Evolución de gramáticas bidimensionales de contexto libre para el diseño de arquitecturas de redes de neuronas artificiales

Miguel A. Guinea, Araceli Sanchis, José M. Molina

Departamento de Informática.
Universidad Carlos III de Madrid.
Avda. de la Universidad, 30
Leganés, Madrid, 28911
miguelangel.cab@teleline.es, masm@inf.uc3m.es, molina@ia.uc3m.es

Resumen

Uno de los pasos decisivos en la aplicación de Redes de Neuronas Artificiales es el diseño de la arquitectura de la red; sin embargo, el diseño de la arquitectura es una tarea de expertos. Por ello el desarrollo de métodos automáticos de determinación de arquitecturas resulta un campo de gran interés entre los investigadores en redes de neuronas artificiales. Estos métodos utilizan fundamentalmente técnicas de búsqueda, como Algoritmos Genéticos, Enfriamiento Simulado, Estrategias Evolutivas o incluso Técnicas Multiagente. La mayoría de estos métodos se basan en la representación directa de los parámetros de la red, lo que hace que sean métodos no escalables, es decir, para representar arquitecturas moderadamente complejas se requieren codificaciones de gran tamaño, lo que afecta al tiempo de convergencia del algoritmo considerado. En este trabajo se propone un esquema de codificación indirecta que se basa en la representación mediante gramáticas bidimensionales. El esquema propuesto, denominado sistema GANET, es genérico y supera las limitaciones de otros sistemas relacionados. El sistema GANET ha demostrado su capacidad para encontrar arquitecturas óptimas para el problema de aproximación de la serie temporal logística.

Palabras clave: Redes de Neuronas, Algoritmos Genéticos, Generación Automática de Arquitecturas.

1. Introducción

Una de las principales claves en la aplicación eficiente de redes de neuronas artificiales es el diseño de la arquitectura de la red. Este diseño es, todavía hoy, un proceso guiado por el conocimiento del experto sobre el problema. Este experto se basa en su experiencia y en un largo proceso de prueba y error para decidir cuál es la mejor arquitectura para un problema dado. El diseño de la arquitectura de una red de neuronas se puede ver como un problema de búsqueda en el espacio de las arquitecturas, donde cada punto del espacio de búsqueda es una arquitectura determinada. Evidentemente, se trata de una tarea larga y tediosa.

Desde hace algunos años, muchos trabajos se han centrado en la resolución automática del diseño de arquitecturas de redes de neuronas artificiales (Harp 1989), (Miller 1989), (Harp 1990), (Gruau 1992), (Gruau 1994) y (Gruau 1995), entre otros. Muchos de estos trabajos utilizan técnicas de computación evolutiva como elemento fundamental de la búsqueda (Harp et al. 1989) y (Miller et al. 1989). Principalmente existen dos perspectivas para aplicar un procedimiento de computación evolutiva para el aprendizaje de la arquitectura de red óptima: una basada en la representación completa de todas las conexiones posibles y otra basada en una representación indirecta de la arquitectura. La

primera de ellas es la aproximación denominada Direct Encoding Method, y se basa en la codificación directa de la red completa (matriz de conexiones) en la estructura a evolucionar mediante la correspondiente técnica de computación evolutiva, el cromosoma de un Algoritmo Genético en la idea original. Esta línea comenzó con el trabajo de Ash (Ash 1988) y continuó con otros trabajos, incluyendo (Fogel et al. 1990) y (Caudell et al. 1989). Otras líneas de trabajo dentro del Direct Encoding Method han tratado de introducir en el proceso evolutivo el proceso de adaptación de la red (Schaffer et al. 1990), (Alba et al.1993), (Mjolsness et al. 1989).

La segunda aproximación, denominada Indirect Encoding Method, consiste en codificar, no la red completa, sino una representación reducida de la misma (Harp et al. 1989). En 1990, Kitano (Kitano 1990) presentó un método novedoso para el diseño de redes de neuronas artificiales mediante Algoritmos Genéticos, donde las redes se representaban mediante gramáticas bidimensionales que se encontraban codificadas en el cromosoma. La aproximación basada en gramáticas no es la única que existe, cabe destacar la representación fractal de Merril y Port (Merril et al. 1991) que se fundamenta en la idea de que esta representación está más directamente relacionada con las ideas biológicas que los algoritmos meramente constructivos, y la aplicación de autómatas celulares a la generación de la arquitectura (Galván et al. 2000), (Gutierrez et al. 2001). Otra aproximación que resulta interesante es la propuesta de (Valls et al. 2000), (Molina et al. 2002) donde una arquitectura multiagente da soporte a un conjunto de agentes redes de base radial de manera que la coordinación entre los agentes guía la búsqueda de la red óptima para un problema dado.

La propuesta de (Kitano 1990) representa las redes de neuronas artificiales mediante gramáticas bidimensionales, Grammar Encoding Method, que se codifican en los cromosomas de los individuos. En el trabajo de Kitano, una red de neuronas se representa como una matriz de ceros y unos, donde un valor 1 en la posición (x,y) de la matriz, representa que la neurona ‘x’ está conectada con la ‘y’; y un valor 0 indica que las dos neuronas no están conectadas. A partir de esta forma de representar a la red de neuronas, se estudian y comparan la codificación directa y la indirecta en el cromosoma del algoritmo genético. Una de las principales conclusiones de ese artículo es la demostración de que el método indirecto es más eficiente que el directo. También se demuestra que mediante la codificación de la gramática que genera el lenguaje al que pertenece la matriz de conexiones, se obtiene una mejor convergencia, en términos de tiempo/ciclos. Más aún, los cromosomas en el método indirecto son de menor tamaño que los empleados en la codificación directa. Sin embargo, a pesar de las evidentes ventajas y aportaciones del trabajo de Kitano, aparecen algunas limitaciones en el método por él propuesto, principalmente relacionadas con la codificación empleada. Esta codificación produce palabras siempre del mismo tamaño. Al fijar el número de nodos de la red, el tamaño de la palabra debe ser constante, aunque algunos nodos pueden o no aparecer en la red, dependiendo de sí quedan conectados o no, es decir que la matriz para una red de “n” neuronas tiene un tamaño de “m x n” (donde m es la potencia más pequeña de 2 mayor que “n”) y de esta matriz sólo se utiliza el triángulo superior. Esta utilización de matrices de tamaño fijo implica restricciones en las gramáticas empleadas, haciendo que no puedan ser recursivas, y en los cromosomas, determinando el tamaño de la matriz o palabra. Para fijar el tamaño y limitar la recursividad, Kitano incluye en el cromosoma una parte variable y una parte fija. Además, introduce un operador de sobrecruzamiento restringido a la parte variable del cromosoma, quedando la parte fija inalterable, para cada cromosoma, durante todo el proceso. Estas limitaciones afectan al método, haciéndolo poco general.

En trabajos anteriores (Molina et al. 2000a), (Molina et al. 2000b), (Guinea et al. 2001), el principal objetivo ha sido la extensión y la mejora del método de Kitano, haciéndolo más general, con el fin de ampliar la potencialidad del mismo. Para ello, se ha desarrollado, en primer lugar un mecanismo de representación de redes de neuronas artificiales mediante gramáticas, sin restricciones respecto al tamaño de las palabras y empleando gramáticas recursivas (Molina et al. 2000a). Dicho mecanismo se integra en el sistema completo GANET, para el diseño de redes de neuronas artificiales aplicadas a la resolución de un problema dado. Las redes que se diseñarán serán perceptrones multicapa con una capa oculta y entrenadas con el algoritmo de retropropagación (Hertz et al. 1991).

En el presente artículo se presenta un método general para el diseño de redes de neuronas basado en gramáticas bidimensionales. El método desarrollado permite la codificación completa de una gramática bidimensional en un cromosoma, de manera que se evitan las limitaciones de métodos anteriores. El conjunto de cromosomas se evolucionará mediante el sistema GANET. Además de presentar el método, por un lado, se analiza la capacidad de generación de la codificación gramatical, y, por otro, se muestra un ejemplo de aplicación a la predicción de series temporales.

