Modelado de Sistemas Mediante Modelos Híbridos de Redes Neuronales y Computación Evolutiva.

Alfonso C. Martínez [1], Francisco J. M. Estudillo [1], Nicolás García[2], César Hervás[2]

[1] Facultad de Ciencias Económicas y Empresariales. ETEA
14005- Córdoba- España
{acme,fjmestud}@etea.com
[2] Departamento de Computación Universidad de Córdoba
14071- Córdoba- España
{npedrajas,chervas}@uco.es

Abstract.[1] En el trabajo presentamos un novedoso modelo de red neuronal que realiza un procedimiento de regresión simbólica partiendo de una determinada tipología de funciones. Tanto el aprendizaje de la red como su topología son guiados por un algoritmo de computación evolutiva. Se evoluciona un tipo de redes neuronales conocidas como redes de regresión para llegar a encontrar una función de la familia que sea capaz de explicar el sistema. Tanto los errores de previsión cometidos como la robustez del algoritmo muestran la viabilidad de este tipo de modelo de redes de regresión.

1 Introducción

El modelado de sistemas es en la actualidad uno de los problemas de mayor interés en numerosas ramas de la ciencia. En diversas áreas de aplicación, surge el problema de establecer una relación funcional entre las diferentes variables que intervienen en el fenómeno que se está estudiando. La resolución de este problema se ha abordado clásicamente usando técnicas de regresión para minimizar una determinada función de error, previo establecimiento por parte del investigador del tipo de modelo a aplicar. En la mayoría de los casos, el modelo a aplicar es no lineal y además suele presentar una alta dimensionalidad, lo que complica considerablemente el proceso. Las redes neuronales han sido utilizadas en los últimos años para la resolución de este problema, [3],[10],[11]. La justificación de esta capacidad de las redes neuronales se encuentra en el siguiente hecho: cualquier función continua puede ser aproximada sobre un conjunto compacto por una red neuronal con una sola capa intermedia, con tal de que ésta tenga un número suficiente de nodos, aunque esto no implica el aprendizaje de la red [1],[2]. Sin embargo, en la práctica, el número de nodos ocultos necesarios para realizar la tarea de aproximación suele ser muy alto, lo que provoca la escasa interpretabilidad de los resultados obtenidos. A este hecho hay que unir que, con frecuencia, en los algoritmos de entrenamiento de búsqueda local, la superficie de

[1] Este trabajo ha sido financiado en parte por la CICYT en el proyecto TIC. 2001-2577

error está plagada de mínimos locales y superficies planas que complican la búsqueda del mínimo global.

Una alternativa interesante es tratar el modelado de sistemas usando algoritmos evolutivos que, por una parte, permitan obtener expresiones analíticas del modelo que puedan ser interpretables, y por otra, no queden atrapados en mínimos locales durante el proceso de búsqueda. Los algoritmos evolutivos son un tipo de algoritmos de búsqueda estocástica que permiten encontrar la solución en espacios complejos. Dichos algoritmos se han utilizado con éxito en el campo de las redes neuronales para encontrar la estructura adecuada de la red. Incluso, también se han utilizado, para calcular el valor de los pesos de las conexiones evitando quedar atrapado en mínimos locales, generalmente derivados de un sobreentrenamiento.[7],[8],[9].

En el proceso de evolución, son esenciales los operadores de selección, replicación y mutación, para introducir por una parte presión selectiva y por otra diversidad de forma tal que el algoritmo de búsqueda de la mejor red presente un compromiso entre su capacidad de explotar la localización de las mejores soluciones y explorar otras localizaciones del espacio, con el fin de obviar el problema de la obtención de óptimos locales. La no inclusión del operador de cruce se debe al problema del engaño [7] asociado a este tipo de codificación del cromosoma, lo que hace que sea en general poco eficiente.

En este trabajo, partiendo del conocimiento previo de la tipología de funciones que se quiere modelar, se plantea el uso de algoritmos evolutivos para realizar la búsqueda de la función que modelice el fenómeno estudiado. En el proceso, se utiliza como elemento de representación de las funciones del modelo, un tipo de redes neuronales conocidas como redes de regresión. Este tipo de redes se caracteriza fundamentalmente por ser fácilmente interpretables. Además, un gran número de funciones pueden ser representadas mediante las redes de regresión. La estructura de la red (el número de capas y de nodos, la conectividad y las funciones de transferencia) dependerán del tipo de función que se desee representar [3].

