos de entrada y salida a un nodo de la capa intermedia como la unidad mínima intercambiable entre individuos [Thierens et al., 1993]. Otros autores [van Wanrooij, 1994], a pesar de encontrar intuitivo considerar un gen como los pesos de entrada a un nodo (intercambiar unidades funcionales completas), concluyen a partir de sus experimentos que dicho operador de cruce no ofrece ninguna ventaja sustancial sobre otros operadores que hagan alguna otra consideración.

Haciendo evolucionar redes RBF, Rivas et al. [Rivas et al., 1999] utilizan un operador de recombinación que intercambia secuencias de neuronas de las capas ocultas de dos redes neuronales. Ambas secuencias son seleccionadas de forma aleatoria, pueden tener cualquier longitud y variar la longitud de una respecto de la de la otra.

El operador de cruce propuesto por Castillo et al. [Castillo et al., 1999a] [Castillo et al., 1999b] [Castillo et al., 2000b] [Castillo et al., 2000c] lleva a cabo el cruce multipunto entre dos MLP, de forma que se obtienen dos redes cuyas neuronas en las capas ocultas son una mezcla de las neuronas ocultas de las dos redes: algunas neuronas ocultas (junto con sus pesos de entrada y salida) de cada MLP padre forman un descendiente, y el resto de neuronas ocultas forman el otro. En realidad, este tipo de operadores actúa como una macromutación, ya que cuando se intercambian nodos ocultos entre dos MLP, puesto que almacenan la información de forma distribuida entre todos sus pesos internos, los efectos producidos pueden degradar las redes resultantes.

3.3 Operadores incrementales y decrementales

Merelo et al. [Merelo et al., 1993b] [Merelo and Prieto, 1995] [Merelo et al., 1998] utilizan en el método G-LVQ, además de los operadores clásicos de un AG (cruce y mutación binarios), tres operadores de incremento y decremento de la longitud: el de *duplicación* toma un individuo y duplica el mejor gen, aumentando su tamaño; el de *eliminación* suprime el peor gen de un individuo, reduciendo así su tamaño; el de *incremento aleatorio* inserta un gen inicializado aleatoriamente en el individuo.

El método propuesto por Ragg et al. [Ragg et al., 1995] determina la topología de un MLP basándose en AE y en la teoría de la información, y utilizando un operador genético que añade o elimina diversas unidades

de las redes, según la información de la relación entre entrada/salida para dichas unidades.

Rivas et al. [Rivas et al., 1999] utilizan dos operadores de incremento/decremento para establecer el tamaño adecuado de redes RBF. El operador de incremento de neuronas duplica neuronas ocultas según una función de probabilidad lineal. Una vez duplicada, los centros y radios de la nueva neurona son modificados usando funciones gaussianas. El operador de decremento de neuronas, destruye neuronas de la capa oculta siguiendo una función de probabilidad lineal.

Castillo et al. [Castillo et al., 1999a] [Castillo et al., 1999b] [Castillo et al., 2000b] [Castillo et al., 2000c] proponen dos operadores pensados para atacar uno de los principales problemas de BP (y sus variantes): la dificultad de *adivinar* el número de neuronas en cada capa oculta. Añadiendo neuronas ocultas evitamos el problema de prefijar el tamaño del espacio de búsqueda del AE.

El *operador de incremento* lleva a cabo lo que se suele llamar "diseño incremental": comienza con estructuras pequeñas y las incrementa añadiendo nuevas unidades inicializadas aleatoriamente a las capas ocultas. En contrapartida, este incremento de tamaño puede hacer que los MLP tengan un tamaño excesivo: las redes grandes son más rápidas en la fase de aprendizaje [Bellido and Fernandez, 1991] [Bebis et al., 1997], aunque una forma de obtener una buena generalización es usar la red más simple que aproxime los datos de entrada [Reed, 1993] [Reed and Marks, 1999] [Yao, 1999].

Al mismo tiempo, y en contrapartida, este incremento de tamaño de los MLP nos plantea el problema del "sobreentrenamiento": las redes de pequeño tamaño generalizan bien aunque son lentas aprendiendo, mientras que redes grandes (alto número de parámetros) son rápidas aprendiendo pero generalizan mal [Bellido and Fernandez, 1991] [Bebis et al., 1997]. Esta es la razón por la cual se aplica este operador junto con el siguiente, y además está compensado con el componente de la función de evaluación que penaliza el tamaño de las redes.

El *operador de decremento* elimina una neurona de una capa oculta. Está pensado para llevar a cabo lo que se llama "diseño decremental": poda ciertas unidades ocultas para obtener redes más pequeñas [Jasic and Poh, 1995] [Pelillo and Fanelli, 1993] [Bebis et al., 1997]. En cierto modo, siguiendo este método se evita el problema de que las redes crezcan demasiado, llegando a tener un tamaño excesivo.

En dichos trabajos se puede ver cómo el uso de ambos operadores de forma conjunta hace que el AE lleve a cabo la búsqueda de la arquitectura de red, estableciendo el número de neuronas en las capas ocultas.

3.4 Macromutación

En vistas de los problemas que presenta el uso del cruce como operador, se plantea la cuestión de hasta qué punto es un operador útil [Holland, 1975]. Aquellos autores que están a favor del uso de este operador argumentan que debe usarse siempre que el problema y la representación de las soluciones permita que se formen "bloques de construcción" [Holland, 1975] (esquemas que representan una buena solución a una parte del problema a resolver). La idea es que si hay varios bloques, estos pueden ser tratados independientemente por el AE y recombinados por el operador de cruce.

Otros autores afirman que el hecho de que el operador de cruce ayude en la búsqueda no quiere decir que esté recombinando, como se supone que debe hacerlo, bloques constructivos, sino que los cambios que produce el operador son beneficiosos para la búsqueda en tanto que sus efectos son similares a utilizar un operador de mutación que provoque grandes cambios (macromutación) [Land, 1998].

Castillo et al. [Castillo et al., 1999a] [Castillo et al., 1999b] [Castillo et al., 2000b] [Castillo et al., 2000c] utilizan un operador de macromutación que sustituye una neurona oculta, elegida aleatoriamente, por otra, inicializada con pesos aleatorios. Se puede considerar una especie de macromutación que afecta a un sólo "gen" en el MLP.

3.5 Operadores de búsqueda local

BP, usado como operador de búsqueda local es un operador que refina las soluciones de forma que no se pueda mejorar localmente dicha solución. Comparado con un AE, este operador es mucho más eficiente buscando el óptimo local en un área concreta. Un AE, eventualmente, llegaría a encontrar dicho óptimo, pero le llevaría más tiempo, ya que el refinamiento que puede hacer se basa en pequeñas mutaciones.

Un problema que presentan este tipo de operadores de búsqueda se basa en el hecho de que algunos individuos que hayan sido refinados pueden llegar a dominar la población en las sucesivas generaciones, deteniendo así la evolución. Debido a esto, la búsqueda genética se puede ver alterada, sufriendo una pérdida de diversidad. Cuando se utiliza un operador genético que realiza una búsqueda local, la población convergerá rápidamente, ya que los individuos tienden a parecerse y a tener muchas características en común (reducción de la diversidad) tras aplicarles el operador de búsqueda local.

Braun y Weisbrod [Braun and Weisbrod, 1993] y Braun y Zagorski [Braun and Zagorski, 1994] propusieron un método según el cual, los individuos eran entrenados con BP hasta alcanzar un óptimo local tras ser modificados por algún operador genético, de forma que los descendientes heredaban los pesos de sus padres. Esta estrategia Lamarckiana (ver sección 4) reduce el tiempo de búsqueda, aunque restringe el espacio de búsqueda a aquellos individuos con mayor valor de la función de evaluación.

Montana y Davis [Montana and Davis, 1989] y Yao y Liu [Yao and Liu, 1998b] proponen utilizar un operador de entrenamiento que aplique el algoritmo BP al MLP para refinarlo, realizando una búsqueda local.

Castillo et al. [Castillo et al., 1999b] [Castillo et al., 2000b] proponen un operador de entrenamiento que aplica el algoritmo QP al MLP para refinarlo (realiza una búsqueda local para mejorar el individuo), tal y como se propone en [Montana and Davis, 1989] y en [Yao and Liu, 1998b]. Al aplicarlo, el operador toma un MLP y lo entrena un número de épocas especificado, devolviéndolo, con los pesos entrenados, a la población.

3.6 El problema de las permutaciones

El problema de entrenar una RNA no es una tarea adecuada para realizarla mediante AE que basen su funcionamiento principalmente en la recombinación de soluciones. Una propiedad de las RNA que provoca esto es que la aplicación entre entradas y salidas puede ser implementada por diversas redes simplemente permutando alguna de las neuronas de las capas ocultas [Radcliffe, 1991b] [Hancock, 1992] [Thierens et al., 1993]. A este problema se le llama *"el problema de las permutaciones"* o *"competing conventions problem"*.

El problema se presenta debido a que el AE no puede resolver dichas permutaciones, ya que sólo trabaja sobre la representación genotípica de la red. Para el AE, redes estructuralmente diferentes son soluciones candidatas diferentes, a pesar de que su funcionamiento sea idéntico. De esta forma, si el cruce se aplica a dos redes funcionalmente idénticas, pero estructuralmente diferentes, se obtendrán descendientes no útiles.