2. Gramáticas bidimensionales de contexto libre

En este trabajo se han empleado gramáticas de contexto libre (Chomsky 1959), (Hopcroft y Ullman 1979), pero bidimensionales (Ehring et al. 1987). Este tipo de gramáticas se describen mediante una quíntupla, G = {Σ_T, Σ_N, S, P, EM}, donde:

- Σ_T = {0,1} es el alfabeto de símbolos terminales
- Σ_N = {A,B,C,D} es el alfabeto de símbolos no terminales
- S = axioma, S $\in \Sigma_N$
- P = conjunto de reglas de producción = { A::= $\begin{pmatrix} B & D \\ D & D \end{pmatrix}$, B::=$\begin{pmatrix} 0 & C \\ 0 & 0 \end{pmatrix}$, C::=$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, D::=$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$ }
- EM = Método de expansión = {(UL) arriba e izquierda, (UR) arriba y derecha, (DL) abajo e izquierda, (DR) abajo y derecha}.

Para generar las palabras a partir de la definición de la gramática, de manera que sean adecuadas para el sistema Ganet, se ha desarrollado un procedimiento de derivación especial, que asegura la rectangularidad de las palabras durante el proceso de derivación, mediante la inserción de símbolos auxiliares que permiten, en su caso, la inserción del espacio necesario para, posteriormente, producir la derivación.

Para poder llevar a cabo el proceso de derivación en una gramática de contexto libre bidimensional, y generar palabras que conserven la rectangularidad es necesario insertar en algunos casos una fila, en otros una columna, en otros una fila y una columna y en otros no es necesario insertar espacio alguno. Respecto al lugar donde se debe insertar el espacio mencionado, éste dependerá tanto de la posición del símbolo a derivar como del modo de expansión elegido. Cuando el método de expansión es Arriba y Derecha la fila a insertar se situará encima del símbolo a derivar y la columna a insertar se situará a la derecha del símbolo a derivar. En la regla de producción mostrada en la Figura 1 se realiza primero la inserción del espacio necesario (fila y columna) y, a continuación, se deriva el símbolo no terminal deseado, en este caso B.

$$A::=\begin{pmatrix} B & D \\ D & D \end{pmatrix} \rightarrow \begin{pmatrix} * & * & * \\ B & * & D \\ D & * & D \end{pmatrix} \rightarrow \begin{pmatrix} 0 & C & * \\ 0 & 0 & D \\ D & * & D \end{pmatrix}$$

Figura 1. Derivación del símbolo no terminal B en una gramática bidimensional, empleando el método de expansión UR.

Además, en el mecanismo propuesto, la derivación de los distintos símbolos no terminales se lleva a cabo "por niveles"; es decir, se derivan en primer lugar todos los símbolos no terminales de un nivel (aquellos que proceden de una misma derivación) y a continuación los generados en la primera de las derivaciones del nivel anterior, luego los generados en la segunda, etc. y dentro de un mismo nivel dependerán del modo de expansión considerado. De esta manera se asegura que el número de símbolos auxiliares insertado es mínimo. Así, para obtener una palabra, se continuará la derivación por niveles de la palabra representada en la Figura 1, cuyo resultado se recoge en la Figura 2.

$$\text{.....} \rightarrow \begin{pmatrix} 0 & C & * & * \\ 0 & 0 & D & * \\ D & * & D & * \end{pmatrix} \rightarrow \begin{pmatrix} 0 & C & 0 & 0 \\ 0 & 0 & 1 & 1 \\ D & * & D & * \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 0 & C & 0 & 0 \\ 0 & 0 & 1 & 1 \\ * & * & * & * \\ D & * & D & * \end{pmatrix} \rightarrow \begin{pmatrix} 0 & C & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & * & * \\ 1 & 1 & D & * \end{pmatrix} \rightarrow \begin{pmatrix} 0 & C & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} * & * & * & * & * \\ 0 & C & * & 0 & 0 \\ 0 & 0 & * & 1 & 1 \\ 0 & 0 & * & 0 & 0 \\ 1 & 1 & * & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} * & 1 & 1 & * & * \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & * & 1 & 1 \\ 0 & 0 & * & 0 & 0 \\ 1 & 1 & * & 1 & 1 \end{pmatrix}$$

Figura 2. Obtención de una palabra en una gramática bidimensional, a partir de la palabra de la Figura 1.

Finalmente, para asegurar la obtención de una palabra (detención del proceso de derivación) será necesario considerar dos parámetros más en la gramática, por una parte el número de veces que puede aplicarse una regla de derivación recursiva y, por otra, el número máximo de niveles de derivación en los que se puede emplear reglas formadas por símbolos no terminales. Para ello, se define un valor máximo de recursividad y se aplica la derivación recursiva un número de veces hasta que se alcanza ese valor. A continuación, en la siguiente derivación, se aplicará alguna otra regla del mismo símbolo no terminal, diferente de la aplicada hasta el momento. Para el segundo caso, se define un número de niveles y, cuando éste se alcance, solamente se permitirá el uso de reglas de

derivación formadas por símbolos terminales que permiten la obtención de palabras. Es recomendable que las gramáticas sean bien formadas (Hopcroft 1979). Si no se trata de gramáticas bien formadas, será necesario eliminar las regla superfluas.

3. Arquitectura de GANET

La arquitectura del sistema propuesto, GANET, consta de tres módulos principales, siguiendo la idea propuesta por Kitano (Kitano 1990). Un esquema del sistema GANET se muestra en la Figura 3.

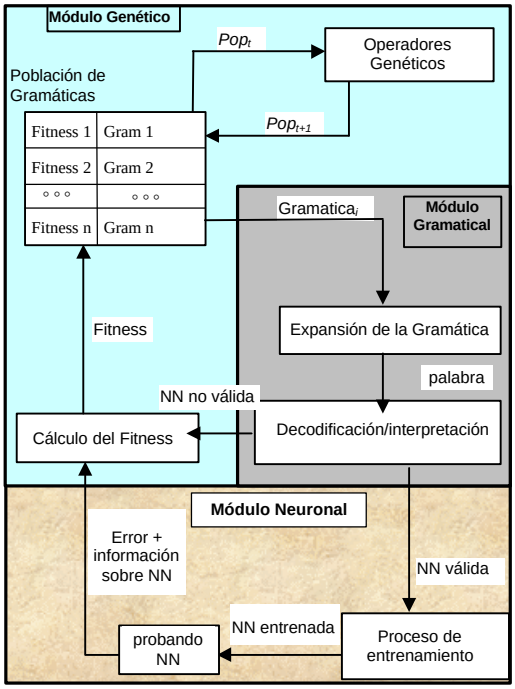

Figura 3. Esquema de la arquitectura del sistema GANET, con los módulos genético, neuronal y gramatical.

Dicho esquema es cíclico y cada ciclo corresponde a una generación del Algoritmo Genético. Mediante los operadores genéticos se obtiene una población de gramáticas, excepto la primera población que, como es usual, se genera aleatoriamente (Goldberg 1989). Los nuevos individuos de las poblaciones son evaluados mediante una función de fitness. Para calcular el fitness, cada gramática, codificada en un cromosoma o individuo, es expandida dando lugar a una matriz (palabra terminal de la gramática). Posteriormente, la palabra es decodificada y convertida en una matriz de conexiones de una red de neuronas. La red de neuronas obtenida se entrena y se evalúa con sendos conjuntos de entrenamiento y verificación. El valor de error resultante cometido por la red en el proceso de verificación, junto con alguna otra información relevante sobre la red de neuronas, como el tamaño de la red, los ciclos de aprendizaje, la actualización de los pesos, etc, es el utilizado como evaluación de cada individuo. El proceso se repite hasta que toda la población ha sido evaluada.

El tipo de redes de neuronas elegido en este trabajo es el perceptrón multicapa con conexiones hacia delante con aprendizaje *backpropagation*, permitiendo únicamente la existencia de una capa oculta. Esta restricción puede hacerse, sin pérdida de generalidad, puesto que no existe ninguna restricción respecto al número de neuronas de esta capa (Cybenko 1989). Para obtener un perceptrón multicapa a partir de la palabra obtenida en el módulo gramatical de GANET, se ha propuesto el siguiente método:

1. Los símbolos auxiliares se sustituyen por ceros. Un cero en la posición *(x,y)* de la matriz significará que las neuronas *x* e *y* no están conectadas, mientras que un 1 significará que sí existe conexión entre ambas neuronas.