El método de trabajo propuesto consiste en aplicar las técnicas y algoritmos propios de la computación evolutiva a las redes de regresión, evolucionando la estructura y los pesos de las conexiones de éstas con el objetivo de llegar a encontrar la función dentro de la familia de funciones propuesta que mejor modelice el conjunto de datos.

2 Metodología Propuesta

Comenzamos definiendo la familia de funciones que se usará en el proceso de modelización y su representación a través de la correspondiente red de regresión.

2.1 Tipología de Funciones y Redes de Regresión Asociada.

El tipo de función con el que trabajaremos es el siguiente:

$$f : A \subset \mathbb{R}^{N^{(0)}} \rightarrow \mathbb{R}$$

$$f(x_1, x_2, ..., x_{N^{(0)}}) = \sum_{j=1}^{N^{(1)}} \overline{w}_j \left(\prod_{i=1}^{N^{(0)}} x_i^{w_{ji}} \right) \quad (\mathbf{1})$$

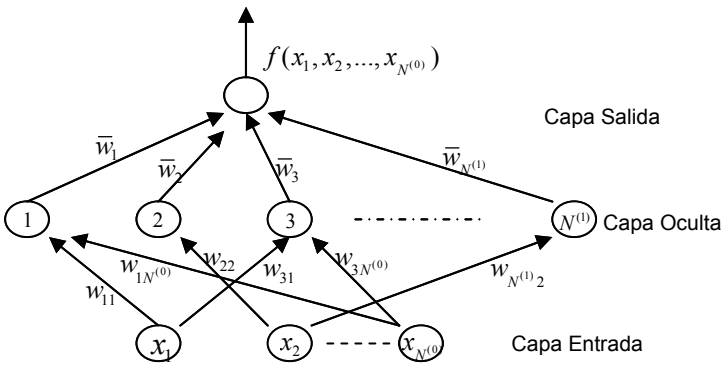
donde $\overline{w}_j \in [-M, M] \subset \mathbb{R}$, $w_{ji} \in [-L, L] \subset \mathbb{R}$, $i = 1, 2, ..., N^{(0)}$, $j = 1, 2, ..., N^{(1)}$ y el dominio de definición de f es $A = \left\{ (x_1, x_2, ..., x_{N^{(0)}}) \in \mathbb{R}^{N^{(0)}} : 0 < x_i \leq K \right\}$. En el caso más sencillo, con dos variables independientes, la función definida en (1), vendría dada por:

$$f(x_1, x_2) = \sum_{j=1}^{N^{(1)}} \overline{w}_j x_1^{w_{j1}} x_2^{w_{j2}} \quad (\mathbf{2})$$

La tipología de funciones definidas en (1) puede considerarse una generalización de las superficies de respuesta [12]. Observemos, por ejemplo, que si en (1) elegimos convenientemente los exponentes $w_{ji} \in \{0, 1\}$ se obtiene una superficie de respuesta cuadrática del tipo:

$$f(x_1, x_2, ..., x_n) = c_0 + \sum_{i=1}^{N^{(0)}} c_i x_i + \sum_{i,j=1}^{N^{(0)}} c_{ij} x_i x_j$$

A continuación, a partir de las redes de regresión, representamos la tipología de funciones definida en (1). Las redes de regresión que consideraremos tienen las siguientes características: una capa de entrada para cada una de las variables de entrada, una sola capa oculta con varios nodos y una única capa de salida con un nodo. Además, los nodos de una misma capa no pueden estar conectados entre si y no existen conexiones directas entre las capas de entrada y salida.
La estructura de la red es por tanto la siguiente:



Suponemos que la red tiene $N^{(0)}$ entradas representadas por las variables independientes, $(x_1, x_2, ..., x_{N^{(0)}})$, $N^{(1)}$ nodos en la capa oculta y $N^{(2)} = 1$ nodo en la capa de salida. La activación de cada nodo j de la capa oculta está dado por

$$\phi_j = \prod_{i=1}^{N^{(0)}} x_i^{w_{ij}} \ , \quad w_{ij} \in [-L, L] \tag{3}$$

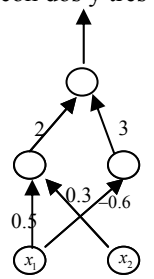
donde w_{ji} son los pesos que conectan el nodo j de la capa oculta con el nodo i de la capa de entrada. Por otra parte, la activación de cada nodo de la capa de salida está dado por

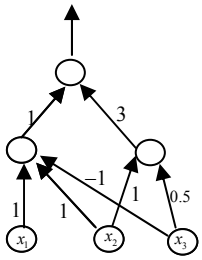
$$\sum_{j=1}^{N^{(1)}} \overline{w}_j \phi_j \ , \quad \overline{w}_j \in [-M, M] \tag{4}$$

siendo $\overline{w}_j$ el peso que conecta el nodo j de la capa oculta con el nodo de salida. Además, las funciones de transferencia de cada nodo de la capa oculta y la función de transferencia del único nodo de la capa de salida son iguales a la función identidad. De esta forma, la tipología de funciones definidas en (1) quedan representadas por la red de regresión que acabamos de definir.

