Algoritmo evolutivo para el reconocimiento de funciones de base potencial

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

Resumen-- Se propone un algoritmo evolutivo para el reconocimiento de funciones de base potencial. Debido a la complejidad de la superficie de error asociada a problemas de regresión no lineal, en donde son numerosos los mínimos locales, los algoritmos de búsqueda local como el algoritmo de Levenberg-Marquardt no resultan en general efectivos al quedar con frecuencia atrapados en mínimos locales. Los resultados obtenidos sin embargo por el algoritmo evolutivo muestran la capacidad de éste para evitar los mínimos locales y, en el caso de quedar atrapado en uno ellos, será en un mínimo con valor próximo al mínimo global. Por último es interesante destacar la bondad de los resultados obtenidos al introducir ruido en los datos de partida, así como la capacidad de generalización del algoritmo evolutivo y su robustez en relación con los parámetros usados en el mismo.

Palabras clave—Algoritmos evolutivos; Levenberg-Marquardt; Funciones de base potencial; regresión no lineal.

I. 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 [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 la superficie de error está plagada de mínimos locales y regiones 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, como ocurre en los algoritmos clásicos que usan el gradiente de la función de error en 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 y mutación, para introducir por una parte presión selectiva y por otra diversidad de forma tal que el algoritmo de búsqueda presente un compromiso entre su capacidad de explotar la localización de las mejores regiones y su capacidad de explorar otras localizaciones del espacio. 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 modele el conjunto de datos del fenómeno estudiado. El método propuesto consiste en aplicar las técnicas de programación evolutiva a las estructuras en red definidas a partir de la tipología de funciones considerada, evolucionando la estructura y los pesos de las conexiones de éstas con el objetivo de llegar a

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

encontrar la función dentro de la familia de funciones propuesta que mejor se ajuste al conjunto de datos. El algoritmo es comparado con el algoritmo de búsqueda local de Levenberg – Marquardt, ejecutado con diferentes condiciones iniciales elegidas aleatoriamente dentro del espacio de búsqueda. Los resultados obtenidos muestran que el algoritmo evolutivo es más eficiente que el clásico algoritmo de búsqueda local Levenberg-Marquardt.

.

II. METODOLOGÍA PROPUESTA

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

A. Tipología de Funciones y Redes Asociada

Consideremos la familia de funciones de base potencial $\mathbb{F}$ definida por:

$$\mathbb{F}=\left\{\begin{array}{l} f: A\subset \mathbb{R}^{N^{(0)}} \to \mathbb{R}: \\ f(x_1,x_2,...,x_{N^{(0)}})=\sum_{j=1}^{N^{(1)}} \overline{w}_j (\prod_{i=1}^{N^{(0)}} x_i^{w_{ji}}) \end{array}\right\} \qquad (1)$$

donde

$$\overline{w}_j \in [-M,M]\subset \mathbb{R},\ w_{ji}\in [-L,L]\subset \mathbb{R},$$
$$i=1,2,...,N^{(0)},\ j=1,2,...,N^{(1)}$$

siendo el dominio de definición de f el subconjunto A de $\mathbb{R}^{N^{(0)}}$ dado por

$$A=\left\{(x_1,x_2,...,x_{N^{(0)}})\in \mathbb{R}^{N^{(0)}} : 0<x_i\le K\right\}.$$

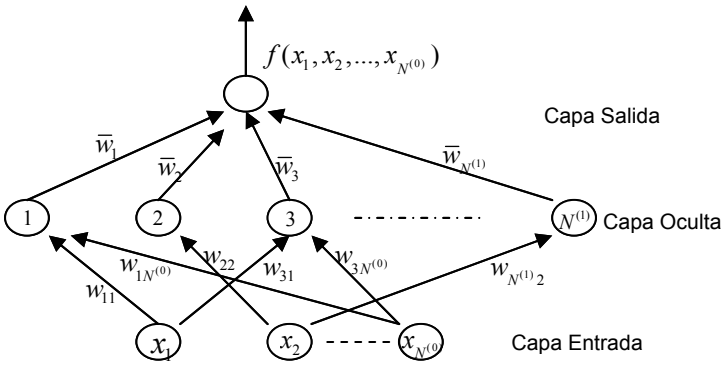
Cada función de la familia puede verse como una expresión polinómica en varias variables en la que los exponentes de cada variable son números reales. En particular, para dos variables independientes, las funciones de la familia $\mathbb{F}$ tienen la siguiente expresión:

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

Por otra parte, es interesante observar que las funciones de base potencial puede considerarse una generalización de las superficies de respuesta [12]. Observemos, por ejemplo, que si en (1) se eligen convenientemente los exponentes $w_{ij}\in\{0,1,2\}$ se obtiene una superficie de respuesta cuadrática del tipo:

$$f(x_1,x_2,...,x_{N^{(0)}})=c_0+\sum_{i=1}^{N^{(0)}} c_i x_i+\sum_{i=1}^{N^{(0)}} c_{ii}x_i^2+\sum_{i<j=1}^{N^{(0)}} c_{ij}x_i x_j$$

Veamos ahora como cualquier función de la familia $\mathbb{F}$ puede representarse mediante una estructura de red [4] [5] [6]. Las redes 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 del modelo, $(x_1,x_2,...,x_{N^{(0)}})$, $N^{(1)}$ 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

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

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

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

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, cada función de la familia $\mathbb{F}$ queda representada por la estructura de red que acabamos de definir.

B. Algoritmo evolutivo

1) Estructura general del algoritmo

En primer lugar construimos la población inicial B de tamaño N_R. A partir de la población inicial B, el algoritmo se estructura, siguiendo la metodología propuesta por Deb [3], de la siguiente forma:

Paso 1: Fase de Selección. Seleccionamos el $r\%$ ($r = 90$) de individuos con mejor aptitud de la población $B^* = B - \{mejores\ individuos\ de\ B\}$ de cardinal $N_R^* = N_R - 1$ y formamos una población P de tamaño $rN_R^*/100$.

Paso 2: Fase de Generación. Aplicamos mutación estructural a cada elemento de la población P y mutación paramétrica sólo a los $(100-r)N_R^*/100$ mejores individuos de P -los individuos a los que se les aplica la mutación paramétrica corresponden en realidad con el $(100-r)\%$ de mejores individuos de la población B^*-. La población obtenida después del plan de generación la denominaremos C. Es fácil ver que el cardinal de C es igual a $N_R^* = N_R - 1$.

Paso 3: Fase de Reemplazamiento. Elegimos todos los elementos de B^* para ser reemplazados.

Paso 4: Fase de Sustitución. Los elementos de B^* son reemplazados por los elementos de C. Por tanto la nueva población es $B = \{Mejores\ de\ B\} \cup C$. De esta forma el tamaño de la población se mantiene constante durante la evolución.

2) Generación de la población inicial

El algoritmo se inicia generando aleatoriamente un número mayor de redes que el usado posteriormente en el proceso de evolución, concretamente generamos $10 * N_R$ redes, siendo N_R el número de redes que tendrá la población durante el proceso de evolución.
Para la generación aleatoria de una red, se comienza eligiendo el número de nodos ocultos a partir de una distribución uniforme en el intervalo $(0, N^{(1)}/2]$, donde $N^{(1)}$ corresponde al número máximo de nodos ocultos de cada una de las redes de la población. De esta forma formamos la población inicial con modelos más sencillos. 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 $(0, N^{(0)}]$, siendo $N^{(0)}$ el número de variables independientes. Siempre se añade una conexión entre el nodo de la capa oculta y el nodo de la capa de salida. 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. Por último, la población inicial que constituye la solución base B se forma seleccionando las N_R redes mejores entre las $10 * N_R$ generadas.