2. Se divide la matriz en tres zonas, una por cada capa de la red. La primera zona corresponde a la capa de entrada y estará formada por el número de neuronas de la capa de entrada, que es fijo y dependiente de cada problema. La segunda zona corresponde a la capa de salida y estará formada por el número de neuronas de la capa de salida, que también viene dado para cada problema. La tercera zona corresponde a la capa oculta, y estará formada por el resto de neuronas, siendo este número variable a lo largo del problema y determinado por el tamaño de la palabra. Esta división en zonas de la matriz de conexiones se realiza tanto para las columnas de la matriz como para las filas, tal y como se ilustra en la Figura 4.

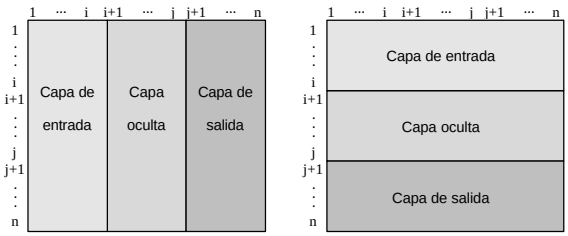

Figura 4. División en zonas (capas de la red) de la palabra generada en el módulo gramátical de GANET.

Así, las *i*-primeras columnas/filas de la matriz corresponderán a neuronas de la capa de entrada, las *(j-i)* siguientes a las neuronas de la capa oculta y, finalmente, las *(n-j)* últimas posiciones

corresponden a las neuronas de la capa de salida. Los valores de *i*, *j* estarán fijados por el problema a resolver y permanecerán constantes durante todo el proceso. El número de neuronas para cada capa no tiene por qué coincidir con el número final de neuronas que tiene la red generada, puesto que sólo se representarán en la red aquellas neuronas para las que exista conexión. Para decidir si existe o no una conexión entre dos neuronas, considérese la intersección de las dos divisiones que aparecen en la Figura 4 tal y como se muestran en la Figura 5. Cada una de las zonas de la Figura 5, representadas por una letra, corresponden a un tipo de conexiones diferente:

- Zona 'A': representa las conexiones de las neuronas de entrada entre sí.
- Zona 'B': representa las conexiones de las neuronas de la capa oculta entre sí.
- Zona 'C': representa las conexiones de las neuronas de salida entre sí.
- Zona 'D': representa las conexiones entre las neuronas de entrada y las neuronas de la capa oculta.
- Zona 'E': representa las conexiones entre las neuronas de entrada y las neuronas de salida.
- Zona 'F': representa las conexiones entre las neuronas de la capa oculta y las neuronas de salida.

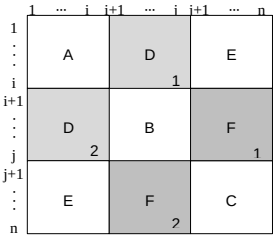

Figura 5. Conexiones entre las neuronas en la palabra generada en el módulo gramatical de GANET. Las zonas marcadas en gris corresponden a las conexiones válidas en un perceptrón multicapa.

Como las redes con las que se trabaja están formadas por tres capas, habrá dos tipos de conexiones: las que se establezcan entre la capa de entrada y la capa oculta, y las conexiones entre la capa oculta y la capa de salida. Cada uno de estos dos tipos de conexiones queda representado por una de las dos zonas mencionadas. La zona 'D' representa las conexiones de la capa de entrada con la capa oculta, mientras que la zona 'F' representa las conexiones de la capa oculta con la capa de salida. Respecto a la interpretación de las conexiones de la red, las zonas 'D' y 'F' abarcan, cada una, dos espacios de la matriz, tal y como se muestra en la Figura 5. Las zonas marcadas con un '1' representan conexiones hacia adelante. Así, la zona 'D_1' representa las conexiones que nacen en la capa de entrada y llegan a la capa oculta; y del mismo modo, la zona 'F_1' representa las conexiones que nacen en la capa oculta y llegan a la capa de salida. Y al contrario, las zonas marcada con un '2', representan conexiones hacia atrás, siendo estas conexiones las que hay que evitar, para poder obtener una red válida.

Para obtener directamente un perceptrón multicapa válido con tres niveles, bastaría con desechar las conexiones de las zonas 'D_2' y 'F_2'. No obstante, en este método se ha optado por emplear estas zonas pero interpretándolas de una forma especial, considerándolas como conexiones hacia adelante. Es decir, los valores binarios que contienen las zonas 'D' y 'F' solamente indican si existe o no conexión. Además, para que exista una conexión entre una neurona de la capa de entrada con otra de la capa oculta, debe reflejarse esta conexión tanto en 'D_1' como en 'D_2'. Del mismo modo se han considerado las conexiones entre la capa oculta y la capa de salida, así solamente se considerarán aquellas conexiones que queden reflejadas, tanto en la zona 'F_1' como en la zona 'F_2'.

En la Figura 6 se muestra un ejemplo de interpretación de la matriz para obtener una red de neuronas, asumiendo que la red tiene 5 neuronas de entrada y una de salida.

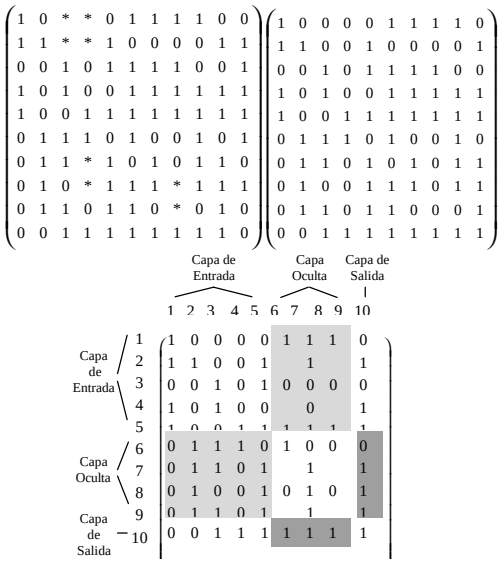

Figura 6. Ejemplo de interpretación de una matriz para obtener una red de neuronas.

El proceso seguido en la Figura 6 consiste, en primer lugar, en pasar de la palabra a una matriz cuadrada formada por 0's y 1's. Como la palabra

tiene 10 filas y 11 columnas, se elimina la última columna, con lo que queda una palabra cuadrada de tamaño 10x10. La necesidad de una matriz cuadrada viene impuesta por el significado de las filas y las columnas, al tener el mismo significado (ver Figura 6) debe haber tantas filas como columnas. A continuación, se sustituyen los símbolos auxiliares '*' por 0's. El siguiente paso, es separar de la matriz aquellas zonas que contienen información válida sobre las conexiones de la red. Si se destacan en color oscuro aquellas zonas que permiten obtener las conexiones, se obtiene la red representada en la Figura 6. Por último, hay que comprobar si existe conexión entre una neurona *'x'* y una neurona *'y'* (las entradas *(x,y)* e *(y,x)* de la matriz deben tener, ambas, el valor 1). Las zonas con un sombreado más claro, y de tamaño mayor, representan conexiones entre la capa de entrada y la capa oculta; y, las zonas con un sombreado más oscuro, y de tamaño menor, representan conexiones entre la capa oculta y la capa de salida. Interpretando las zonas se obtienen las conexiones de la red:

Entrada	Oculta	Valor Matriz 1	Valor Matriz 2	Conexión
'1'	'6'	(1,6)=1	(6,1)=0	no
'1'	'7'	(1,7)=1	(7,1)=0	no
'1'	'8'	(1,8)=1	(9,1)=0	no
'1'	'9'	(1,8)=1	(9,1)=0	no
'2'	'6'	(2,6)=0	(6,2)=1	no
'2'	'7'	(2,7)=0	(7,2)=1	no
'2'	'8'	(2,8)=0	(8,2)=1	no
'2'	'9'	(2,9)=0	(9,2)=1	no
'3'	'6'	(3,6)=1	(6,3)=1	sí
'3'	'7'	(3,7)=1	(7,3)=1	sí
'3'	'8'	(3,8)=1	(8,3)=0	no
'3'	'9'	(3,9)=0	(9,3)=1	no
'4'	'6'	(4,6)=1	(6,4)=1	sí
'4'	'7'	(4,7)=1	(7,4)=0	no
'4'	'8'	(4,8)=1	(8,4)=0	no
'4'	'9'	(4,9)=1	(9,4)=0	no
'5'	'6'	(5,6)=1	(6,5)=0	no
'5'	'7'	(5,7)=1	(7,5)=1	sí
'5'	'8'	(5,8)=1	(8,5)=1	sí
'5'	'9'	(5,9)=1	(9,5)=1	sí

Oculta	Salida	Valor Matriz 1	Valor Matriz 2	Conexión
'6'	'10'	(6,10)=0	(10,6)=1	no
'7'	'10'	(7,10)=1	(10,7)=1	sí
'8'	'10'	(8,10)=1	(10,8)=1	sí
'9'	'10'	(9,10)=1	(10,9)=1	sí

Con estas conexiones, se obtiene la red de neuronas mostrada en la Figura 7.