La manera más simple de evitar este problema es no utilizar el operador de cruce [Porto and Fogel, 1990] [Angeline et al., 1994] [Bornholdt and Graudenz, 1992] [Braun and Zagorski, 1994].

Otros autores han ideado operadores genéticos que eviten el problema sin renunciar al uso del operador de cruce. Montana y Davis [Montana and Davis, 1989] estudiaron diversas formas de identificar ciertos aspectos funcionales de los nodos ocultos durante el cruce para hacer algún tipo de cruce inteligente. Radcliffe [Radcliffe, 1991a] propuso evitar posibles permutaciones en la representación tratando de identificar aquellos nodos ocultos con el mismo patrón de conectividad en las redes que iban a ser recombinadas. Hancock [Hancock, 1992] utilizó la idea anterior y la extendió para descubrir similaridades entre nodos ocultos. Korning [Korning, 1994] propuso un operador de cruce que elimine automáticamente aquellos descendientes que no superen un mínimo en el valor de su función de evaluación. De es-

ta forma se filtran los descendientes producidos por padres incompatibles. Thierens et al. [Thierens et al., 1993] consideran que la función realizada por una unidad oculta viene dada por el signo de sus pesos. Así, antes de hacer el cruce, proponen hacer una reordenación de la cadena genética de los padres de forma que las neuronas ocultas con un número similar de pesos positivos y negativos estén en la misma posición de la cadena.

Finalmente, algunos autores utilizan el AE para encontrar un buen conjunto de pesos iniciales y posteriormente llevan a cabo una búsqueda mediante un algoritmo de descenso de gradiente [Skinner and Broughton, 1995] [Land, 1998]. Dicha aproximación ofrece mejores resultados que usar métodos basados en descenso de gradiente para resolver problemas con redes muy grandes.

4 Efecto Baldwin

La conocida hipótesis lamarckiana [Lamarck, 1809] establece que las características adquiridas durante la vida de un organismo se transmiten genéticamente a la descendencia. Dicha hipótesis se refiere normalmente a características físicas, aunque se puede pensar en aquellas cosas aprendidas durante la vida del organismo como características aprendidas. Así pues, según Lamarck, el conocimiento adquirido puede dirigir la evolución directamente, siendo pasado a las siguientes generaciones codificado en el código genético de los individuos. Dichas hipótesis han sido rechazadas por los biólogos, ya que no existe un mecanismo para transcribir las características adquiridas al código genético.

Esto no significa que el aprendizaje no tenga efectos en la evolución. Baldwin [Baldwin, 1896] (y posteriormente Waddington [Waddington, 1942]) sugirió que si el aprendizaje ayuda a sobrevivir, aquellos organismos con más capacidad de aprendizaje tendrán más posibilidades de reproducirse, con lo cual los genes responsables del aprendizaje incrementarán su frecuencia en la población. Es más, si aquellas cosas que se necesitaban aprender permanecen constantes en el entorno, la selección natural puede llevar a que dichas características aprendidas acaben codificadas en el genotipo.

En este sentido, el aprendizaje guia la evolución, aunque aquello que se ha aprendido no se transmite directamente al código genético.

Se puede decir que la capacidad de aprender ayuda a los organismos a responder ante cambios de su entorno (aspectos que cambian demasiado rápido para que la evolución los asimile en el código genético). El efecto Baldwin afirma que el aprendizaje ayuda a los organismos a adaptarse a aspectos del entorno predecibles genéticamente, y que también ayuda indirectamente a que esas adaptaciones acaben codificadas en el código genético.

Las ideas anteriores han sido utilizadas por numerosos investigadores en diferentes enfoques:

Mecanismos Lamarckianos en algoritmos híbridos. La teoría de Lamarck está hoy día totalmente desacreditada, sin embargo es posible implementar en AE la evolución Lamarckiana, de forma que un individuo pueda modificar su material genético. Estas ideas han sido usadas por diversos investigadores con especial éxito en problemas en los que la aplicación de un operador de búsqueda local a los individuos consigue una mejora sustancial (problema del viajante de comercio, Gorges-Schleuter [Gorges-Schleuter, 1997], Merz y Freisleben [Merz and Freisleben, 1997], Ross [Ross, 1999]). En general, los algoritmos híbridos están considerados como la mejor solución a un ámplio rango de problemas de optimización.

Estudio del efecto Baldwin en algoritmos híbridos. Otros autores estudian el efecto Baldwin, esto es, llevan a cabo una búsqueda local sobre ciertos individuos para mejorar el valor de su función de evaluación sin modificar el individuo en sí. Esta es la estrategia presentada por Hinton y Nowlan en [Hinton and Nowlan, 1987], donde se presenta un ejemplo de algoritmo híbrido para ilustrar el efecto Baldwin, llegando a la conclusión de que el aprendizaje altera el espacio de búsqueda en el que la evolución está operando, y el efecto Baldwin hace que aquellos organismos con capacidades de aprender evolucionen más rápidamente que otros que no posean esas capacidades, aunque aquello que aprendan no

quede codificado en el código genético. Belew [Belew, 1989] y Harvey [Harvey, 1993] llevaron a cabo sendas críticas del método utilizado por Hinton y Nowlan y de los anómalos resultados presentados por estos en su trabajo sobre el efecto Baldwin. Ackley y Littman [Ackley and Littman, 1992] estudiaron el efecto Baldwin en un sistema de vida artificial, llegando al resultado de que en los experimentos en los que los individuos tenían capacidades de aprendizaje se obtenían los mejores resultados. Boers et al. [Boers et al., 1995] describen un algoritmo híbrido para evolucionar arquitecturas de red, cuya efectividad queda explicada con el efecto Baldwin, implementado no como un proceso de aprendizaje en la red, sino cambiando la arquitectura de red como parte del proceso de aprendizaje.

Estudios comparativos de los mecanismos Lamarckianos y del efecto Baldwin en algoritmos híbridos. Por otro lado, existen estudios comparativos en los que se intenta decidir si una estrategia basada en un algoritmo híbrido en el que se produce el efecto Baldwin es más o menos adecuada que implementar mecanismos Lamarckianos para acelerar la búsqueda [Castillo et al., 2001b]. Los resultados obtenidos en dichos estudios han sido dispares, y bastante dependientes del problema en cuestión. Así, en los estudios realizados por Gruau y Whitley [Gruau and Whitley, 1993b] se comparan estrategias Baldwinianas, Lamarckianas y Darwinianas aplicadas a un algoritmo genético que evoluciona RNA, obteniendo que las dos primeras estrategias son igualmente efectivas en su problema. Sin embargo, para otro problema, los resultados presentados por Whitley et al. [Whitley et al., 1994] muestran que el aprovechar el efecto Baldwin puede conseguir la convergencia al óptimo global, mientras que una estrategia Lamarckiana, a pesar de ser más rápida, puede converger a un óptimo local. Por otro lado, los resultados presentados por Ku y Mak [Ku and Mak, 1997] con un AG diseñado para evolucionar redes recurrentes, muestran que el uso de una estrategia Lamarckiana implica una mejora en el algoritmo, mientras que el efecto Baldwin no lo consigue. En Houck et al. [Houck et al., 1997] se estudian varios algoritmos, llegándose a conclusiones similares a las anteriores, al igual que en [Julstrom, 1999], donde se realiza una comparación entre las búsquedas Darwiniana, Baldwiniana y Lamarckiana.

Las estrategias Lamarckianas presentan la ventaja de que pueden acelerar la búsqueda, y cuando menos, la refinan, haciendo que la solución sub-óptima (pero muy cercana al óptimo) encontrada por el AE se acerque un poco más al óptimo; sin embargo, presentan algunas desventajas que hacen que la aplicación de operadores Lamarckianos se deba llevar a cabo con ciertas limitaciones. Dichos problemas son que estas estrategias limitan la diversidad y pueden llegar a detener la evolución, ya que aquellas características aprendidas al principio de la simulación pueden hacer que ciertos individuos pasen a dominar la población por la ventaja adquirida en cierto momento y continúen siendo los mejores individuos hasta el final de la simulación [Oliveira et al., 1999] [Castillo et al., 2001b].

5 Individuos de Longitud Variable. Intrones

En aquellos AE que utilizan individuos de longitud variable (en principio no tienen un número de genes fijo), en los cuales cada gen codifica una parte de la solución, y dicha codificación no depende de la posición de los genes, sino sólo del valor de estos, aparecen en los individuos *segmentos de código genético no utilizados para codificar características*, también llamados "intrones" [Castellano et al., 2001].

Este tipo de AE (con individuos de longitud variable) se utiliza en problemas en los que las soluciones tienen una cantidad variable de partes indistinguibles, tal y como RNA [Harvey and Thompon, 1996] [Merelo et al., 1993b]
[Merelo and Prieto, 1995] [Merelo et al., 1998] y controladores difusos [Rojas et al., 1997].

Los intrones se han utilizado de diversas formas:

- Como *bits que no codifican características*, añadidos uniformemente a la representación de la solución. En este caso, los intrones rellenan espacio entre los parámetros reales [Wu and Lindsay, 1995] [Levenick, 1991].