2.1.1 Ejemplos.

A continuación, se muestra cómo sería la estructura de la red de regresión para dos funciones ejemplo con dos y tres variables.





$$f(x_1, x_2) = 2x_1^{0.5} x_2^{0.3} + 3x_1^{-0.6} \qquad f(x_1, x_2, x_3) = x_1 x_2 x_3^{-1} + 3x_2 x_3^{0.5}$$

2.2 Función de Error.

El problema del modelado de un sistema mediante un conjunto de datos se traduce en encontrar el mínimo global de una función error que mide la diferencia existente entre los valores conocidos de la variable predictiva y la predicción de los mismos realizada a través del modelo, y cuyas variables son los parámetros del tipo de función que se está utilizando en el modelado. Normalmente, la función de error utilizada es la suma de residuos al cuadrado. En el presente trabajo, debido a la tipología de funciones considerada y al dominio en el que se definen, es frecuente que los valores de la función tengan su variación en un rango muy amplio, especialmente cuando los exponentes de la función toman valores cercanos al rango máximo establecido. Por este motivo, se ha elegido la siguiente función de error que establece el valor relativo

de las desviaciones absolutas entre el valor real y_k de la función en cada punto e $\hat{y}_k$ la correspondiente predicción realizada:

$$E = \frac{1}{N^{(p)}} \left[\sum_{k=1}^{N^{(p)}} \frac{\left(y_k - \hat{y}_k\right)^2}{|y_k|} \right]^{1/2} \quad (5)$$

La función de error definida en (5) ha dado buenos resultados en el modelado de redes de regresión [3].

La aptitud $A(R)$ de una red R se calcula como una función decreciente del error $E(R)$ a partir de la expresión:

$$A(R) = \frac{1}{1 + E(R)}, \quad 0 < A(R) \leq 1 \quad (6)$$

Definida la tipología de funciones y la función de error con las que trabajaremos, pasamos a continuación a describir el algoritmo de evolución usado en el proceso. El algoritmo está basado en los siguientes trabajos [6],[7],[9]

2.3 Proceso General de Evolución.

La idea básica es el uso de los operadores de selección, de replicación y de mutación (paramétrica y estructural) en el proceso de evolución. La evolución de la topología de la red corresponde, en el contexto que acabamos de describir de las redes de regresión, a la búsqueda de la estructura de la función que mejor se ajuste a la nube de puntos del conjunto de entrenamiento, determinando el número de monomios de la función y las variables presentes en cada monomio. Mientras que la evolución de los pesos de la red se corresponde con la evolución de los coeficientes de cada monomio y de los exponentes de cada una de las variables. El algoritmo se estructura en los siguientes pasos:

1.- Generación de la población
2.- Repetir hasta que se cumpla la condición de parada
- Cálculo de la aptitud para cada individuo
- Ordenación de mayor a menor según la aptitud.
- Copia del mejor en la siguiente generación
- Replicación del 10% mejor de la población por el 10% peor.
- Aplicar mutación paramétrica al 10% mejor
- Aplicar mutación estructural al 90% restante

2.4. Algoritmo Evolutivo.

El algoritmo se inicia generando aleatoriamente N_R redes. Se comienza eligiendo el número de nodos ocultos a partir de una distribución uniforme en el intervalo

$\left[0, N^{(1)}\right]$, donde $N^{(1)}$ corresponde al número máximo de nodos ocultos de cada una de las redes de la población. El número de conexiones entre cada nodo de la capa oculta y los nodos de la capa de entrada queda determinado a partir de una distribución uniforme en el intervalo $\left[0, N^{(0)}\right]$, siendo $N^{(0)}$ el número de variables independientes. Definida la topología de la red, se asigna a cada conexión un peso, a partir de una distribución uniforme en el intervalo $[-L, L]$ para los pesos entre la capa de entrada y la oculta y $[-M, M]$ para los pesos entre la capa oculta y la de salida. Tras la generación aleatoria de la población de N_R redes, se inicia el proceso de selección. Se realiza una selección por elitismo del 10%. Las redes seleccionadas sustituyen al 10% con peor aptitud, de forma que el número N_R de redes permanece constante durante la evolución.