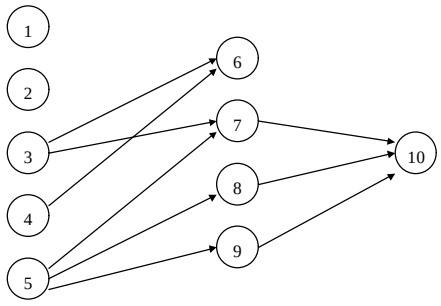

Figura 7. Red de neuronas interpretada a partir de la palabra de la Figura 6, en el sistema GANET.

4. Representación Genética de una Gramática Bidimensional

GANET trabaja con una población de gramáticas, así, la estructura del cromosoma de los individuos corresponderá a una codificación binaria de una gramática bidimensional, cuya expansión debe generar una palabra que se interpretará como una matriz de conexiones de una red.

Se trata, por lo tanto, de codificar la quíntupla que formaliza a toda gramática bidimensional dentro del cromosoma de los individuos. Respecto al alfabeto de símbolos terminales, ST, los símbolos terminales empleados son 0 y 1, por ser los únicos valores aceptados en la matriz de conexiones (0 para indicar que no existe conexión y un 1 para indicar la existencia de conexión; en caso de que la palabra contenga símbolos auxiliares estos serán interpretados como 0´s). El asegurar un espacio de búsqueda suficientemente amplio, se consigue en cierta medida, permitiendo que las palabras puedan alcanzar gran tamaño. Por ejemplo, si la mayoría de las palabras son matrices de 10 x 10, se está limitando el método a redes que no tengan más de 10 nodos. Una forma de permitir que las palabras crezcan, es haciendo que la expansión sea lo más larga posible. Esto es así, porque las gramáticas bidimensionales hacen crecer a la palabra en cada nivel de derivación, al ser gramáticas de contexto libre. Hacer que el proceso de derivación sea largo se consigue con los símbolos no terminales. Al fijar las partes izquierdas de las reglas se fija también el número de símbolos no terminales que va a tener la gramática. Ahora bien, esto no asegura que la palabra vaya a ser de tamaño grande ya que en las partes derechas pueden predominar los símbolos terminales. Por lo tanto, establecer un número mínimo de símbolos no terminales en las partes derechas de las reglas serviría para asegurar que las palabras sean de tamaño grande, y en definitiva, que el espacio de búsqueda sea lo más amplio posible.

Para codificar las reglas de producción, se considerará la estructura del cromosoma formada por la codificación de cinco reglas que conducen a símbolos no terminales (excluyendo el axioma como es habitual en las gramáticas bien formadas), seguidas de la codificación de cuatro reglas que estarán formadas por símbolos terminales. Así, se evita la aparición de reglas superfluas y se asegura que la gramática genere palabras de tamaño suficiente, al tener cada símbolo no terminal una regla formada exclusivamente por símbolos no terminales. Además, para obtener palabras de tamaño considerable, el módulo de expansión de la gramática da prioridad, durante un número predeterminado de niveles de derivación, a la aplicación de reglas de producción formadas por símbolos no terminales (NT).

Además el cromosoma contiene el modo de expansión (E) y un campo que contiene las cotas para el número de niveles y aplicación de reglas recursivas (N).

Para simplificar la codificación en el cromosoma, los símbolos no terminales, y en extensión las partes izquierdas de las reglas de producción, pueden ahorrarse en la codificación, si se fija el número y orden de estas reglas de producción. La estructura de la gramática que fija la estructura del cromosoma es:

G = {Σ_T, Σ_{NT}, Axioma, P, Exp}:

- Σ_T = {0,1},
- Σ_{NT} = {S, A, B, C, D},
- Axioma = S,
- P = {S::=…., A::= …., B::= …., C::= …., D::= …., A::= …., B::= …., C::= …., D::= …. },
- Exp = { }.

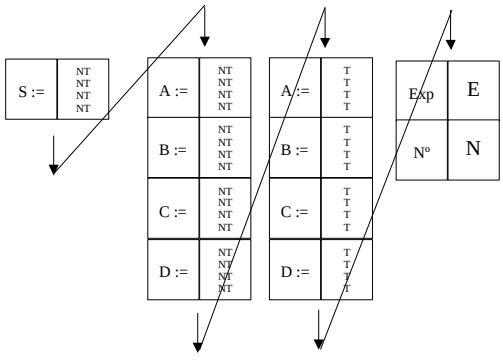

Figura 8. Codificación de la estructura gramatical en el cromosoma.

Por lo tanto la codificación en el cromosoma se corresponde con una representación binaria de las partes derechas de las reglas, la codificación del modo de expansión *Exp* y las cotas para el nivel y las reglas recursivas (N), queda el cromosoma como puede apreciarse en la Figura 8 y un ejemplo de cromosoma se muestra en la Figura 9.

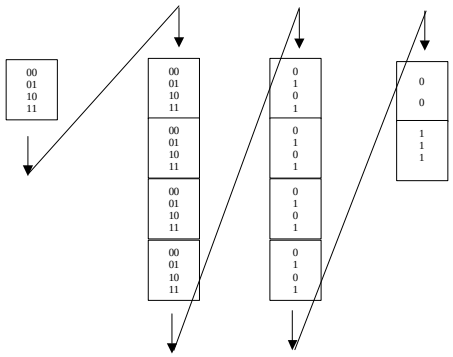

Figura 9. Codificación de la estructura gramatical en el cromosoma.

El tamaño máximo de red que se puede obtener, a partir del número máximo permitido de niveles de derivación, N, se obtiene aplicando la Ecuación 1.

$$tamaño\ máximo = 2^{\text{niveles de derivación}} + 1 \quad \text{(Ec 1)}$$

El número máximo de niveles se codifica en el cromosoma (Figura 8, campos etiquetados con N) dentro de tres nuevos genes que se colocan al final de la estructura que ya se tenía y que ya incluía el modo de expansión, (Figura 8) para formar una estructura de 61 genes. Estos tres nuevos genes indican un nivel en binario máximo que puede oscilar del 1, se corresponde con 000 para los genes (se descarta la existencia del cero), al 8, que se corresponde con 111 en los nuevos genes. Así el número máximo de neuronas es de 2^{8+1} = 512 neuronas para toda red, de las cuales 506 son ocultas. La cota para el uso de reglas recursivas se establece como la mitad del número de niveles que indican los tres nuevos genes. Así, si el nivel es 6, el número máximo de aplicaciones de cada regla recursiva es de tres.

Tabla 1. Detalle del valor de los genes incluidos en N de la estructura cromosómica.

N	Niveles	Recur.	Nº máx Total neuronas	Nº máx. neuronas ocultas
000	1	-	-	-
001	2	1	2^{2+1}= 8	2
010	3	1	2^{3+1}= 16	10
011	4	2	2^{4+1}= 32	26
100	5	2	2^{5+1}= 64	58
101	6	3	2^{6+1}= 128	122
110	7	3	2^{7+1}= 256	250
111	8	4	2^{8+1}= 512	506

En la Tabla 1 se resumen las características que tendrán las redes a generar según el valor de N en la

estructura cromosómica de cada individuo. Los datos tabulados corresponden al caso de una topología de red con cinco neuronas en la capa de entrada y una en la de salida. Se puede observar en la Tabla 1 como la combinación N = 000 no es productiva para la topología expuesta, no obstante, tendrá valor para otras topologías con menos neuronas en la capa de entrada.