- Como *partes no funcionales* de la representación, esto es, partes que realmente no hacen nada (no contribuyen al valor de la función de evaluación del individuo). Esto es lo que ocurre habitualmente en programación genética [Nordin and Banzhaf, 1995] y en aquellos métodos cuyos individuos se desarrollan tras ser creados [Nolfi and Parisi, 1992].

- Como *partes no útiles a posteriori* del individuo, esto es, partes de la representación genética que no contribuyen al valor de la función de evaluación de la solución. Esto es lo que ocurre habitualmente en algunos tipos de RNA, en los cuales sólo ciertas neuronas llamadas *ganadoras* cambian sus pesos durante el entrenamiento [Kohonen, 1982a] [Kohonen, 1982b] [Kohonen, 1990] [Kohonen, 1997] [Rumelhart, 1985]. Aquellas neuronas que no han llegado a ser ganadoras durante el entrenamiento no resultan útiles, aunque eso se sabe *a posteriori*, y de hecho, con otra secuencia de entrenamiento podrían resultar funcionales [Castellano et al., 2001].

Wu et al. [Wu and Lindsay, 1995] mantienen la hipótesis de que los intrones de los individuos mantienen la variabilidad entre los individuos (esto es, mantienen alta la diversidad en la población). Por otro lado, Harvey et al. [Harvey and Thompon, 1996] probaron que los intrones fomentan la evolución de la población.

Con el fin de controlar el porcentaje de intrones en los individuos de la población se suelen utilizar dos técnicas:

- *Utilizar operadores que alteran la longitud de los individuos* y que introducen o eliminan intrones. Los experimentos que siguen esta línea muestran que una ligera presión selectiva sobre los intrones mejora la búsqueda, mientras que la eliminación de estos prima la exploración del espacio de búsqueda, perdiendo buenas soluciones [Castellano et al., 2001].

- *Por selección*: de forma que dependiendo del número de intrones, el valor de la función de evaluacion del individuo será mayor o menor [Harvey and Thompon, 1996].

6 Conclusiones

En este trabajo se ha realizado una revisión de los diferentes enfoques evolutivos que tratan de diseñar RNA, estableciendo los diversos parámetros que la definen, adaptándolas al problema a resolver.

Los diferentes enfoques estudiados se centran básicamente en tres niveles:

- *Evolución de los pesos de conexión*: La simplicidad y generalidad del enfoque evolutivo y el hecho de que los algoritmos de entrenamiento basados en descenso de gradiente pueden quedar atrapados en óptimos locales hacen que el uso de un AE para llevar a cabo el entrenamiento de los pesos de conexión sea una aproximación plausible.

- *Evolución de la arquitectura de red*: Como se comentó anteriormente (ver 2.3) este enfoque tiene varias ventajas sobre otras heurísticas de diseño de arquitecturas de red. A pesar de esto, la evolución de la arquitectura junto con los pesos de conexión usualmente genera mejores resultados.

- *Evolución de la regla de aprendizaje*: En este caso, la evolución se usa para hacer que la RNA adapte su regla de aprendizaje. Dicha adaptación se puede llevar a cabo haciendo evolucionar los parámetros de aprendizaje de la red (el momento o la tasa de aprendizaje); o bien las reglas de aprendizaje, esto es, las reglas de actualización de los pesos en el algoritmo de entrenamiento de la red.

La principal ventaja del diseño de RNA mediante un AE radica en la capacidad de éste para optimizar los diversos parámetros de la red (arquitectura, conectividad, pesos iniciales, regla de aprendizaje), y su paralelismo inherente (las diferentes redes pueden ser entrenadas simultáneamente en distintos procesadores).

La mayoría de los métodos comentados a lo largo de este trabajo implementan dos o a lo sumo tres de los enfoques estudiados, dando lugar a intentos sólo parciales de optimización de la RNA, con lo cual no está garantizada la consecución de un óptimo global. Es por esto que

resultará beneficioso utilizar varios de los enfoques estudiados al mismo tiempo, sobre todo en aquellos casos en los que haya poco conocimiento a priori acerca del problema, ya que en estas circunstancias, el uso de métodos de prueba y error o de métodos heurísticos no es efectivo.

Los algoritmos tradicionales tratan de optimizar una sola cantidad (por ejemplo, la capacidad de clasificación) a expensas de otras, como el tamaño de la red; utilizando dos criterios se pueden optimizar la capacidad de clasificación/aproximación y el tamaño de la red (número de pesos total).

El estudio del efecto Baldwin en aquellos métodos que hacen evolución de RNA puede producir, de alguna forma, la mejora de dichos métodos, ya que de acuerdo con los resultados presentados por diversos autores, aprovechar el efecto Baldwin puede dar lugar a la convergencia al óptimo global; mientras que estrategias basadas en la utilización de operadores genéticos lamarckianos, a pesar de una mayor rapidez, pueden converger a un óptimo local.

Actualmente, la mayoría de los métodos comentados en este trabajo sólo implementan dos o, a lo sumo tres enfoques de los estudiados. Sólo son intentos parciales de optimización de RNA, por lo que no se garantiza la obtención del optimo global. A corto plazo, sería interesante utilizar varios de los enfoques al mismo tiempo, sobre todo en aquellos casos en los que haya poco conocimiento a prioiri sobre el problema. Así pues, un método que optimice los diferentes parámetros de una RNA (tamaño de red, pesos iniciales, parámetros de aprendizaje, vector de entradas a la red y los conjuntos de entrenamiento/validación/test) para una determinada arquitectura sería muy últil. De esta forma, una vez seleccionada una arquitectura adecuada junto con el algoritmo de entrenamiento de la red, el método podría buscar la mejor red de ese tipo para resolver el problema. En este sentido, el método G-Prop (ver [Castillo et al., 2000b, Castillo et al., 2000c, Castillo et al., 2001a]) realiza la búsqueda del tamaño de red, de los pesos iniciales y los parámetros de aprendizaje para la optimización de MLP entrenados usando quickpropagation.

A largo plazo, sería interesante desarrollar un método que busque los diferentes parámetros de la RNA, además de la arquitectura y el algoritmo de aprendizaje; así, dependiendo del problema a resolver, el método busca la mejor arquitectura (RBF, MLP, recurrente, etc.), un algoritmo de entrenamiento adecuado (BP, QP, OLS, SVD, etc.), y entonces buscaría la mejor red para resolver el problema. Desde el punto de vista del software, sería muy últil desarrollar una librería de computación evolutiva y de RNA que incluyera todos los enfoques presentados.

Reconocimientos

Este trabajo ha sido financiado en parte por los proyectos FEDER I+D 1FD97-0439-TEL1, CICYT TIC99-0550 e INTAS 97-30950.

Referencias

[Ackley and Littman, 1991] Ackley, D. and Littman, M. (1991). Interactions between learning and evolution. *in Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X (C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, eds.), (Reading, MA), pp. 487-509, Addison-Wesley.*

[Ackley and Littman, 1992] Ackley, D. and Littman, M. (1992). Interactions between learning and evolution. *In C.G. Langton, C. Taylor, J.D. Garmer, and S. Rasmussen (Editors), Artificial Life II, 487-507, Addison-Wesley, Reading, MA.*

[Alba et al., 1993] Alba, E., Aldana, J., and Troya, J. (1993). Fully Automatic ANN Design: A Genetic Approach. *Lecture Notes in Computer Science, Vol. 686, pp. 399-404, Springer-Verlag.*

[Angeline, 1997] Angeline, P. (1997). Evolving basis functions with dynamic receptive fields. *in Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Part 5 (of 5), (Piscataway, NJ, USA), pp. 4109-4114, IEEE Press.*

[Angeline et al., 1994] Angeline, P., Saunders, G., and Pollack, J. (1994). An evolutionary algorithm that constructs recurrent neu-

ral networks. *IEEE Transactions on Neural Networks, 5(1):54-65.*

[Balakrishnan and Honavar, 1995] Balakrishnan, K. and Honavar, V. (1995). Evolutionary design of neural architectures – a preliminary taxonomy and guide to literature. Technical report, AI Research Group. CS-TR 95-01.