3) Asignación de la aptitud

Sea $D = \{(\mathbf{x_i}, \hat{y}_i) : i = 1, 2, ..., n\}$ el conjunto de datos de entrenamiento, donde $N^{(p)}$ es el número de ejemplos. Se asume que cada ejemplo de entrenamiento $(\mathbf{x_i}, \hat{y}_i)$ es independiente e idénticamente distribuido. Consideramos el error relativo $E(g)$ de cada individuo g de la población como:

$$E(g) = \frac{1}{n}\left[\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{|\hat{y}_i|}\right]^{1/2}$$

donde $\hat{y}_i$ son los valores esperados.
Por otra parte, se define la función aptitud $A(g)$ como una transformación estrictamente decreciente del error:

$$A(g) = \frac{1}{1 + E(g)}$$

4) Mutación paramétrica

La mutación paramétrica es un operador basado en técnicas de enfriamiento simulado [13],[14] y [15]. La severidad de la mutación que se le aplica a un individuo g se basa en la función temperatura $T(g)$ dada por:

$$T(g) = 1 - A(g) \quad \text{, donde}$$
$$0 \le T(g) < 1$$

De esta manera, la temperatura está determinada por lo alejado que nos encontremos de la solución del problema. Las mutaciones paramétricas se aplican a cada parámetro $w_{ji}, \overline{w}_j$ de (1) con ruido gausiano, donde la varianza de la distribución normal depende de la temperatura. Así, redes con temperatura alta son mutadas severamente mientras que redes con temperatura baja sufrirán mutaciones suaves. Esto permite una búsqueda menos afinada inicialmente, pero conforme nos acercamos a la solución la búsqueda será más afinada. Particularmente, los exponentes w_{ji} de la función, que están representados con los pesos de las conexiones entre los nodos de la capa de entrada y los nodos de la capa oculta son modificados de la siguiente manera:

$$w_{ji}(t+1) = w_{ji}(t) + \xi_1(t),$$
$$i = 1, ..., N^{(0)},\ j = 1, ..., N^{(1)}$$

donde $\xi_1(t) \sim N(0, \alpha_1(t)T(R))$ representa una distribución normal de una variable de media 0 y varianza $\alpha_1(t)T(R)$.

Mientras que los coeficientes $\overline{w}_j$ de la función, que representan los pesos de las conexiones entre los nodos de la capa oculta y el nodo de la capa de salida, se modifican como sigue:

$$\overline{w}_j(t+1) = \overline{w}_j(t) + \xi_2(t),$$
$$j = 1, ..., N^{(1)}$$

donde $\xi_2(t) \sim N(0, \alpha_2(t)T(R))$ representa una distribución normal de una variable de media 0 y varianza $\alpha_2(t)T(R)$.

Cuando las mutaciones son realizadas, la aptitud del individuo g se recalcula y se aplica la técnica de enfriamiento simulado. Siendo ΔA el incremento de la función de aptitud antes y después de la mutación: Si $\Delta A \geq 0$ se acepta el cambio y si $\Delta A < 0$ se acepta con una probabilidad igual a $\exp(\Delta A / T(g))$.

Obsérvese que la severidad de la modificación de los exponentes w_{ji} ha de ser diferente a la de los coeficientes $\overline{w}_i$, por tanto $\alpha_1(t) \ll \alpha_2(t), \quad \forall t \in \mathbb{N}$, los cuales son cambiados adaptivamente en cada generación de acuerdo a una regla prefijada. Los parámetros α_1, α_2 que junto a la temperatura determinan las varianzas de la distribuciones normales, van cambiando durante el proceso de evolución, por lo que se realiza un aprendizaje adaptativo, a diferencia de lo que ocurre en [7], [9] donde estos parámetros permanecen fijos durante toda la evolución. De esta forma evitamos quedar atrapados en mínimos locales y aceleramos el proceso de evolución cuando las condiciones del proceso lo permitan. Concretamente, la evolución de los parámetros α_1, α_2 viene dada por:

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

$i = 1, 2$, donde $A(s)$ representa la aptitud de la red con aptitud máxima en la generación s.