Respecto al otro punto clave en la aplicación de técnicas genéticas, la medida de la aptitud (valor de *fitness* empleado por el módulo genético) de una red de neuronas, ésta se suele hacer en función de tres criterios, el error de prueba de la red, el tamaño de la red y el número de ciclos de aprendizaje. Se pueden realizar distintas combinaciones de estos parámetros u otros equivalentes para guiar la búsqueda del algoritmo genético. A modo de ejemplo algunas de las que se han utilizado en trabajos previos:

- Función de *fitness* que sólo mide la aptitud de un individuo a partir del error de la red (1/error) (Molina et al. 2000b). En este caso, no se debe esperar que la arquitectura de red obtenida destaque por su pequeño tamaño, es más, incluso debería esperarse una red sobredimensionada, en la que el error sea muy pequeño.
- Función de *fitness* que aplica un factor de corrección sobre la función de fitness anterior en función del número de conexiones ($1/K^{\text{numero de conexiones}}$) (Molina et al. 2000a). Esta función permite reducir el tamaño de la red a costa de ajustar la constante *K*, que será dependiente del problema.
- Función de *fitness* que contabiliza el número de adaptaciones de los pesos de la red (Guinea et al. 2001). En este caso el entrenamiento debe constar de muchos ciclos de entrenamiento para provocar grandes diferencias entre redes buenas y redes menos óptimas. Así, además, se agiliza la convergencia, pues un número grande de ciclos perfilará más las redes buenas y las diferencias entre redes serán más extremas. Esta cantidad de ciclos, provoca que el rango del número acumulado de adaptaciones sea muy grande, del orden de cientos de millones. Este número de ciclos a los que es sometida una red es completado por todas las redes durante el entrenamiento salvo en el caso de redes inverosímiles (generadas por algunos individuos) y que son descartadas inmediatamente con un valor de error máximo y *fitness* mínimo (como se veía en el esquema expuesto en la Figura 3, con NN no válida, aquella que no conecta la entrada con la salida). El número de ciclos de entrenamiento debe ser completado por todas las redes pues el rango de variaciones posibles debe ser igual para todas las redes a fin de evaluarlas siguiendo el mismo baremo. Así, el número de adaptaciones por ciclo de entrenamiento será proporcional al número de conexiones existentes en la estructura de la red de neuronas. Este número irá reduciéndose a medida que la red tratada se encuentre más cerca de la red de neuronas óptima, luego cuanto más cerca se encuentre de la red óptima, el error será menor y por tanto las adaptaciones se harán innecesarias. Existe otra manera de formular esta función de fitness calculando el número de adaptaciones de los pesos como el inverso del producto del número de ciclos por el número de pesos. Esta función permite mejorar el tiempo de ejecución evaluando el mismo criterio (Gutiérrez et al. 2001).

5. Análisis de la Capacidad Generativa de GANET

Para comprobar que tanto la codificación indirecta como la forma de derivar las gramáticas no determina un tipo de matriz específico y que esto no provoca que los resultados de los experimentos estén sesgados no pudiendo cubrir todo el espacio de las arquitecturas de red de neuronas. Para ello, se debe verificar que dada una muestra aleatoria de cromosomas, las matrices que se generan, se distribuyen por todo el espacio de matrices posibles.

Las matrices que se pueden generar con las gramáticas independientes del contexto bidimensionales codificadas en el módulo genético de GANET, son siempre cuadradas, después de haber sido ajustadas (ejemplo Figura 6), rondando dimensiones que oscilan de 8x8 hasta un máximo de $2^9 \times 2^9$, siempre respetando que la dimensión de las matrices se calculará como $2^{\text{nivel de derivación} + 1} \times 2^{\text{nivel de derivación} + 1}$. Se pueden generar, por tanto, 2^{262144} matrices diferentes de dimensión 512x512, 2^{65536} matrices diferentes de dimensión 256x256, etc... Esto produce como resultado una población colosal de la cual es necesario extraer una muestra representativa. En este caso, se aplicará un intervalo de confianza del 95%, el tamaño de muestra estará por debajo del 5% de la población, por lo que se puede considerar la población infinita y no aplicar los correctores propios de las poblaciones finitas. La formula empleada para averiguar el tamaño muestral

en una población infinita considerando el peor caso posible (p = 50%) es la presentada en la Ecuación 2.

$$n = \frac{1{,}96^2 * 50^2}{e^2} \qquad \text{(Ec 2)}$$

Donde:

e: máximo error que se quiere admitir.
n: tamaño necesario de la muestra.

En nuestro caso usaremos una *e = 5%.* De este modo obtendremos el deseado intervalo de confianza del 95%. Tanto para el valor de e como para el de *p*, se han usado valores que se toman por defecto a la hora de estimar muestras. Hay que destacar que el tamaño de las muestras crece vertiginosamente en cuanto se trata de pasar del 95%. Aplicando la Ecuación 2, se obtiene un tamaño de muestra de 400 individuos para un intervalo de confianza del 95%. (384,16 individuos exactamente). No obstante se ha utilizado un tamaño muestral de casi 1000 individuos (957 exactamente) en este caso para asegurar unos resultados óptimos.

Las características que se pueden representar de la matriz son su dimensión, y los valores que contiene dentro de las diferentes posiciones. Se van a representar la dimensión en el eje de abscisas y los valores de sus posiciones en las ordenadas. La dimensión está compuesta por dos valores, por lo tanto, habrá que aplicar una función que permita representar ésta con un solo valor. Y lo mismo sucede con los valores de las diferentes posiciones que componen la matriz.

La opción elegida es la siguiente. En las abscisas se debe representar la dimensión de la matriz. Durante la generación de matrices, estas no resultan cuadradas, estas matrices luego se cuadran mediante un módulo de GANET. El valor que se va a representar es, si *i* y *j* son las dimensiones vertical y horizontal respectivamente, el indicado en la Ecuación 3:

$$log10(i \times 2^j) \qquad \text{(Ec 3)}$$

Una vez determinado cómo se han de representar las dimensiones, se aplica la misma función para el valor de las posiciones de la matriz completa. Este valor se calcula como la suma de los valores de cada una de sus filas. Como el valor de la posición es 0 o 1, los valores 0 no se suman. Entonces, si *k y h* son la posición dentro de la matriz, y el valor almacenado en esa posición es igual a 1, se transformará según la Ecuación 4.

$$log10(k \times 2^h) \qquad \text{(Ec 4)}$$

Para cada posición dentro de una fila que contenga un 1 se aplica la Ecuación 4 y se suman al final los resultados de todas las posiciones de la fila. Una vez hecho esto para todas las filas, se suman todos los totales por filas y ese es el valor de la matriz. Se obtiene mediante la función de la Ecuación 5.

$$\sum_{1}^{H}\sum_{1}^{K} \log_{10}(k * 2^h) * X_{k,h} \qquad \text{(Ec 5)}$$

Donde:

H: dimensión vertical de la matriz.
K: dimensión horizontal de la matriz.
k: columna de la matriz.
h: fila de la matriz.
$X_{k,h}$: valor de posición en la columna *k* y fila *h* de la matriz.

Con los valores calculados procedemos a su representación gráfica en dos dimensiones. En el eje de abscisas se sitúa el valor calculado para la dimensión de cada matriz y en de ordenadas se sitúa el valor calculado para las posiciones de la matriz. En la Figura 10 se ilustran los resultados obtenidos.

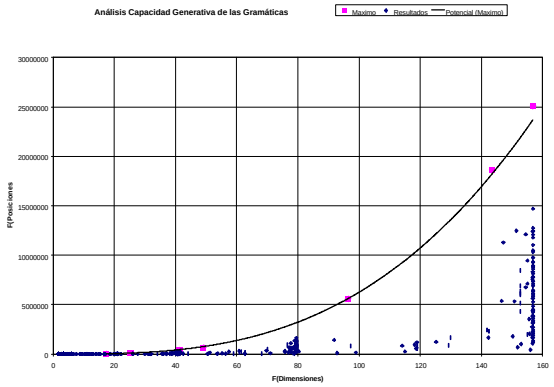

Figura 10. Estudio de capacidad generativa.

En la Figura 11 se muestran los resultados de la Figura 10 pero empleando una escala menor en las posiciones.