[Baldwin, 1896] Baldwin, J. (1896). A new factor in evolution. *American Naturalist 30, 441-451.*

[Bartlett and Downs, 1990] Bartlett, P. and Downs, T. (1990). Training a neural network with a genetic algorithm. *Technical Report, Dept. of Elec Eng., Univ of Queensland.*

[Baxter, 1992] Baxter, J. (1992). The evolution of learning algorithms for artificial neural networks. *in Complex Systems (D. Green and T. Bossomaier, eds.), pp. 313-326, IOS Press, Amsterdam.*

[Bebis et al., 1997] Bebis, G., Georgiopoulos, M., and Kasparis, T. (1997). Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing 17 (1997) 167-194.*

[Belew, 1989] Belew, R. (1989). When both individuals and populations search: Adding simple learning to the genetic algorithm. *In 3th Intern. Conf. on Genetic Algorithms, D. Schaffer, ed., Morgan Kaufmann.*

[Belew, 1990] Belew, R. (1990). Evolution, learning and culture: Computational metaphors for the adaptive algorithms. *Complex Systems vol. 4, pp. 11-49.*

[Belew et al., 1991] Belew, R., McInerney, J., and Schraudolph, N. (1991). Evolving networks: using genetic algorithm with connectionist learning. *Tech. Rep. CS90-174 (Revised, Computer Science and Engr. Dept. (C-014), Univ. of California at San Diego, La Jolla, CA 92093, USA.*

[Bellido and Fernandez, 1991] Bellido, I. and Fernandez, G. (1991). Backpropagation Growing Networks: Towards Local Minima Elimination. *Lecture Notes in Computer Science, Vol. 540, pp. 130-135, Springer-Verlag.*

[Bengio et al., 1992] Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. (1992). On the optimization of a synaptic learning rule. *in Preprints of the Conference on Optimality in Artificial and Biological Neural Networks, (Univ. of Texas, Dallas).*

[Bengio and Bengio, 1990] Bengio, Y. and Bengio, S. (1990). Learning a synaptic learning rule. *Tech. Rep. 751, Départment d'Informatique et de Recherche Opérationelle, Université de Montréal, Canada.*

[Bernier et al., 2000a] Bernier, J., Ortega, J., Rojas, I., and Prieto, A. (2000a). Improving the tolerance of multilayer perceptrons by minimizing thestatistical sensitivity to weight deviations. *Neurocomputing, Vol.31, No.1-4. pp.87-103.*

[Bernier et al., 1999] Bernier, J., Ortega, J., Ros, E., Rojas, I., and Prieto, A. (1999). A new measurement of noise immunity and generalization ability for mlps. *International Journal of Neural Systems, Vol. 9, No. 6,pp.511-522.*

[Bernier et al., 2000b] Bernier, J., Ortega, J., Ros, E., Rojas, I., and Prieto, A. (2000b). A Quantitative Study of Fault Tolerance, Noise Immunity and Generalization Ability of MLPs. *Neural Computation, Vol.12, pp.2941-2964.*

[Bernier et al., 2000c] Bernier, J., Ortega, J., Ros, E., Rojas, I., and Prieto, A. (2000c). Obtaining fault tolerant multilayer perceptrons using an explicit regularization. *Neural Processing Letters, Vol. 12, No.2, pp. 107-113.*

[Boers et al., 1995] Boers, E., Borst, M., and Sprinkhuizen-Kuyper, I. (1995). Evolving Artificial Neural Networks using the Baldwin Effect. *In D.W. Pearson, N.C. Steele and R.F. Albrecht (eds.) and Artificial Neural Nets and Genetic Algorithms. Proceedings of the International Conference in Alès, France, 333-336, Springer Verlag Wien New York.*

[Boers and H.Kuiper, 1992] Boers, E. and H.Kuiper (1992). Biological Metaphors and the Design of Modular Artificial Neural Networks. *Master's thesis, Leiden University, Leiden , The Netherlands.*

[Bornholdt and Graudenz, 1992] Bornholdt, S. and Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks, 5:327-334.*

[Bottou, 1988] Bottou, L. (1988). Reconnaissance de la parole par reseaux multi-couches. *In Neuro-Nimes'88. ISBN:2-906899-14-3.*

[Braun and Weisbrod, 1993] Braun, H. and Weisbrod, J. (1993). Evolving neural feedforward networks. *In Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms, pp. 25-32. Springer-Verlag.*

[Braun and Zagorski, 1994] Braun, H. and Zagorski, P. (1994). Enzo-II - a powerful design tool to evolve multilayer feed forward networks. *In Proceedings of the first IEEE Conference on Evolutionary Computation, vol. 2, pp. 278-283.*

[Brill et al., 1992] Brill, F., Brown, D., and Martin, W. (1992). Fast genetic selection of features for neural network classifiers. *IEEE Transctions on Neural Networks, vol. 3, pp. 324-328.*

[Buntine and Weigend, 1994] Buntine, W. and Weigend, A. (1994). Calculating second derivatives on feed-forward networks: a review. *IEEE Transactions on Neural Networks 5, 480-488.*

[Castellano et al., 2001] Castellano, J., Castillo, P., and Merelo, J. (2001). Scrapping or recycling: the role of chromosome length-altering operators in Genetic Algorithms. *Submitted to 5th International Conference on Artificial Evolution. Bourgogne, October 29-31.*

[Castillo et al., 2000a] Castillo, P., Arenas, M., Castellano, J., Carpio, J., Merelo, J., Prieto, A., Rivas, V., and Romero, G. (2000a). Function Approximation with Evolved Multilayer Perceptrons. *in Proc. of Int'l Workshop on Evolutionary Computation (IWEC'2000) pp. 209-224. State Key Laboratory of Software Engineering. Wuhan University.*

[Castillo et al., 2001a] Castillo, P., Arenas, M., Castellano, J., M.Cillero, Merelo, J., Prieto, A., Rivas, V., and Romero, G. (2001a). Function Approximation with Evolved Multilayer Perceptrons. *Advances in Neural Networks and Applications. Artificial Intelligence Series. Nikos E. Mastorakis Editor. ISBN:960-8052-26-2, pp.195-200, Published by World Scientific and Engineering Society Press.*

[Castillo et al., 2000b] Castillo, P., Carpio, J., Merelo, J., Rivas, V., Romero, G., and Prieto, A. (2000b). Evolving Multilayer Perceptrons. *Neural Processing Letters, vol. 12, no. 2, pp.115-127. October.*

[Castillo et al., 2001b] Castillo, P., Castellano, J., Merelo, J., and Romero, G. (2001b). Lamarckian Evolution and Baldwin Effect in Artificial Neural Networks Evolution. *Submitted to 5th International Conference on Artificial Evolution. Bourgogne, October 29-31.*

[Castillo et al., 2001c] Castillo, P., de la Torre, J., Merelo, J., and Román, I. (2001c). Forecasting business failure. A comparison of neural networks and logistic regression for the Spanish companies. *In 24th Annual Congress European Accounting Association, pp.182. Athens, 18-20 April.*

[Castillo et al., 1999a] Castillo, P., González, J., Merelo, J., Rivas, V., Romero, G., and Prieto, A. (1999a). G-Prop-II: Global Optimization of Multilayer Perceptrons using GAs. *In Congress on Evolutionary Computation, ISBN:0-7803-5536-9, Volume III, pp. 2022-2027, Washington D.C., USA.*

[Castillo et al., 1999b] Castillo, P., González, J., Merelo, J., Rivas, V., Romero, G., and Prieto, A. (1999b). G-Prop-III: Global Optimization of Multilayer Perceptrons using an Evolutionary Algorithm. *In Congress on Evolutionary Computation,In Genetic and Evolutionary Computation Conference, ISBN:1-55860-611-4, Volume I, pp. 942, Orlando, USA.*

[Castillo et al., 1999c] Castillo, P., González, J., Merelo, J., Rivas, V., Romero, G., and Prieto, A. (1999c). SA-Prop: Optimization of Multilayer Perceptron Parameters using Simulated Annealing. *Lecture Notes in Computer Science, ISBN:3-540-66069-0, Vol. 1606, pp. 661-670, Springer-Verlag.*

[Castillo et al., 2000c] Castillo, P., Merelo, J., Rivas, V., Romero, G., and Prieto, A. (2000c). G-Prop: Global Optimization of Multilayer Perceptrons using GAs. *Neurocomputing, Vol.35/1-4, pp.149-163.*

[Caudell and Dolan, 1989] Caudell, T. and Dolan, C. (1989). Parametric connectivity: training of constrained networks using genetic algorithms. *in Proc. of the Third Int'l Conf. on Genetic Algorithms and Their Applications (J.D. Schaffer, ed.), pp. 370-374, Morgan Kaufmann, San Mateo, CA.*

[Chalmers, 1990] Chalmers, D. (1990). The evolution of learning: an experiment in genetic connectionism. *in Proceedings of the 1990 Connectionist Models Summer School (D.S. Touretzky, J.L. Elman, and G.E. Hinton, eds.), pp. 81-90, Morgan Kaufmann, San Mateo, CA.*

[Chen and Nutter, 1991] Chen, C. and Nutter, R. (1991). Improving the training speed of three-layer feedforward neural nets by optimal estimation of the initial weights. *In International Joint Conference on Neural Networks, vol. 3, pp. 2063-2068. IEEE.*

[Cox and Cox, 1994] Cox, T. and Cox, M. (1994). Multidimensional scaling. *London: Chapman and Hall.*

[Crosher, 1993] Crosher, D. (1993). The artificial evolution of a generalized class of adaptive processes. *in Preprints of AI'93 Workshop on Evolutionary Computation (X. Yao, ed.), pp. 18-36.*

[Cun et al., 1990] Cun, Y. L., Denker, J., and Solla, S. (1990). Optimal brain damage. *Neural Information Systems 2. Touretzky, D.S. (ed) Morgan-Kauffman, pp. 598-605.*

[DasGupta and Schnitger, 1992] DasGupta, B. and Schnitger, G. (1992). Efficient approximation with neural networks: a comparison of gate functions. *tech. rep., Delp. of Computer Sci., Pennsylvania State Univ., University Park, PA 16802.*

[de Falco et al., 1998a] de Falco, I., Cioppa, A. D., Iazzetta, A., Natale, P., and Tarantino, E. (1998a). Optimizing Neural Networks for Time Series Prediction. *Third World Conference on Soft Computing (WSC3), June 1998.*