Las mutaciones realizadas en el algoritmo son de dos tipos: las mutaciones paramétricas afectan a los pesos de la red y las mutaciones estructurales afectan a la topología de la red (nodos ocultos y conexiones). La severidad de las mutaciones depende de la temperatura $T(R)$ de la red definida por:

$$T(R) = 1 - A(R), \quad 0 \le T(R) \le 1 \tag{7}$$

Cuando el proceso de búsqueda se está iniciando, la aptitud de la red es cercana a cero y la temperatura es alta, por lo que las mutaciones son más fuertes. A medida que nos acercamos a la solución, aumenta la temperatura, por lo que los cambios que se realizan tanto en los pesos como en la estructura de las redes van siendo más pequeños.

Las mutaciones paramétricas consisten en la alteración de cada peso $w_{ji}, \overline{w}_j$ con ruido gaussiano, donde la varianza de la distribución de Gauss depende de la temperatura de la red. Concretamente, los pesos w_{ji} de la capa intermedia quedan actualizados usando:

$$w_{ji}(t+1) = w_{ji}(t) + N(0, \alpha_1(t)T(R)), \quad \forall i, j \tag{8}$$

mientras que los pesos de la capa de salida se actualizan de forma análoga,

$$\overline{w}_j(t+1) = \overline{w}_j(t) + N(0, \alpha_2(t)T(R)) \tag{9}$$

donde $\alpha_1(t) \ll \alpha_2(t), \quad \forall t$. Las mutaciones de los pesos w_{ji}, correspondientes a los exponentes de las variables de la función que queremos modelizar, han de ser menores que las mutaciones realizadas de los pesos $\overline{w}_j$, correspondientes a los monomios de la función, debido al diferente efecto que tienen dichas modificaciones sobre los valores de la función.

Una vez realizado el cambio en el espacio de los pesos, la aptitud del individuo es recalculada y se aplica un algoritmo estándar de enfriamiento simulado. Si llamamos ΔA a la variación de la aptitud antes y después del cambio en los pesos, resulta que

- Si $\Delta A \ge 0$, se acepta el cambio

- Si $\Delta A < 0$, se acepta el cambio si $e^{\frac{\Delta A}{T}} < \gamma$, donde $\gamma \in U(0,1)$ y $U(0,1)$ representa una distribución uniforme en el intervalo [0,1].

Los parámetros α_1, α_2 que determinan la varianza, junto con la temperatura, de la distribución normal, a diferencia que en [7],[9] que permanecen fijos durante toda la evolución, van cambiando durante el proceso de evolución.
Concretamente, la evolución de los parámetros α_1, α_2 viene dada por:

$$\alpha_i(t+1) = \begin{cases} (1+\beta)\alpha_i(t), & \text{si } A(s) > A(s-1), \quad \forall s \in \{t, t-1, ..., t-r\} \\ (1-\beta)\alpha_i(t), & \text{si } A(s) = A(s-1), \quad \forall s \in \{t, t-1, ..., t-r\} \\ \alpha_i(t) & \text{en el resto de casos} \end{cases}$$

$i = 1,2$, donde $A(s)$ representa la aptitud de la red con aptitud máxima en la generación s. Los valores considerados para los parámetros $\alpha_1, \alpha_2, \beta, r$ en todos los experimentos realizados han sido: $\alpha_1(0) = 1, \alpha_2(0) = 5, \beta = 0.1, r = 10$, aunque el algoritmo es bastante robusto con respecto a los valores $\alpha_1(0), \alpha_2(0)$.

La mutación estructural que se usa en el algoritmo modifica el número de nodos ocultos y las conexiones entre los nodos de la capa intermedia y los nodos de la capa de entrada y salida, afectando, de esta forma, a la topología de la red. Hemos utilizado 5 tipos de mutaciones: añadir nodos (AN), eliminar nodos (EN), añadir conexiones (AC), eliminar conexiones (EC) utilizadas en [7], a la que nosotros añadimos la mutación unir nodos (UN).