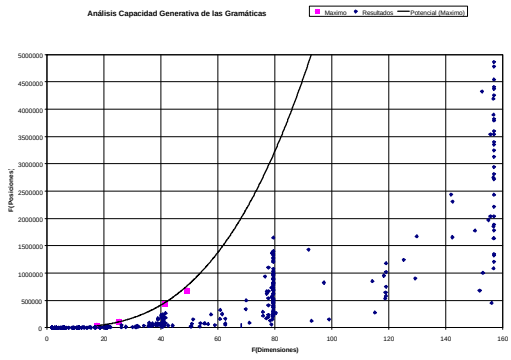

Figura 11. Estudio de capacidad generativa (escala menor).

A mayor dimensión tenga la matriz más a la derecha estará el punto que la representa. Y a más unos dentro de la matriz, más hacia arriba. Los puntos más grandes y la recta de tendencia de la Figura 10 y Figura 11 representan los valores máximos de la función. Los valores generados tienen que estar siempre por debajo de los puntos máximos. La primera peculiaridad que destaca es la aparición de "hueco" entre los valores 100 y 140 del eje horizontal. Esto se debe a que por la forma de las gramáticas bidimensionales utilizadas, los resultados se acumulan en líneas rectas verticales en las gráficas debido a que, lo más común, es obtener una matriz de dimensión 2^n x 2^n o acercarse a ella, esto es, 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, 256x256, 512x512. Lo de acercarse a ella, se debe a que las matrices que se obtienen no son cuadradas en primera instancia se "cuadran" en un proceso posterior. Así, de 957 matrices que forman parte de la muestra utilizada:

- 113 matrices tienen dimensión 4x4 y por tanto su valor para el eje X de abscisas es 1,80618.
- 114 matrices tienen dimensión 8x8 y por tanto, su valor para el eje X de abscisas es 3,31133.
- 92 matrices tienen dimensión 16x16 y por tanto, su valor para el eje X de abscisas es 6,0206.
- 79 matrices tienen dimensión 32x32 y por tanto, su valor para el eje X de abscisas es 11,13811.
- 68 matrices tienen dimensión 64x64 y por tanto, su valor para el eje X de abscisas es 21,0721.
- 69 matrices tienen dimensión 128x128 y por tanto, su valor para el eje X de abscisas es 40,639049.
- 56 matrices tienen dimensión 256x256 y por tanto, su valor para el eje X de abscisas es 79,471919.
- 41 matrices tienen dimensión 512x512 y por tanto, su valor para el eje X de abscisas es 156,836628.

En total, 632 matrices se ajustan a la forma 2^n x 2^n, más de dos tercios de la muestra. Entonces, si los valores que se han representado en el eje X van desde 0 a 160, resulta que en el 25% inicial (de 0 a 40) del eje X, correspondiente a la zona más fina por la forma logarítmica, se encuentran un total de 535 matrices. Posteriormente, se salta del valor 40,639049 al 79,471919, casi un 25% más del eje X en el que se representan todos los puntos que identifican matrices que no tienen la dimensión en forma 2^n x 2^n y que por tanto son minoritarias. Estas matrices pertenecen al caso de obtener matrices no cuadradas que son ajustadas posteriormente.

En la mitad del eje X hay condensación de puntos pues, en esa región, se representan las matrices con dimensión 256 x 256. También al final del eje X, existe acumulación, debido a las matrices 521 x 512 que tienen un valor de 156,836628. Pero entre la mitad del eje y el final del mismo, sólo unos pocos puntos pueblan el gráfico y esto se debe a que esa región pertenece a las matrices que no tienen la forma 2n x 2n y que tampoco se acercan a ella.

La segunda peculiaridad observable se resume en la siguiente pregunta ¿por qué al avanzar en el eje X, aparecen menos puntos representados? o lo que es lo mismo ¿por qué hay más matrices pequeñas que grandes?. la respuesta puede ser que una matriz con un nivel máximo de recursividad *N* igual a 5 (valor 010 en los genes últimos del cromosoma) puede dar lugar al decodificar y derivar la gramática en él contenida a todo tipo de matrices hasta la dimensión 32x32. Lo mismo ocurre con el resto, y así, los cromosomas con nivel 8 pueden generar palabras con cualquier dimensión hasta 512 x 512. Esto sin duda favorece la aparición de matrices de dimensiones pequeñas y medias, lo que sin duda ha favorecido a la convergencia en los experimentos llevados a cabo.

Resumiendo, de los resultados representados en las Figuras 10 y 11 se puede decir que el método muestra la capacidad de generar todo tipo de tamaños de redes, aunque se favorece la formación de matrices 2^n x 2^n y dentro de estas, se favorece la aparición de matrices pequeñas.

6. Análisis y Validación del Sistema

El sistema GANET ha sido aplicado para determinar la red de neuronas con conexiones hacia adelante más sencilla, capaz de aproximar la serie logística. La serie logística viene dada por la Ecuación 6.

$$x_{t+1} = l \cdot x_t \cdot \left(1 - x_t\right) \qquad \text{(Ec 6)}$$

El término λ de la Ecuación 6 es una constante, que determina el comportamiento de la serie. En este trabajo se ha empleado un valor de λ=3.97 y x(0) = 0.5 y el comportamiento de la serie es fuertemente caótico. El uso de la serie logística presenta la ventaja de que la red óptima para predecir la serie es conocida. Como el valor de la serie en el instante *t* depende únicamente del valor de la serie en el instante *t-1*, la red óptima deberá considerar únicamente la entrada que transmite la señal *t-1*. Así, con un horizonte de predicción de *t+1*, la

ejecución del sistema GANET deberá obtener una arquitectura de red de neuronas, capaz de predecir el valor x_{t+1} de la serie, dado el valor x_t.

El número de neuronas de las capas de entrada y de salida se mantiene constante durante la ejecución de Ganet, siendo éstos parámetros de entrada al problema. Así, lo que realmente se calcula con la aplicación del método, es el número de neuronas de la capa oculta, y las conexiones entre estas neuronas con las de la capa de entrada y de salida. Para resolver el problema se ha fijado en 5 el número de neuronas de entrada. El hecho de establecer en cinco el número de neuronas de entrada, cuando en realidad solamente es necesaria una única neurona de entrada, persigue el objetivo de comprobar si el método es capaz de eliminar las cuatro primeras neuronas de entrada, que son innecesarias en este problema. En este caso, el método devolverá una arquitectura de red, donde las cuatro primeras neuronas no estarán conectadas con la capa oculta, tal y como se muestra en la Figura 12.

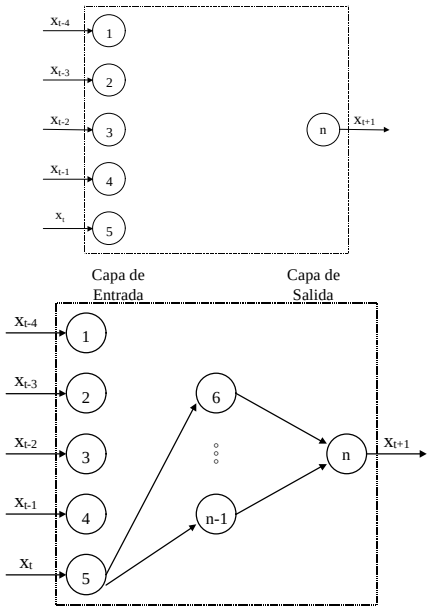

Figura 12. Descripción del problema y una red típica para resolverlo.

La red de la Figura 12, no es la única posible, pues, aunque se acaba de indicar que las cuatro primeras neuronas resultan innecesarias para la predicción del valor x_{t+1} de la serie logística, ocurre que estas cuatro neuronas influyen de forma positiva en el valor de salida de la red, a partir de un cierto número de neuronas ocultas. Es decir, si estas neuronas de entrada se conectasen con neuronas de la capa oculta, el error de prueba de la red sería menor que el que se obtendría si solamente fuese la última neurona de entrada, la que se conectase con la capa oculta. Por lo tanto, el grado de validez de la arquitectura de red de neuronas que se obtenga con la ejecución del método, vendrá determinado por los objetivos que se hayan marcado en la propia búsqueda de dicha arquitectura, quedando estos objetivos reflejados en la función de *fitness* (elección de la red mayor con el menor error vs. la red con el menor tamaño, aunque el error sea mayor, todo ello con un límite de error aceptable).