[de Falco et al., 1998b] de Falco, I., Iazzetta, A., Natale, P., and Tarantino, E. (1998b). Evolutionary Neural Networks for Nonlinear Dynamics Modeling. *Lecture Notes in Computer Science, Vol. 1498, pp. 593-602, Springer-Verlag.*

[de Garis, 1991] de Garis, H. (1991). Gennets: genetically programmed neural nets - using the genetic algorithm to train neural nets whose inputs and/or outputs vary in time. *in Proc. of 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Singapore), vol. 2, pp. 1391-1396, IEEE Press, New York.*

[Dellaert and Vandewalle, 1994] Dellaert, F. and Vandewalle, J. (1994). Automatic design of cellular neural networks by means of genetic algorithms: Finding a feature detector. *in Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, (Piscataway, NJ, USA), pp. 189-194, IEEE Press.*

[Denoeux and Lengellé, 1993] Denoeux, T. and Lengellé, R. (1993). Initializaing back propagation nerworks with prototypes. *Neural Networks, vol. 6, pp. 351-363, Pergamon Press Ltd.*

[Erkmen and Ozdogan, 1997] Erkmen, I. and Ozdogan, A. (1997). Shjort term load forecasting using genetically optimized neural network cascaded with a modified kohonen clustering process. *in Proceedings of the 1997 IEEE International Symposium on Intelligent Control, (Piscataway, NJ, USA), pp. 107-112, IEEE Press.*

[Fahlman, 1988] Fahlman, S. (1988). Faster-learning variations of back-propagation: An empirical study. *In D.S. Touretzky, G. Hinton, and T. Sejnowski, editors, Proceedings of the 1988 Connectionist Models Summer School, pp.38-51. Morgan Kaufmann, San Mateo.*

[Fahlman and Lebière, 1990] Fahlman, S. and Lebière, C. (1990). The Cascade-Correlation Learning Architecture. *Neural Information Systems 2. Touretzky, D.S. (ed) Morgan-Kauffman, 524-532.*

[Fels and Hinton, 1993] Fels, S. and Hinton, G. (1993). Glove-talk: a neural network interface between a data-glove and a speech synthesizer. *IEEE Trans. on Neural Networks, vol. 4, pp. 2-8.*

[Fogel et al., 1991] Fogel, D., Fogel, L., and Atmar, J. (1991). Meta-evolutionary programming. *In R.R. Chen, editor, Proceedings of 25th Asilomar Conference on Signals, Systems and Computers, pp. 540-545, Pacific Grove, California.*

[Fogel et al., 1990] Fogel, D., Fogel, L., and Porto, V. (1990). Evolving neural networks. *Biological Cybernetics, vol. 63, pp. 487-493.*

[Fogel et al., 1995] Fogel, D., Wasson, E., and Boughton, E. (1995). Evolving neural networks for detecting breast cancer. *Cancer Letters, vol. 96, no. 1, pp. 49-53.*

[Fogel et al., 1997] Fogel, D., Wasson, E., and Porto, V. (1997). A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer Letters, vol. 119, no. 1, pp.93.*

[Fontanari and Meir, 1991] Fontanari, J. and Meir, R. (1991). Evolving a learning algoritm for the binary perceptron. *Network, vol. 2, pp. 353-359.*

[Frean, 1990] Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation 2, pp. 198-209.*

[Gallant, 1990] Gallant, S. (1990). Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks 1, pp. 179-191.*

[Gallant, 1993] Gallant, S. (1993). Neural network learning and expert systems. *Cambridge, MA: The MIT Press.*

[Goldberg, 1989] Goldberg, D. (1989). Genetic algorithms in search, optimization, and machine learning. *Addison-Wesley.*

[Gorges-Schleuter, 1997] Gorges-Schleuter, M. (1997). Asparagos96 and the traveling salesman problem. *In Proceedings of 1997 IEEE International Conference on Evolutionary Computation, pp. 171-174. IEEE.*

[Greenwood, 1997] Greenwood, G. (1997). Training partially recurrent neural networks using evolutionary strategies. *IEEE Transactions on Speech and Audio Processing, vol. 5, no. 2, pp. 192-194.*

[Grönroos, 1998] Grönroos, M. (1998). Evolutionary Design of Neural Networks. *Master of Science Thesis in Computer Science. Department of Mathematical Sciences. University of Turku.*

[Gruau, 1992a] Gruau, F. (1992a). Cellular enconding of genetic neural networks. *Technical Report, LIP-IMAG Ecole Normale Superieure de Lyon, 46 Allée d'Italie 69007 Lyon, France.*

[Gruau, 1992b] Gruau, F. (1992b). Genetic synthesis of boolean neural networks with a cell rewriting developmental process. *in Proc. of the Int'l Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN'92) (D. Whitley and J.D. Schaffer, eds.), pp. 55-74, IEEE Computer Society Press, Los Alamitos, CA.*

[Gruau and Whitley, 1993a] Gruau, F. and Whitley, D. (1993a). Adding learning to the cellular development of neural networks: Evolution and the Baldwin efect. *Evolutionary Computation, Volume I, No. 3, pp. 213-233.*

[Gruau and Whitley, 1993b] Gruau, F. and Whitley, D. (1993b). Adding learning to the cellular development of neural networks: Evolution and the Baldwin efect. *Evolutionary Computation, Volume I, No. 3, pp. 213-233.*

[Guo and Uhrig, 1992] Guo, Z. and Uhrig, R. (1992). Using genetic algorithms to select inputs for neural networks. *in Proc. of the Int'l Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)(D. Whitley and J.D. Schaffer, eds.), pp. 223-234, IEEE Computer Society Press, Los Alamitos, CA.*

[Hancock, 1992] Hancock, P. (1992). Coding strategies for genetic algorithms and neural nets. *PhD thesis, Department of Computing Science and Mathematics, University of Stirling.*

[Hansen and Meservy, 1996] Hansen, J. and Meservy, R. (1996). Learning experiments with genetic optimization of a generalized regression neural network. *Decision Support Systems, vol. 18, no. 3-4, pp. 317-325.*

[Harp et al., 1989] Harp, S., Samad, T., and Guha, A. (1989). Towards the genetic synthesis of neural networks. *in Proc. of the*

Third Int'l Conf. on Genetic Algorithms and Their Applications (J.D. Schaffer, ed.), pp. 360-369, Morgan Kaufmann, San Mateo, CA.

[Harp et al., 1990] Harp, S., Samad, T., and Guha, A. (1990). Designing application-specific neural networks using the genetic algorithm. *in Advances in Neural Information Processing Systems 2 (D.S. Touretzky, ed.), pp. 447-454, Morgan Kaufmann, San Mateo, CA.*

[Harvey, 1993] Harvey, I. (1993). The puzzle of the persistent question marks: a case study of genetic drift. *In 5th International Conference on Genetic Algorithms, pp. 15-22, S. Forrest, ed. Morgan Kaufmann.*

[Harvey and Thompon, 1996] Harvey, I. and Thompon, A. (1996). Through the labyrinth evolution finds a way: A silicon ridge. *In Proc. of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES'96). Springer-Verlag.*

[Hassibi et al., 1994] Hassibi, B., Stork, D., Wolff, G., and Watanabe, T. (1994). Optimal Brain Surgeon: extensions and performance comparisons. *In NIPS6, pp. 263-270.*

[Hinton and Nowlan, 1987] Hinton, G. and Nowlan, S. (1987). How learning can guide evolution. *Complex Systems, 1, 495-502.*

[Holland, 1975] Holland, J. (1975). Adaptation in natural and artificial systems. *University of Michigan Press (Second Edition: MIT Press, 1992).*

[Hornyak and Monostori, 1997] Hornyak, J. and Monostori, L. (1997). Feature extraction technique for ann-based financial forecasting. *Neural Network World, vol. 7, no. 4-5, pp. 543-552.*

[Houck et al., 1997] Houck, C., Joines, J., Kay, M., and Wilson, J. (1997). Empirical investigation of the benefits of partial Lamarckianism. *Evolutionary Computation, v.5, n.1, pp. 31-60.*

[Hsu and Wu, 1992] Hsu, L. and Wu, Z. (1992). Input pattern enconding through generalised adaptive search. *in Proc. of the Int'l Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)(D. Whitley and J.D. Schaffer, eds.), pp. 235-247, IEEE Computer Society Press, Los Alamitos, CA.*

[Janson and Frenzel, 1992] Janson, D. and Frenzel, J. (1992). Application of genetic algorithms to the training of higher order neural networks. *Journal of Systems Engineering, vol. 2, pp. 272-276.*

[Janson and Frenzel, 1993] Janson, D. and Frenzel, J. (1993). Training product unit neural networks with genetic algorithms. *IEEE Expert, vol. 8, no. 5, pp. 26-33.*

[Jasic and Poh, 1995] Jasic, T. and Poh, H. (1995). Analysis of Pruning in Backpropagation Networks for Artificial and Real World Mapping Problems. *Lecture Notes in Computer Science, Vol. 930, pp. 239-245, Springer-Verlag.*