Teniendo en cuenta que una generación es satisfactoria si el mejor individuo de la población es mejor que el mejor de la anterior generación. Si observamos varias generaciones satisfactorias consecutivamente, es indicativo de que los mejores de la población se encuentran en una buena región dentro del espacio de búsqueda. En este caso, incrementamos el paso de la mutación esperando encontrar mejores soluciones alrededor de la solución óptima. Si la aptitud del mejor individuo es constante durante varias generaciones consecutivas decrementamos la longitud de la mutación. En otros casos, mantenemos constante dicho paso.

5) Mutación Estructural

La mutación estructural es más compleja, pues implica una modificación en la estructura de la función. La mutación estructural permite realizar la exploración de diferentes regiones en el espacio de búsqueda y mantener la diversidad de la población. Utilizamos cinco mutaciones estructurales diferentes: añadir nodos (AN), eliminar nodos (EN), añadir conexiones (AC), eliminar conexiones (EC) y unir nodos (UN) que se aplican secuencialmente a cada red. Las cuatro primeras son similares a las utilizadas en el modelo GNARL [7]. Añadir nodo, AN: El nodo de la capa oculta se añade con conexiones entre el nodo de la capa oculta y el nodo de la capa de salida con pesos aleatorios y conexiones aleatorias con los nodos de la capa de entrada. Eliminar nodo, EN: Seleccionamos un nodo de la capa oculta y lo eliminamos junto a sus conexiones. Añadir conexiones, AC: Elegimos aleatoriamente un nodo de la capa de entrada y de la capa oculta y establecemos una conexión entre ellos con peso aleatorio. Eliminar conexión, EC: Elegimos aleatoriamente una conexión entre un nodo de la capa de entrada y un nodo de la capa oculta y lo borramos. Por último, la mutación Unir nodos, UN, consiste en sustituir dos nodos de la capa oculta *a* y *b* elegidos aleatoriamente por otro nodo *c*. Las conexiones comunes se conservan y las no comunes se conservan con una probabilidad de 0.5. Los pesos de las conexiones no comunes se mantienen y los pesos de las conexiones comunes se calculan de la siguiente manera:

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

Para cada mutación (excepto en la mutación unir nodos) hay un valor mínimo, Δ_m y un valor máximo Δ_M. El número de elementos (nodos y conexiones) involucrados en la mutación se calcula según la siguiente expresión:

$$\Delta_{MIN} + \lfloor uT(R)(\Delta_{MAX} - \Delta_{MIN}) \rfloor$$

donde *u* es una variable aleatoria uniformemente distribuida en [0,1]. Todas las mutaciones anteriores son aplicadas secuencialmente y en el orden indicado con probabilidad *T(g)* en la misma generación y para cada red. En el caso de que la probabilidad indique que no se aplique ninguna

mutación, se aplicará una de las mutaciones elegida al azar.

6) Criterio de Parada

La condición de parada del algoritmo se alcanza cuando se llega a una aptitud previamente marcada, o bien, los valores $\alpha_1(t)$ y $\alpha_2(t)$ llegan a un valor próximo a cero, o bien no mejora durante 20 generaciones consecutivas ni la media del 10% mejor de la población ni el mejor de la población.

III. Sección Experimental

A continuación mostramos varios ejemplos de funciones de la familia considerada que han sido reconocidas por el algoritmo evolutivo descrito anteriormente, así como la forma en la que han sido realizados los experimentos y los valores usados en los diferentes parámetros que aparecen en el algoritmo.