La mutación (UN) consiste en sustituir dos nodos de la capa oculta *a* y *b* elegidos al azar por otro nuevo *c*. Se conservarán las conexiones con los nodos comunes y con una probabilidad de 0.5 las que no sean comunes. El peso de las conexiones comunes se calculará de la siguiente manera:

$$\overline{w}_c = \overline{w}_a + \overline{w}_b, \; w_{ic} = \frac{w_{ia} + w_{ib}}{2}$$

El peso de las conexiones no comunes conservará el valor que tuviera.

Para cada individuo que se le aplique la mutación estructural se le aplicarán secuencialmente cada uno de los diferentes tipos de mutación estructural [7],[9].

Con el fin de limitar la carga computacional del algoritmo, fijamos un intervalo $[\Delta_{MIN}, \Delta_{MAX}]$ para cada tipo de mutación estructural- excepto para la mutación (UN)-, donde el número de modificaciones está dado por:

$$\Delta_{MIN} + \lfloor U(0,1)T(R)(\Delta_{MAX} - \Delta_{MIN}) \rfloor \qquad \textbf{(10)}$$

Los valores mínimo y máximo establecidos para cada tipo de mutación son los siguientes:

	AN	EN	AC	EC
Δ_{MIN}	1	1	1	1
Δ_{MAX}	2	3	$3N^{(0)}$	$3N^{(0)}$

La condición de parada del algoritmo se alcanza cuando se llega a una aptitud previamente marcada, o bien, los valores $\alpha_1(t), \alpha_2(t)$ llegan a un valor próximo a cero.

3 Sección Experimental

A continuación se detalla la forma en la que hemos realizado los experimentos y los valores usados en los parámetros que aparecen en el algoritmo.

3.1 Conjunto de ejemplos.

Los $N^{(p)}$ ejemplos se han generado aleatoriamente. A partir de una función de la familia se han elegido uniformemente en el intervalo $[0, K]$ los valores de las variables independientes y se ha calculado el valor de la variable dependiente a partir de la función del modelo elegida. Para simular valores reales, se ha introducido a los valores de salida y_k el ruido gaussiano dado por:

$$y_k = y_k + N(0, r\, y_k),$$

donde los valores de r considerados han sido $r = 0.01, r = 0.03$.

Estos ejemplos servirán para entrenar las redes con el algoritmo. Igualmente generamos un conjunto de generalización con el 20% de los $N^{(p)}$ ejemplos, que nos servirá al final de la evolución para comprobar la capacidad de generalización del algoritmo.

3.2 Parámetros usados en el algoritmo evolutivo

El algoritmo ha sido probado con diferentes valores del número de patrones $N^{(p)}$. En los ejemplos considerados $N^{(p)}$ ha sido igual a 100. El dominio de definición de la función definida en (1) y usada en el proceso de modelización ha sido:

$$A = \left\{(x_1, x_2, ..., x_n) \in \mathbb{R}^{N^{(0)}} : 0 < x_i \leq 10\right\}$$

y el número de variables independientes $N^{(0)}$ consideradas en los diferentes ejemplos han sido 2, 3, y 4. Por otra parte, los parámetros que intervienen en la definición han tenido los siguientes rangos de variación: los exponentes w_{ji} han variado en el intervalo $[-5,5]$ y los coficientes $\overline{w}_j$ de cada uno de los monomios han variado en el intervalo $[-10,10]$. El número máximo de monomios considerados ha sido $N^{(1)} = 8$. El tamaño de la población N_R ha sido 1000 redes.

4. Resultados Obtenidos

A continuación reflejamos los resultados obtenidos por el algoritmo de evolución con diferentes tipos de modelos.

En la Tabla 1 aparece la función considerada en el modelo para la generación de los datos, el valor r que determina el ruido considerado, el número de generaciones que ha necesitado el algoritmo para modelizar el conjunto de datos generado, la función a la que llega el algoritmo y los valores del coeficiente de determinación R^2 de la regresión.

En la Tabla 2 se muestra el error relativo E y los valores del error cuadrático medio definido por: $SS = \frac{1}{N^{(p)}}\left[\sum_{k=1}^{N^{(p)}}(y_k - \hat{y}_k)^2\right]$ para cada uno de los casos de prueba. Los valores del error relativo y del error cuadrático medio son establecidos para el conjunto de entrenamiento y para el conjunto de generalización.