[Julstrom, 1999] Julstrom, B. (1999). Comparing Darwinian, Baldwinian and Lamarckian Search in a Genetic Algorithm for the 4-Cycle Problem. *In Congress on Evolutionary Computation,In Genetic and Evolutionary Computation Conference, Late Breaking Papers, pp. 134-138, Orlando, USA.*

[Karunanithi et al., 1992] Karunanithi, N., Das, R., and Whitley, D. (1992). Genetic cascade learning for neural networks. *In Schaffer and Whitley, editors, Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 134-144.*

[Kim et al., 1996] Kim, H., Jung, S., Kim, T., and Park, K. (1996). Fast learning method for backpropagation neural network by evolutionary adaptation of learning rates. *Neurocomputing, vol. 11, no. 1, pp. 101-106.*

[Kim and Choi, 1997] Kim, M. and Choi, C. (1997). A new weight initialization method for the MLP with the BP in multiclass classification problems. *Neural Processing Letters 6: 11-23.*

[Kinnebrock, 1994] Kinnebrock, W. (1994). Accelerating the standard backpropagation method using a genetic approach. *Neurocomputing, 6, 583-588.*

[Kitano, 1990a] Kitano, H. (1990a). Designing neural networks using genetic algorithms

with graph generation system. *Complex Systems, vol. 4, no. 4, pp. 461-476.*

[Kitano, 1990b] Kitano, H. (1990b). Empirical studies on the speed of convergence of neural network training using genetic algorithms. *in Proc. of the Eighth Nat'l Conf. on AI (AAAI-90), pp. 789-795, MIT Press, Cambridge, MA.*

[Knerr et al., 1992] Knerr, S., Personnaz, L., and Dreyfus, G. (1992). Handwritten digit recognition by neural networks with single-layer training. *IEEE Trans. on Neural Networks, vol. 3, pp. 962-968.*

[Koeppen et al., 1997] Koeppen, M., Teunis, M., and Nickolay, B. (1997). Neural network that uses evolutionary learning. *in Proceedings of the 1997 Internaational Conference on Evolutionary Computation, ICEC'97, (Piscataway, NJ, USA), pp. 1023-1028, IEEE Press.*

[Kohonen, 1982a] Kohonen, T. (1982a). Clustering, taxonomy, and topological maps of patterns. *In Proc. of the 6th Int. Conf. on Pattern Recognition. IEEE Computer Society Press.*

[Kohonen, 1982b] Kohonen, T. (1982b). Self-organizing formation of topologically correct features maps. *Biological Cybernetics, vol. 43, pp.59-69.*

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. *Procs. IEEE, vol. 78, no.9, pp. 1464-1480.*

[Kohonen, 1997] Kohonen, T. (1997). Self-organizing maps. *Segunda Edición, Springer.*

[Kolen and Pollack, 1990] Kolen, J. and Pollack, J. (1990). Back Propagation is Sensitive to Initial Conditions. *Technical Report TR 90-JK-BPSIC. Laboratory for Artificial Intelligence Research, Computer and Information Science Department.*

[Korning, 1994] Korning, P. (1994). Training of neural networks by means of genetic algorithm working on very long chromosomes. *Tech. Report, Computer Science Department, Aarhus C, Denmark.*

[Koza and Rice, 1991] Koza, J. and Rice, J. (1991). Genetic generation of both the weights and architecture for a neural network. *in Proc. of 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Seattle), vol. 2, pp. 397-404, IEEE Press, ew York.*

[Ku and Mak, 1997] Ku, K. and Mak, M. (1997). Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks. *In Proceedings of 1997 IEEE International Conference on Evolutionary Computation, pp. 159-163. IEEE.*

[Lamarck, 1809] Lamarck, J. (1809). Philosophie zoologique.

[Land, 1998] Land, M. (1998). Evolutionary algorithms with local search for combinatorial optimization. *PhD thesis, Computer Science and Engr. Dept. - Univ. California. San Diego.*

[Lang et al., 1990] Lang, K., Waibel, A., and Hinton, G. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks, vol. 3, pp. 33-43.*

[Lee, 1996] Lee, S. (1996). Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 6, pp. 648-652.*

[Lee et al., 1990] Lee, T., Peterson, A., and Tsai, J. (1990). A multilayer feed-foward neural network with dynamically adjustable structures. *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Los Angeles, pp. 367-369. Long Beach, CA: IEEE Press.*

[Lee et al., 1993] Lee, Y., Oh, S., and Kim, M. (1993). An Analysis of Premature Saturation in Back-Propagation Learning. *Neural Networks, vol. 6, pp. 719-728.*

[Levenick, 1991] Levenick, J. (1991). Inserting introns improves genetic algorithm success rate: taking a cue from biology. *In Procs. of the 4th. International Conference on Genetic Algorithms, ICDA'91. Morgan Kaufmann.*

[Liu and Yao, 1996a] Liu, Y. and Yao, X. (1996a). Evolutionary design of artificial neural networks with different nodes. *in Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96), Nagoya,*

Japan, pp. 670-675, IEEE Press, New York, NY 10017-2394.

[Liu and Yao, 1996b] Liu, Y. and Yao, X. (1996b). A population-based learning algorithm which learns both architectures and weights of neural networks. *Chinese Journal of Advanced Software Research (Allerton Press, Inc., New York, NY 10011), vol. 3, no. 1, pp. 54-65.*

[Lovell and Tsoi, 1992] Lovell, D. and Tsoi, A. (1992). The performance of the neocognitron with various S-cell and C-cell transfer functions. *Intelligent Machenes Lab., Dept. of Elec. Eng., Univ. of Queensland.*

[Mani, 1990] Mani, G. (1990). Learning by gradient descent in function space. *in Proc. of IEEE int'l Conf. on System, Man, and Cybernetics, (Los Angeles, CA), pp. 242-247.*

[Marín, 1997] Marín, F. (1997). Optimización de redes neuronales artificiales mediante algoritmos genéticos. aplicación a la predicción de carga. *Tesis Doctoral, Universidad de Málaga. Escuela Técnica Superior de Ingeniería Informática. Málaga, España.*

[Marín and Sandoval, 1993] Marín, F. and Sandoval, F. (1993). Genetic synthesis of discrete-time recurrent neural network. *Lecture Notes in Computer Science, Vol. 686, pp.179-184, Springer-Verlag.*

[Marín and Sandoval, 1995] Marín, F. and Sandoval, F. (1995). Diseño de redes neuronales artificiales mediante algoritmos genéticos. *Computación Neuronal. Universidad de Santiago de Compostela, pp. 385-424.*

[Martí, 1992] Martí, L. (1992). Genetically generated neural networks i: representational effects. *in Proc. of Int'l Joint Conf. on Neural Networks (IJCNN'92 Baltimore), Vol. IV, pp. 537-542, IEEE Press, New York NY 10017-2394.*

[Merelo et al., 1993a] Merelo, J., Patón, M., Canas, A., Prieto, A., and Morán, F. (1993a). Genetic optimization of a multilayer neural network for cluster classification tasks. *Neural Network World, vol.3, pp.175-186.*

[Merelo et al., 1993b] Merelo, J., Patón, M., Canas, A., Prieto, A., and Morán, F. (1993b). Optimization of a competitive learning neural network by genetic algorithms. *Lecture Notes in Computer Science, Vol. 686, pp. 185-192, Springer-Verlag.*

[Merelo and Prieto, 1995] Merelo, J. and Prieto, A. (1995). G-LVQ, a combination of genetic algorithms and LVQ. *in Artificial Neural Nets and Genetic Algorithms, D.W. Pearson, N.C. Steele and R.F. Albrecht, Edts., pp. 92-95, Springer-Verlag, ISBN 3-211-82692-0.*

[Merelo et al., 1998] Merelo, J., Prieto, A., Morán, F., Marabini, R., and Carazo, J. (1998). Automatic classification of biological particles from electron-microscopy images using conventional and genetic-algorithm optimized learning vector quantization. *Neural Processing Letters, vol.8, pp.55-65.*

[Merz and Freisleben, 1997] Merz, P. and Freisleben, B. (1997). Genetic local search for the TSP: New results. *In Proceedings of 1997 IEEE International Conference on Evolutionary Computation, pp. 159-163. IEEE.*

[Mézard and Nadal, 1989] Mézard, M. and Nadal, J. (1989). Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A 22:2191-2203.*

[Michalewicz, 1992] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag.

[Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs , Third, Extended Edition.* Springer-Verlag.

[Miller et al., 1989] Miller, G., Todd, P., and Hegde, S. (1989). Designing neural networks using genetic algorithms. *In J.D.Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 379-384, San Mateo, 1989.*

[Mjolsness et al., 1989] Mjolsness, E., Sharp, D., and Alpert, B. (1989). Scaling, machine learning, and genetic neural nets. *Advances in Applied Mathematics, vol. 10, pp. 137-163.*

[Montana and Davis, 1989] Montana, D. and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. *Proc.*

11th Internat. Joint Conf. on Artificial Intelligence, 762-767.