A. Conjunto de Ejemplos

A partir de la función de la familia que se desea reconocer, se han generado aleatoriamente $N^{(p)}$ ejemplos tomando cada variable en el intervalo $[0, K]$ y se han calculado los valores correspondientes de las variables dependientes.
Estos ejemplos constituyen la única información proporcionada al algoritmo para que realice la búsqueda de la función considerada. De la misma forma a como se han generado los ejemplos para el entrenamiento, se genera un conjunto de generalización con cardinal igual al 20% del número de ejemplos $N^{(p)}$, que finalizado el proceso de evolución servirán para comprobar la capacidad de generalización del algoritmo.
En una de las funciones con las que hemos trabajado, para simular valores reales y mostrar el funcionamiento del algoritmo en este caso, se ha introducido a los valores de salida de los ejemplos de entrenamiento y de generalización, el ruido gaussiano dado por:

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

1) Parámetros utilizados en el algoritmo evolutivo.

El número de ejemplos $N^{(p)}$ considerados en el algoritmo ha sido igual a 100. El dominio de definición de las funciones definidas en (1) y usadas en el proceso de modelización ha sido:

$$A = \left\{ (x_1, x_2, ..., x_n) \in \mathbb{R}^{N^{(0)}} : 0 < x_i \le 5 \right\}$$

y el número de variables independientes $N^{(0)}$ consideradas en los diferentes ejemplos han sido 3 y 4. Por otra parte, los parámetros que intervienen en la definición de cada función de la familia han tenido los siguientes rangos de variación: los exponentes w_{ji} han variado en el intervalo $[-5,5]$ y los coeficientes $\overline{w}_j$ 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. Por último, los valores considerados para los parámetros asociados a las mutaciones paramétricas $\alpha_1(0), \alpha_2(0), \beta, r, \rho$ en todos los experimentos realizados han sido: $\alpha_1(0) = 1, \alpha_2(0) = 5, \beta = 0.1, r = 90, \rho = 10$. Mientras que el intervalo $[\Delta_{\min}, \Delta_{\max}]$ que determina la severidad en las diferentes mutaciones estructurales está dado para cada mutación por: AN= $[1,2]$, EN=$[1,3]$, AC=$[1, 2N^{(1)}]$ y EN=$[1, 2N^{(1)}]$.

B. Resultados obtenidos

A continuación reflejamos los resultados obtenidos por el algoritmo evolutivo con diferentes funciones de base potencial y los resultados obtenidos por el algoritmo de Levenberg-Marquardt. El algoritmo evolutivo ha sido ejecutado 10 veces en cada función. A partir de la función obtenida al final del proceso de evolución, se han calculado para el conjunto de entrenamiento y para el conjunto de generalización los valores del error cuadrático medio definido por: $SS = \frac{1}{N^{(p)}} \left[\sum_{k=1}^{N^{(p)}} (y_k - \hat{y}_k)^2 \right]$ y el error relativo (SEP) dado por la expresión:

$$SEP = \frac{100}{\overline{y}} \sqrt{\frac{\sum_{i=1}^{N^{(0)}} (y_i - \hat{y}_i)^2}{n}}$$

donde y_i corresponde a la predicción realizada del valor teórico $\hat{y}_i$, mientras que $\overline{y}$ representa la media aritmética de los valores teóricos.
Por otra parte, para comparar la eficiencia del algoritmo con un algoritmo clásico de búsqueda local, se ha elegido el algoritmo de Levenberg-Marquardt. La información proporcionada al algoritmo de Levenberg-Marquardt ha sido la tipología de la función de partida, determinando el número de sumandos, así como el mismo conjunto de ejemplos de entrenamiento generados a partir de dicha función y usado por el algoritmo evolutivo. El algoritmo de Levenberg-Marquardt se ha ejecutado

10 veces partiendo de 10 condiciones iniciales diferentes, una de ellas el origen de coordenadas y las nueve restantes elegidas aleatoriamente dentro del espacio de búsqueda de los coeficientes del modelo.