Los experimentos que se incluyen en este apartado, se basan en la función de fitness que contabiliza el número de actualizaciones de los pesos de la red. El número de ciclos de entrenamiento máximo se ha fijado en 15.000 y el error que se debe obtener se ha determinado en 0,007. Las redes que tras 15.000 ciclos de entrenamiento no hayan llegado al error deseado serán penalizadas con un valor muy elevado. Dentro de las redes que obtienen el error deseado, las que lo hagan con menos neuronas en su capa oculta obtendrán un número de modificaciones acumuladas inferior a redes con más neuronas en su capa intermedia. De esta manera la función de fitness es la que se presenta en la ecuación 7, si la red obtiene un error mayor al contrastar sus salidas con los patrones de salida programados, el contador de adaptaciones pasa a valer 9.999.999.999 de modificaciones, si la red, en cambio, alcanza un valor al menos de 0,007 en el error, se respeta su valor en el contador de adaptaciones, que contiene el número de adaptaciones de los pesos de la red durante los 15000 ciclos de entrenamiento, y es este valor, el que se usa para calcular el grado de eficiencia de la red mediante la función de *fitness* de esa red de neuronas.

En ningún caso, el número de adaptaciones puede alcanzar las 9.999.999.999 modificaciones, por tanto, se establece una diferencia abismal entre las redes de neuronas que han proporcionado un error aceptable durante el entrenamiento y las que no lo han conseguido a la hora de procesarse los resultados obtenidos mediante la aplicación del algoritmo genético.

A continuación, en la Ecuación 7 se resume la función de *fitness* utilizada.

$$f = \begin{cases} \dfrac{1.000.000}{N} & si \quad \dfrac{\sum_{j=1}^{p} (s_j - d_j)^2}{p} < 0.008 \\ \dfrac{1.000.000}{9.999.999.999} & si \quad \dfrac{\sum_{j=1}^{p} (s_j - d_j)^2}{p} \geq 0.008 \end{cases} \quad \text{(Ec 7)}$$

Donde:

N: número de adaptaciones de los pesos durante el proceso de entrenamiento.

p: nº patrones de entrenamiento.

S: salida de la red
d: salida deseada

Tras la fase de entrenamiento, una buena red adquiere el conocimiento y la experiencia necesarios para afrontar la toma de decisiones ya sin supervisión. En este momento se dispone de una red preparada para medir su eficiencia. Con los pesos de las conexiones calibrados tras 15000 ciclos de entrenamiento, se vuelve a aplicar un ciclo (en lo que se denomina "probando NN" de la Figura 3 con los patrones correspondientes, para ver el error que es capaz de conseguir dicha red utilizando para medirlo una función que mide el error cuadrático).

En los experimentos se han empleado 95 patrones, utilizando estos patrones tanto para el entrenamiento como para la validación de la red de neuronas. Los patrones se corresponden con 100 iteraciones de la serie logística, con x0=0.5 y λ=3.97. Así mismo, en los experimentos se ha empleado un factor de aprendizaje α=0.9, y no se ha incluido el término momento en el algoritmo de aprendizaje. Los parámetros del módulo genético de Ganet son los siguientes:

- 200 generaciones para el módulo genético.
- Tamaño de la población de 50 individuos.
- Tamaño de la población de padres de 35 individuos.
- Porcentaje de solapamiento del 10%, incrementado un 2% cada 40 generaciones.
- Porcentaje de mutación del 1%.

A continuación se muestran los resultados obtenidos como valor medio de 15 ejecuciones independientes del sistema. En la Figura 13 y Figura 14 se muestra la evolución del error durante los experimentos. Se muestran los individuos con mejor error, con peor error y el error medio. Además se incluye el error que obtiene el mejor valor de la función de *fitness* (mejor individuo).

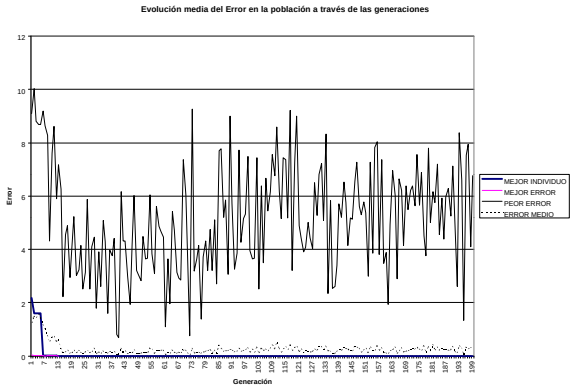

Figura 13. Representación gráfica de la evolución media del error.

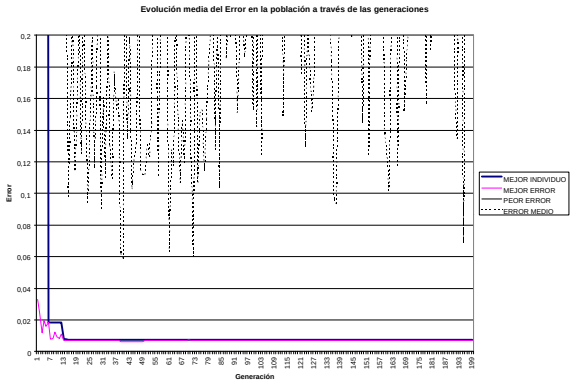

Figura 14. Representación gráfica de la evolución media del error mostrada en la Figura 13 (escala menor).

En la Figura 13 y Figura 14 se puede observar como en 13 generaciones el mejor individuo coincide con el individuo que tiene un menor error. Esto quiere decir que la condición impuesta de ajustarse a una cota de error determinada se satisface para el mejor individuo en muy pocas generaciones.

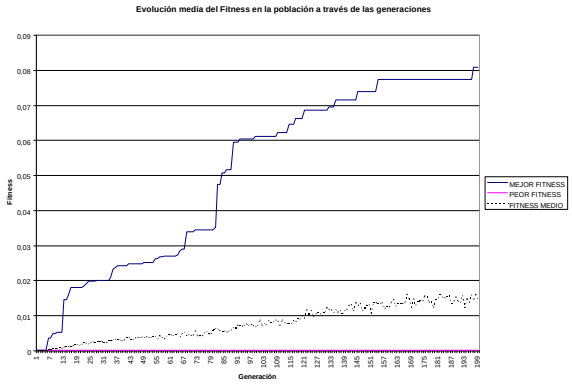

Figura 15. Representación gráfica de la evolución media del fitness.

En la Figura 15 y Figura 16 se muestra la evolución del fitness durante los experimentos. El fitness muestra una tendencia creciente como es habitual en la aplicación de algoritmos genéticos. Un valor de fitness de 0,083333 se corresponde con un valor de 12.000.000 actualizaciones, siendo este número el de una arquitectura similar a la óptima. Por el contrario valores cercanos a 0 representan el caso en que la red no ha superado el error máximo permitido ya que 1.000.000/9.999.999.999 es 0,0001.

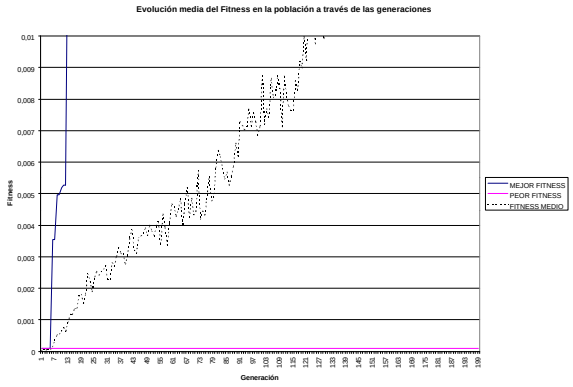

Figura16. Representación gráfica de la evolución media del fitness, (Figura 15 escala menor).

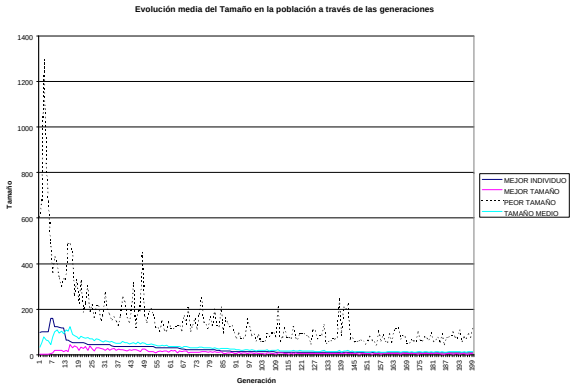

Figura 17. Representación gráfica de la evolución media del tamaño.