[Muhlenbein, 1988] Muhlenbein, H. (1988). Adaptation in open systems: learning and evolution. *in Workshop Konnektionismus (J. Kindermann and C. Lischka, eds.), pp. 122-130, GMD, Postfach 1240, D-5205 St., Augustin, Germany.*

[Muhlenbein and Kindermann, 1989] Muhlenbein, H. and Kindermann, J. (1989). The dinamics of evolution and learning - towards genetic neural networks. *in Connectionism in Perspective (R. Pfifer et al., ed.), pp. 173-198, Elsevier Science Publishers B.V., Amsterdam.*

[Nguyen and Widrow, 1990] Nguyen, D. and Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *In Proceedings of the International Joint Conference on Neural Networks (IJCNN) San Diego, vol. III, pp. 21-26, Edward Brothers.*

[Nolfi et al., 1990] Nolfi, S., Elmann, J., and Parisi, D. (1990). Learning and evolution in neural networks. *Tech. Rep. CRT-9019, Center for Research in Language, University of California, San Diego, La Jolla, CA 92093-0126, USA.*

[Nolfi and Parisi, 1992] Nolfi, S. and Parisi, D. (1992). Growing neural networks. *Technical Report PCIA-91-15, Institute of Psychology, CNR, Rome, 1991. Also in Proceedings of ALIFE III.*

[Nordin and Banzhaf, 1995] Nordin, P. and Banzhaf, W. (1995). Complexity compression and evolution. *In Procs. of the 6th. International Conference on Genetic Algorithms, ICGA'95, pp. 310-317. Morgan Kaufmann.*

[Oliveira et al., 1999] Oliveira, M., Barreiros, J., Costa, E., and Pereira, F. (1999). LamBaDa: An Artificial Environment to Study the Interaction beween Evolution and Learning. *In Congress on Evolutionary Computation, Volume I, pp. 145-152, Washington D.C., USA.*

[Omatu and Deris, 1996] Omatu, S. and Deris, S. (1996). Stabilization of inverted pendulum by the genetic algorithm. *in Proceedings of the 1996 IEEE Conference on Emerging Technologies and Factory Automation, ETFA'96. Part 1 (of 2), (Piscataway, NJ, USA), pp. 282-287, IEEE Press.*

[Omatu and Yoshioka, 1997] Omatu, S. and Yoshioka, M. (1997). Self-tuning neuro-pid control and applications. *in Proceedings of the 1997 IEEE Conference on Systems, Man and Cybernetics. Part 3 (of 5), (Piscataway, NJ, USA), pp. 1985-1989, IEEE Press.*

[Osmera, 1995] Osmera, P. (1995). Optimization of neural networks by genetic algorithms. *Neural Network World, vol. 5, no. 6, pp. 965-976.*

[Paredis, 1991] Paredis, J. (1991). The evolution of behavior: some experiments. *in Proc. of the First Int'l Conf. on Simulation of Adaptive Behavior: From Animals to Animats (J. Meyer and S.W. Wilson, eds.), MIT Press, Cambridge, MA.*

[Patel, 1996] Patel, D. (1996). Using genetic algorithms to construct a network for financial prediction. *in Proceedings of SPIE: Applications of Artificial Neural Networks in Image Processing, (Bellingham, WA, USA), pp. 204-213, Society of Photo-Optical Instrumentation Engineers.*

[Pelillo and Fanelli, 1993] Pelillo, M. and Fanelli, A. (1993). A Method of Pruning Layered Feed-Forward Neural Networks. *Lecture Notes in Computer Science, Vol. 686, pp. 278-283, Springer-Verlag.*

[Porto and Fogel, 1990] Porto, V. and Fogel, D. (1990). Neural network techniques for navigation of auvs. *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology (pp. 137-141). Washington, DC: IEEE.*

[Porto et al., 1995] Porto, V., Fogel, D., and Fogel, L. (1995). Alternative neural network training methods. *IEEE Expert, vol. 10, no. 3, pp. 16-22.*

[Prados, 1992a] Prados, D. (1992a). New learning algorithm for training multilayered neurla networks that uses genetic-algorithm techniques. *Electronics Letters, vol. 28, pp. 1560-1561.*

[Prados, 1992b] Prados, D. (1992b). Training multilayered neural networks by replacing

the least fit hidden neurons. *in Proc. of IEEE SOUTHEASTCON '92, vol. 2, pp. 634-637, IEEE Press, New York, NY.*

[Prechelt, 1994] Prechelt, L. (1994). PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany.

[Radcliffe, 1991a] Radcliffe, N. (1991a). Equivalence class analysis of genetic algorithms. *Complex Systems 5, no.2: 183-205.*

[Radcliffe, 1991b] Radcliffe, N. (1991b). Genetic set recombination and its applicaton to neural network topology optimization. *Tech. Report EPCC-TR-91-21, University of Edinburgh, Edinburgh, Scotland.*

[Ragg et al., 1995] Ragg, T., Gutjahr, S., and Sa, H. (1995). Automatic determination of optimal network topologies based on information theory and evolution. *in IEEE, Proceedings of the 23rd Euromicro Conference, Track on Computational Intelligence, pp.549-555.*

[Reed, 1993] Reed, R. (1993). Pruning algorithms – a survey. *IEEE Transactions on Neural Networks, 4(5): 740-744.*

[Reed and Marks, 1999] Reed, R. and Marks, R. (1999). *Neural Smithing.* Bradford. The MIT Press, Cambridge, Massachusetts, London, England.

[Renders and Flasse, 1996] Renders, J. and Flasse, S. (1996). Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, Vol.26, No.2, pp.243-258.*

[Ribert et al., 1994] Ribert, A., Stocker, E., Lecourtier, Y., and Ennaji, A. (1994). Optimizing a Neural Network Architecture with an Adaptive Parameter Genetic Algorithm. *Lecture Notes in Computer Science, Vol. 1240, pp. 527-535, Springer-Verlag.*

[Rivas et al., 1999] Rivas, V., Merelo, J., Rojas, I., Castillo, P., Carpio, J., and Romero, G. (1999). Evolving 2-Dimensional Fuzzy Logic Controllers. *Submitted to Fuzzy Sets and Systems. Second revision.*

[Roberts and Turega, 1995] Roberts, S. and Turega, M. (1995). Evolving neural networks: an evaluation of enconding techniques. *in Artificial Neural Nets and Genetic Algorithms, D.W. Pearson, N.C. Steele and R.F. Albrecht, Edts., pp. 96-99, Springer-Verlag, ISBN 3-211-82692-0.*

[Rojas et al., 1997] Rojas, I., Merelo, J., Pomares, H., and Prieto, A. (1997). Genetic algorithms for optimum designing of fuzzy controllers. *In Proceedings of the 2nd. Int. ICSC Symposium on Fuzzy Logic and Applications (ISFL'97), pp.165-170. International Computer Science Conventions, ICSC Academic Press.*

[Ross, 1999] Ross, B. (1999). A lamarckian evolution strategy for genetic algorithms. *In Lance D. Chambers, editor, Practical Handbook of Genetic Algorithms: Complex Coding Systems, volume III, pp. 1-16. Boca Raton, FL: CRC Press.*

[Rumelhart, 1985] Rumelhart, D. (1985). Feature discovery by competitive learning. *Cognitive Science (9).*

[Rumelhart and McClelland, 1986] Rumelhart, D. and McClelland, J. (1986). Parallel Distributed Processing: Explorations in the Microstructures of Cognition. *Cambridge, MA: MIT Press.*

[Saravanan and Fogel, 1995] Saravanan, N. and Fogel, D. (1995). Evolving neural control systems. *IEEE Expert, vol. 10, no. 3, pp. 23-27.*

[Sarkar and Yegnanarayana, 1997] Sarkar, M. and Yegnanarayana, B. (1997). Evolutionary programming-based probabilistic neural networks construction technique. *in Proceedings of the 1997 IEEE International Conference on Neural Networks. Part 1 (of 4), (Piscataway, NJ, USA), pp. 456-461, IEEE Press.*

[Schaffer et al., 1990] Schaffer, J., Caruana, R., and Eshelman, L. (1990). Using genetic search to exploit the emergent behavior of neural networks. *Physica D, vol. 42, pp. 244-248.*

[Schiffmann et al., 1990] Schiffmann, W., Joost, M., and Werner, R. (1990). Performance evaluation of evolutionarily created neural network topologies. *In H.P. Schwefel*

and R. Männer, editors, Parallel Problem Solving from Nature, pp. 274-283. Springer.