En las tablas 1, 2, 3 y 4 se muestran los resultados obtenidos por el algoritmo evolutivo y por el algoritmo de Levenberg-Marquardt para cada una de las funciones consideradas. Los resultados comparados de ambos algoritmos contienen los valores medios y la desviación típica del error cuadrático medio *SS* y del error relativo SEP efectuadas las 10 ejecuciones de cada algoritmo, tanto para el conjunto de entrenamiento como para el conjunto de generalización, así como el mejor y el peor de los resultados obtenidos.

TABLAS DE RESULTADOS:
ALGORITMOS EVOLUTIVO Y LEVENBERG-MARQUARDT

	ALGORITMO EVOLUTIVO				LEVENBERG-MARQUARDT			
	ENTRENAMIENTO		VALIDACIÓN		ENTRENAMIENTO		VALIDACIÓN	
	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP
MEDIA	0	0	0.068	0	1459.8	-13.052	3875.2	88.11
SD	0	0	0.001	0	2187.6	19.150	2.36	104.4
BEST	0	0	0	0	0	0	0	0
WORST	0	0	0	0	5340.4	-57	3872	198.21

TABLA 1 $f_1(x,y,z,t) = x^{-2}y^{-2} - 2z^2w^2 - 1.3\sqrt{x}\sqrt[3]{y}z^{-2}$

	ALGORITMO EVOLUTIVO				LEVENBERG-MARQUARDT			
	ENTRENAMIENT0		VALIDACIÓN		ENTRENAMIENTO		VALIDACIÓN	
	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP
MEDIA	38.54	0.24	0.029	5.403	31307452	394.628	808699	17511.9
SD	0.506	$4.47\ 10^{-2}$	$4.47\ 10^{-4}$	$4.47\ 10^{-2}$	$1.32\ 10^{7}$	477.27	$1.09\ 10^{6}$	$2.38\ 10^{4}$
BEST	38.96	0.649	0.028	5.288	16.22	0.41	0.2	13.91
WORST	42.44	0.677	0.037	6.19	77300718	914.44	2041675	44152

TABLA 2 $f_1(x,y,z,t) = x^{-2}y^{-2} - 2z^2w^2 - 1.3\sqrt{x}\sqrt[3]{y}z^{-2}$ (ruido $\eta = 0.03$)

	ALGORITMO EVOLUTIVO				LEVENBERG-MARQUARDT			
	ENTRENAMIENTO		VALIDACIÓN		ENTRENAMIENTO		VALIDACIÓN	
	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP
MEDIA	10.85	5.7	15.09	5.11	18.55	9.52	5041	30.06
SD	18.78	7.35	20.58	5.67	15.99	7.38	14938	59.21
BEST	0	$9.3\ 10^{-5}$	0	$7\ 10^{-5}$	0	0	0	0
WORST	58.54	21.31	47.20	13.25	25.66	13.88	328.4	35.47

TABLA 3 $f_2(x,y,z,t) = x^2y - zw + x^3$

	ALGORITMO EVOLUTIVO				LEVENBERG-MARQUARDT			
	ENTRENAMIENTO		VALIDACIÓN		ENTRENAMIENTO		VALIDACIÓN	
	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP	SS/N	%SEP
MEDIA	5.83	-0.192	0.068	0.167	11092618	-255.38	10068.4	59.59
SD	15.85	0.286	0.118	0.158	$1.78\ 10^{7}$	401.4	$1.7\ 10^{4}$	65.47
BEST	$2.9\ 10^{-4}$	-0.002	$1.3\ 10^{-5}$	0.003	22.27	-0.72	41.09	5.5
WORST	50.89	-0.983	0.025	0.136	36937321	-837	38839	169.09

TABLA 4 $f_3(x,y,z,t) = 5x\sqrt{y}z^2 - x^2yz^{-2} + 6\sqrt[3]{x}\sqrt{y}\dfrac{1}{\sqrt{x^3}}$

De la observación de los resultados que aparecen en las tablas anteriores se pueden destacar los siguientes hechos:

En las tres funciones estudiadas, el algoritmo evolutivo ha conseguido reconocer la función de partida en alguna de las ejecuciones (en el caso de la función f_1 en todas las ejecuciones), mientras que el algoritmo de Levenberg-Marquardt ha reconocido las funciones f_1 y f_2 en alguna de las ejecuciones y no ha podido reconocer la función f_3 en ninguna de las 10 ejecuciones efectuadas.