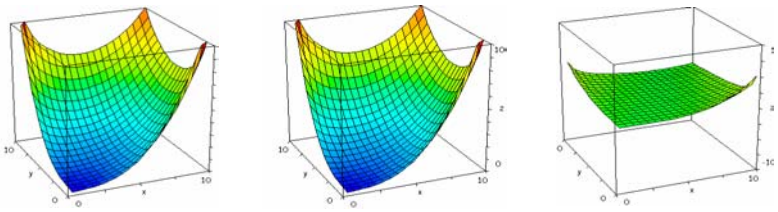
Tabla 1. Resultados del algoritmo para tres funciones con diferente nº de variables

	Función objetivo	r	R^2	Nº Gen.	Función Resultado
f_1	$x_1^3 x_2^{-0.3} + x_1^{-0.3} x_2^3$	0.01	0.9999	1000	$1.006\, x_1^{2.996} x_2^{-0.299} + 0.998\, x_1^{-0.299} x_2^3$
f_2	$5x\sqrt{y}z^2 - x^2\frac{y}{z^2} + 6\sqrt[3]{x}\sqrt{y}\frac{1}{\sqrt{z^3}}$	0.01	0.9998	5000	$6.24x^{0.326}y^{0.476}z^{-1.498} -$ $1.01x^{1.999}y^{0.991}z^{-1.997} +$ $5x^{0.999}y^{0.501}z^{1.997}$
f_3	$x_2^2 x_4^3 - \frac{x_1^2}{x_2\sqrt{x_3}}$	0.03	0,9999	2112	$0.9956x_2^2 x_3^{-0.001} x_4^{3.002} -$ $0.9942x_1^{2.003} x_2^{-1.001} x_3^{-0.5019}$

Tabla 2. Errores en los conjuntos de entrenamiento y generalización para cada ejemplo

	Conjunto de entrenamiento		Conjunto de generalización	
	Error Relativo E	*Error Cuadrático* SS	*Error Relativo* E	*Error Cuadrático* SS
f_1	0.006	1.64	0.014	2.111
f_2	0.009	36.78	0.04	163.07
f_3	0.03	2379.94	0.0772	1304.25

En el caso de la función de dos variables, hemos representado la gráfica de la función que genera los datos y la gráfica de la función que ha modelado los datos, junto con la superficie dada por las diferencia entre la función modelo $f_1(x_1, x_2)$ y la función generada $f_{1p}(x_1, x_2)$.



5 Conclusiones

La evolución de redes de regresión nos permite obtener funciones complejas, cuya modelización por otros procedimientos no es viable, con gran capacidad de entrenamiento y de generalización. Asimismo, hemos obtenido buenos resultados con diferente número de variables independientes. El algoritmo soporta datos con ruido, por lo que los resultados obtenidos son muy prometedores a la hora de su aplicación para el modelado de sistemas reales.

6 Bibliografía

1. G. Cybenko. Aproximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems, 2:302-314, 1989.
2. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal aproximators. Neural Networks, 2:359-366, 1989.
3. T. Van der Walt, E. Barnard, and J. Deventer, "Process Modeling with the Regression Network", IEEE Transactions on Neural Networks, vol. 6, No. 1, Jannuary 1995
4. D. B. Fogel, "An introduction to simulated evolutionary optimisation" IEEE Trans. On Neural Networks, vol. 5, no 1, pp. 3-14, 1994.
5. T. Bäck, U. Hammel, and H. _P. Schwefel, "Evolutionary computation: comments on the history and current state," IEEE Transactions on evolutionary Computation, vol 1, no. 1, pp. 3-17, 1997.
6. D. B. Fogel, "Using evolutionary programming to greater neural networks that are capable of playing Tic-Tac-Toe," in International Conference on Neural Networks, San Francisco, CA; IEEE Press, 1993, pp. 875-880.
7. P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary Algorithm that construct recurrent neural networks", IEEE Transactions on Neural Networks, vol. 5, no. 1, January 1994.
8. Xin Yao "Evolving Artificial Neural Networks", Proceedings of the IEEE, vol.9, no. 87, pp. 1423-1447,1999.
9. N. García, C. Hervás and J. Muñoz, "Symbiont: Cooperative coevolución of neural networks", IEEE Transactions on Neural Networks. In press.
10. S. Wang. "Neural Network techniques for monotonic nonlinear models". Computers Operation Research. vol.21 no. 2, 143-154, 1994.
11. Wang. "Nonlinear regression: a hybrid model". Computers Operation Research 26 799-817. 1999.
12. R. H. Myers and D. C. Montgomery. Response Surface Methodology: process and product optimization using designed experiments. John Wiley and Sons 2002