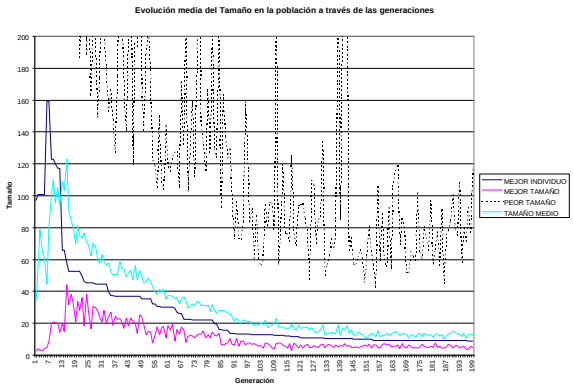

Figura 18. Representación gráfica de la evolución media del tamaño (Figura 17 a escala menor)

En la Figura 17 y Figura 18 se muestra la evolución del tamaño de las redes de neuronas a lo largo de los experimentos. Como en el caso del error se muestran los tamaños mejor, peor y medio, además se superpone el tamaño del individuo con mejor fitness. En este caso se observa como el tamaño del mejor individuo se aproxima a un valor constante a mucha menor velocidad que en el caso del error. Esto se debe a que la presión selectiva sobre el tamaño es mucho menor que sobre el error máximo permitido. Además el tamaño menor no se alcanza nunca debido a que dicho tamaño se corresponde con redes que no tiene sentido y son penalizadas.

7. Conclusiones

La principal consecuencia de este trabajo es que el sistema GANET propuesto es una extensión, con éxito, del método propuesto por Kitano en (Kitano 1990). Se ha desarrollado un método de diseño de arquitecturas de redes de neuronas, mediante la evolución de gramáticas bidimensionales, general y potente. Las gramáticas empleadas superan las restricciones del método de Kitano. Estas restricciones afectan al contenido de la gramática, a la estructura de las redes de neuronas y al proceso de reproducción de los algoritmos genéticos.

En el método desarrollado, las gramáticas que se emplean son más genéricas, porque se permite recursividad y la mezcla tanto de símbolos no terminales como terminales en la parte derecha de las reglas de producción. En el sistema de Kitano el número de nodos de las redes venía impuesto porque el tamaño de palabra era fijado en el cromosoma; en el modelo desarrollado, GANET, el tamaño de cada palabra viene determinado tanto por la gramática, como por su expansión. El cromosoma de los individuos, en el modelo desarrollado, tiene dos ventajas: se asegura la correcta expansión de las gramáticas, al no incluirse reglas superfluas, y se puede emplear el operador de sobrecruzamiento sobre la totalidad del cromosoma, obteniéndose siempre una red válida.

Además, el sistema GANET permite obtener resultados de calidad tanto desde el punto de vista de las redes obtenidas, como desde el punto de vista de la evolución genética. Respecto a la evolución genética, se ha llegado a la conclusión que emplear gramáticas bidimensionales en la codificación de las redes de neuronas dentro del cromosoma de los individuos, permite reducir el tamaño de estos cromosomas y mejora la convergencia del algoritmo. También es importante reseñar, que el emplear gramáticas en lugar de codificar las conexiones de la red directamente, no reduce el tipo de redes que pueden representarse, con lo que el espacio de búsqueda no se ve afectado por el hecho de emplear gramáticas bidimensionales.

Referencias

Alba, E. Aldana J.F. and Troya J.M. (1993). Fully automatic ANN design: A genetic approach. In

Proc. of International workshop on artificial neural networks, pp 179-184.

Ash T. (1988). Dynamic Node Creation in Backpropagation Networks, ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988).

Caudell T.P. and Dolan C.P. (1989). Parametric Conectivity: Training of Constrained Networks using Genetic Algorithms, Proceedings of the third International Conference on Genetic Algorithms and their Applications, 370-374. Morgan Kaufman.

Chomsky N. (1959). On Certain Formal Properties of Grammars, Information and Control 2,137-167.

Cybenko G. (1989). Approximation by supperposition of a sigmoidal function. Mathematics and Control, Signals and Systems, 2, 303-314.

Ehring, Nagel, Rozemberg and Rosenfeld (1987). Graph Grammars and their Applications to Computer Science, Lecture Notes in Computer Science.

Fogel D.B., Fogel L.J. and Porto V.W. (1990). Evolving Neural Network, Biological Cybernetics, 63, 487-493.

Galván I.M., Sanchis A., Isasi P., Molina J. M., (2000) Neural Networks Architectures Design by Cellular Automata Evolution, The 4th World Multiconference on Systemics, Cybernetics and Informatics SCI 2000 and The 6th International Conference on Information Systems, Analysis and Synthesis ISAS 2000. Vol III, pp. 457-462. USA. Julio.

Goldberg D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York

Gruau F. (1992). Genetic Synthesis of Boolean Neural Netorks with a Cell Rewriting Developmental Process. Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press,

Gruau F. (1994). Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm. PhD Thesis, Ecole Normale Supèrieure de Lyon.

Gruau F. (1995). Automatic Definition of Modular Neural Networks. Adaptive Behavior, vol. 2, 3, 151-183.

Guinea M. A., Sanchis A., Molina J. M.. (2001) Characteristics of Grammars Codification for Evolution of NN Architectures, IASTED International Conference Artificial Intelligence and Applications (AIA 2001), pp 72-76. España, Septiembre.

Gutiérrez G., Isasi P., Molina J.M., Sanchis A. and Galván I. M.. (2001) Evolutionary Cellular Configurations for Designing Feed-Forward Neural Networks Architectures, 6th International Work-Conference on Artificial and Neural Networks, IWANN'01, pp 514-521. Granada, España. Junio.

Harp S., Samad T. and Guha A. (1989). Towards the Genetic Synthesis of Neural Networks, Proceedings of the Third International Conference on Genetic Algorithms.

Harp S., Samad T. and Guha A. (1990). Designing Application-Specific Neural Networks using the Genetic Algorithm, Advances in Neural Information Processing Systems, vol2, 447-454, Morgan Kaufmann.

Hertz J., Krough A. and R.G. Palmer (1991). Introduction to the Theory of Neural Computation. Addison-Wesley.

Hopcroft J.E. and Ullman J.D. (1979). Introduction to Automata Theory, Languages and Computation, Addison-Wesley.

Kitano H. (1990). Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476.

Merril J.W.L. and Port R.F. (1991). Fractally configured Neural Networks. Neural Networks, 4, 53-60.

Miller G., Todd P. and Hedge S. (1989). Designing Neural Networks using Genetic Algorithms, Proceedings of the Third International Conference on Genetic Algorithms.

Mjolsness E., Sharp D.H. and Alpert B.K. (1989). Scaling machine learning and genetic neural nets. Advances in applied mathematic, 10, pp 137-163.

Molina J. M., Galván I., Isasi P., Sanchis A.. (2000b) Grammars and Cellular Automata for Evolving Neural Networks Architectures, IEEE International Conference on Systems, Man and Cybernetics. pp. 2497-2502. USA. Octubre.

Molina J. M., Galván I.M., Valls J.M., Leal A.. (2002) Optimizing the Number of Learning Cycles in the Design of Radial Basis Neural Networks using a Multi-Agent System, Computers and Informatics (anteriormente Computers and Artificial Intelligence). Aceptado para su publicación.

Molina J. M., Torresano A., Galván I., Isasi P., Sanchis A.. (2000a) Evolution of Context-free Grammars for Designing Optimal Neural Networks Architectures, GECCO 2000, Workshop on Evolutionary Computation in the Development of ANN. USA. Julio, 2000.

Schaffer, J.D. Caruana R.A. and Eshelman L.J. (1990). Using genetic search to exploit the emergent behaviour of neural networks. Physica D, 42, pp 244-248.

Valls J.M., Galván I.M. and Molina J. M. (2000),Multiagent System for designing optimal Radial Basis Neural Networks, Information Processing and Management of Uncertainty in Knowledge Based Systems. Spain..