Los resultados obtenidos por el algoritmo de Levenberg-Marquardt muestran una gran dependencia respecto de las condiciones iniciales consideradas, quedando el algoritmo atrapado en varias ocasiones en mínimos locales.

Comparando las desviaciones típicas de los resultados obtenidos por los dos algoritmos se observa que la desviación típica en el algoritmo evolutivo es menor en todos los ejemplos que la obtenida por el algoritmo de Levenberg-Marquardt.

Cuando se introduce ruido en los datos generados por la función (véase la tabla 2) los resultados medios del algoritmo evolutivo son mejores que los que obtiene el algoritmo de Levenberg-Marquardt.

Otro hecho resaltable es que en aquellos casos en los que el algoritmo de Levenberg-Marquardt ha conseguido unos resultados aceptables tanto en la media cuadrática como en el SEP para el conjunto de entrenamiento, los resultados en el conjunto de generalización no han sido buenos. Esto no ocurre con los modelos obtenidos por el algoritmo de evolución en el que los resultados para los conjuntos de entrenamiento y de generalización son similares. Se puede afirmar a partir de los resultados comparados que los modelos obtenidos por el algoritmo evolutivo generalizan mejor que los que resultan del algoritmo de Levenberg-Marquardt.

IV. CONCLUSIONES

El algoritmo evolutivo que se propone resulta ser una herramienta eficaz para ser utilizada en la regresión de modelos que respondan a la tipología de funciones considerada. La complejidad de la superficie de error asociada a problemas de regresión no lineal, en donde son numerosos los mínimos locales y las regiones llanas, es el motivo por el cual algoritmos de búsqueda local basados en el gradiente de la función de error, como el algoritmo de Levenberg-Marquardt, no resulten en general efectivos al quedar con frecuencia atrapados en mínimos locales. Los resultados obtenidos sin embargo por el algoritmo evolutivo muestran la capacidad de éste para evitar los mínimos locales y, en el caso de quedar atrapado en uno ellos, conseguir que sea en un mínimo con valor próximo al mínimo global. Por último es interesante destacar la bondad de los resultados obtenidos al introducir ruido en los datos de partida, así como la capacidad de generalización del algoritmo evolutivo y su robustez en relación con los parámetros usados en el mismo.

V. REFERENCIAS

[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] Deb, K. "A population-based algorithm-generator for real-parameter optimization". Soft Computing. Springer. (In press).2003

[4] Durbin, R, Rumelhart, D. "Products Units: A computationally powerful and biologically plausible extension to backpropagation networks". Neural Computation, 1,133-142.1989

[5] Leerink, L.R.,Giles, C.L.,Horne,B.G.,Jabri, M.A. "Learning with products units", Advances in Neural Information Processing Systems, 7, 537, 1995.

[6] Gurney, K.N., "Training Nets of Hardware Realizable Sigma-Pi Units", Neural Networks, Vol. 5, No 2, pp. 289-303, 1992.

[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, "COVNET: A Cooperative coevolutionary model for evolving neural networks", IEEE Transactions on Neural Networks, vol 14, no 3, Mayo, 2003.

[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

[13] Geyer, C.J. & Thompson, E. A. *Annealing Markov Chain Monte Carlo with applications to ancestral inference.* J. Amer. Statist. Assoc. pp 909-920.1996

[14] Kirkpatrik, S. Gelatt, C. D. and Vecchi, M. P. "Optimization by simulated annealing". Science 220, 671-680.1983

[15] Otten, R. H. J. M. and van Ginneken, L.P.P.P. *The annealing algorithm*. Ed. Kluwer. Boston, MA, 1989