[Schiffmann et al., 1992] Schiffmann, W., Joost, M., and Werner, R. (1992). Synthesis and performance analysis of multilayer neural network architectures. *Tech. Rep. 16/1992, University of Koblenz, Institute für Physics, Rheinau 3-4, D-5400 Koblenz.*

[Sexton et al., 1998] Sexton, R., Dorsey, R., and Johnson, J. (1998). Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems, vol. 22, no. 2, pp. 171-185.*

[Skinner and Broughton, 1995] Skinner, A. and Broughton, J. (1995). Neural networks in computational materials science: Training algorithms. *Modelling and Simulation in Materials Science and Engineering, 3:371-390.*

[Smieja, 1991] Smieja, F. (1991). Hyperplane spin dynamics, network plasticity and backpropagation learning. *GMD report, GMD, St. Augustin, Germany.*

[Srivinas and Patnaik, 1991] Srivinas, M. and Patnaik, L. (1991). Learning neural network weights using genetic algorithms - improving preformance by search-space reduction. *in Proc. of 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Singapore), vol. 3, pp. 2331-2336, IEEE Press, New York.*

[Stork et al., 1990] Stork, D., Walker, S., Burns, M., and Jackson, B. (1990). Preadaptation in neural circuits. *in Proc. of Int'l Joint Conf. on Neural Networks, Vol. I, (Washington, DC), pp. 202-205, Lawrence Erlbaum Associates, Hillsdale, NJ.*

[Sutton, 1986] Sutton, R. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. *In Proceedings of the 8th Annual Conference of the Cognitive Science Society, pp.823-831. Erlbaum, Hillsdale, NJ.*

[Tang et al., 1995] Tang, K., Chan, C., Man, K., and Kwong, S. (1995). Genetic structure for NN topology and weights optimization. *in Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), (Stevenage, England), pp. 250-255, IEE Conference Publication 414.*

[Thierens et al., 1993] Thierens, D., Suykens, J., Vandewalle, J., and Moor, B. D. (1993). Genetic weight optimization of a feedforward neural network controller. *In Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms, pp. 658-663. Springer-Verlag.*

[Thimm and Fiesler, 1995] Thimm, G. and Fiesler, E. (1995). Neural network initialization. *Lecture Notes in Computer Science, Vol. 930, pp. 535-542, Springer-Verlag.*

[Topchy and Lebedko, 1997] Topchy, A. and Lebedko, O. (1997). Neural network training by means of cooperative evolutionary search. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 389, no. 1-2, pp. 240-241.*

[Topchy et al., 1996] Topchy, A., Lebedko, O., and Miagkikh, V. (1996). Fast learning in multilayered neural networks by means of hybrid evolutionary and gradient algorithms. *to appear in Proc. of IC on Evolutionary Computation and Its Applications, Moscow.*

[Turney et al., 1996] Turney, P., Whitley, D., and Anderson, R. (1996). Special issue on the baldwin effect. *Evolutionary Computation, vol. 4, no. 3, pp. 213-329.*

[Utrecht and Trint, 1994] Utrecht, U. and Trint, K. (1994). Mutation operators for structure evolution of neural networks. *In R. Maenner Y. Davidor, H.P. Schwefel, editors, Parallel Problem Solving from Nature, Workshop-Proceedings, pp. 492-501. Springer.*

[Valderrábano et al., 2001] Valderrábano, J., Madinaveitia, E., Castillo, P., and Merelo, J. (2001). NOTORIEDAD Y PRESIÓN PUBLICITARIA: Pueden los nuevos métodos matemáticos ayudarnos a mejorar la eficacia de la publicidad? *Ponencias del 96 Seminario. 17 Seminario de Televisión, pp.149-159. AEDEMO. Jerez de la Frontera, 7 al 9 de Febrero.*

[van Wanrooij, 1994] van Wanrooij, E. (1994). Evolving squential neural networks for time

series forecasting. *Master's thesis, Department of Computer Science, University of Utrecht, Netherlands.*

[Vonk et al., 1995] Vonk, E., Jain, L., and Johnson, R. (1995). Using genetic algorithms with grammar encoding to generate neural networks. *in Proceedings of the 1995 IEEE International Conference on Neural Networks. Part 4 (of 6), (Piscataway, NJ, USA), pp. 1928-1931, IEEE Press.*

[Waddington, 1942] Waddington, C. (1942). Canalization of development and the inheritance of acquired characteristics. *Nature, 3811, pp. 563-565.*

[Weller et al., 1995] Weller, P., Summers, R., and Thompson, A. (1995). Using a genetic algorithm to evolve an optimum input set for a predictive neural network. *in Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), (Stevenage, England), pp. 256-258, IEE Conference Publication 414.*

[Wessels and Barnard, 1992] Wessels, L. and Barnard, E. (1992). Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks, vol. 3, num. 6, pp. 899-905.*

[White and Ligomenides, 1993] White, D. and Ligomenides, P. (1993). GANNet: A Genetic Algorithm for Optimizing Topology and Weights in Neural Network Design. *Lecture Notes in Computer Science, Vol. 686, pp. 322-327, Springer-Verlag.*

[Whitley, 1989a] Whitley, D. (1989a). *The GENITOR Algorithm and Selection Presure: Why rank-based allocation of reproductive trials is best.* in J.D. Schaffer (Ed.), Proceedings of The Third International Conference on Genetic Algorithms, Morgan Kauffmann, Publishers, 116-121.

[Whitley et al., 1994] Whitley, D., Gordon, V., and Mathias, K. (1994). Lamarckian Evolution, The Baldwin Effect and Function Optimization. *Parallel Problem Solving from Nature-PPSN III. Y. Davidor, H.P. Schwefel and R. Manner, eds. pp. 6-15. Springer-Verlag.*

[Whitley et al., 1990] Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing, vol. 14, no. 3, pp. 347-361.*

[Whitley, 1989b] Whitley, L. (1989b). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *In J.D. Schaffer, ed., Proceedings of the Third International Conference on Genetic Algotirhms. Morgan Kaufmann.*

[Wieland, 1991] Wieland, A. (1991). Evolving neural network controllers for unstable systems. *in Proc. of 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Seattle), vol. 2, pp. 667-673, IEEE Press, New York.*

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82.*

[Wu and Lindsay, 1995] Wu, A. and Lindsay, R. (1995). Empirical studies of the genetic algorithm with non coding segments. *Evolutionary Computation, 3(2).*

[Yao, 1991] Yao, X. (1991). Evolution of evolutionary artificial neural networks. *in Preprints of the Int'l Symp. on AI, Reasoning and Creativity (T. Dartnall, ed.), (Queensland, Australia), pp. 49-52, Griffith University.*

[Yao, 1993a] Yao, X. (1993a). Evolutionary artificial neural networks. *International Journal of Intelligent Systems, vol. 4, no. 3, pp. 203-222.*

[Yao, 1993b] Yao, X. (1993b). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems, vol. 8, no. 4, pp. 539-567.*

[Yao, 1994] Yao, X. (1994). The evolution of connectionist networks. *in Artificial Intelligence and Creativity (T. Dartnall, ed.), pp. 233-243, Dorcrecht: Kluwer Academic Publishers.*

[Yao, 1995] Yao, X. (1995). Evolutionary artificial neural networks. *in Encyclopedia of Computer Science and Technology (A. Kent and J.G. Williams, eds.), vol. 33, pp. 137-170, New York, NY 10016: Marcel Dekker Inc.*

[Yao, 1999] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87(9):1423-1447.*

[Yao et al., 1997] Yao, X., Lin, G., and Liu, Y. (1997). An analysis of evolutionary algorithms based on neighbourhood and step sizes. *in Evolutionary Programming VI: Proc. of the Sixth Annual Conference on Evolutionary Programming (P.J. Angeline, R.G. Reynolds, J.R. McDonnell and R. Eberhart, eds.), vol. 1213 of Lecture Notes in Computer Science, (Berlin), pp. 297-307, Springer-Verlag.*

[Yao and Liu, 1996a] Yao, X. and Liu, Y. (1996a). Ensemble structure of evolutionary artificial neural networks. *in Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96), Nagoya, Japan, pp. 659-664, IEEE Press, New York, NY 10017-2394.*

[Yao and Liu, 1996b] Yao, X. and Liu, Y. (1996b). Evolutionary artificial neural networks that learn and generalise well. *in 1996 IEEE International Conference on Neural Networks, Washington DC, USA, Volume on Plenary, Panel and Special Sessions, pp. 159-164, IEEE Press, New York.*

[Yao and Liu, 1996c] Yao, X. and Liu, Y. (1996c). Evolving artificial neural networks through evolutionary programming. *in Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming (L.J. Fogel, P.J. Angeline, and T. Back, eds.), (Cambridge, MA), pp. 257-266, The MIT Press.*

[Yao and Liu, 1996d] Yao, X. and Liu, Y. (1996d). Fast evolutionary programming. *in Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming (L.J. Fogel, P.J. Angeline, and T. Back, eds.), (Cambridge, MA), pp. 451-460, The MIT Press.*

[Yao and Liu, 1997a] Yao, X. and Liu, Y. (1997a). Epnet for chaotic time-series prediction. *in Selected Papers from the First Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'96)(S. Yao, J.H. Kim and T. Furuhashi, eds.), vol. 1285 of Lecture Notes in Artificial Intelligence, (Berlin), pp. 146-156, Springer-Verlag.*

[Yao and Liu, 1997b] Yao, X. and Liu, Y. (1997b). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 694-713.*

[Yao and Liu, 1998a] Yao, X. and Liu, Y. (1998a). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 28, no. 3, pp. 417-425.*

[Yao and Liu, 1998b] Yao, X. and Liu, Y. (1998b). Towards Designing Artificial Neural Networks by Evolution. *Applied Mathematics and Computation, 91(1):83-90.*

[Yao and Shi, 1995] Yao, X. and Shi, Y. (1995). A preliminary study on designing artificial neural networks using coevolution. *in Proc. of the IEEE Singapore Int'l Conf. on Intelligent Control and Instrumentation, (Singapore), pp. 149-154, IEEE Singapore Section.*

[Yoon et al., 1994] Yoon, B., Holmes, D., and Langholz, G. (1994). Efficient genetic algorithms for training layered feedforward neural networks. *Information Sciences, vol. 76, no. 1-2, pp. 67-85.*