Regresión no lineal mediante la evolución de modelos Híbridos de Redes Neuronales[1]

César Hervás
Departamento de Informática y A.N. Universidad de Córdoba
14071- Córdoba-
{chervas}@uco.es

Francisco J. Martínez
Facultad de Ciencias Económicas y Empresariales (ETEA). Universidad de Córdoba
{fjmestud}@etea.com.

Pedro A. Gutiérrez, Aarón Ruiz
Departamento de Informática y A.N. Universidad de Córdoba
14071- Córdoba
{i02gupep,i02rumoa}@uco.esT

Resumen

El presente trabajo es una primera aproximación a la formación de modelos de redes neuronales con unidades ocultas de tipo híbrido (sigmoides, producto) que siendo aproximadores universales, puedan utilizarse como modelos no lineales de regresión cuando las características del espacio de las variables independientes lo aconsejen. Dada la dificultad que presenta la aplicación de algoritmos de aprendizaje de búsqueda local para esta tipología de modelos, se utiliza un algoritmo de programación evolutiva donde se definen operadores de mutación específicos. Los experimentos realizados con cuatro funciones de prueba, las tres funciones de Friedman y una propuesta por los autores, muestran resultados muy prometedores en esta dirección.

1. Introducción

El análisis de métodos de regresión no lineal es una de las áreas de mayor interés en la actualidad en el campo del modelado de sistemas dinámicos reales en diferentes áreas de investigación. Asociado a este análisis se han desarrollado en los últimos años diferentes modelos de redes neuronales artificiales para regresión y clasificación, redes de tipo perceptrón multicapa (MLP) basadas en la utilización de unidades de base sigmoide, US, como funciones de transferencia en los nodos de la capa oculta de la red; funciones de base radial (RBF), también llamadas funciones de ventana radial, redes neuronales de regresión generales (GRNN) desarrolladas por Specht [1], y redes multiplicativas, y dentro de ellas, redes de unidades de base producto, UP, [2]. Por otra parte, a pesar del carácter de aproximadores universales de las redes neuronales basadas en unidades de diferente tipo (MLP, RBF y UP), la velocidad de aprendizaje y la complejidad de la red final pueden diferir de forma significativa de un caso a otro. La velocidad de convergencia y la complejidad de las redes utilizadas se convierte en uno de los principales problemas a resolver. Este problema ha sido abordado desde diferentes puntos de vista. Una de las propuestas en este sentido ha sido la formación de modelos híbridos con el diseño de redes neuronales con diferentes tipos de unidades en la capa oculta.

Así, se han definido modelos híbridos donde los nodos de la capa oculta tienen diferentes funciones de activación/transferencia. Entre estos trabajos, merece la pena destacar, por una parte, los trabajos de Duch y Jankowski [3], en los que se proponen diferentes funciones de transferencia en los nodos de la capa oculta y, por otra, la propuesta de Cohen-Intrator, [4]-[5], que contemplan la dualidad entre funciones que utilizan aproximaciones de funciones basadas en proyección, (funciones de base sigmoide, de base producto, etc) y funciones de base radial. Concretamente la técnica utilizada se basa en un agrupamiento de los datos en clases (que corresponde a la división del espacio de las variables independientes en regiones), la selección del tipo de nodo a partir de técnicas Bayesianas (BIC, Bayesian Information Criteria) o de la técnica MDL (minimum description length) y la poda del modelo a partir del análisis de sensibilidad de los parámetros del mismo. Desde un punto de vista analítico es de destacar en este contexto el trabajo de [6] donde se proporciona una base de tipo teórico a esta descomposición,

[1]Este trabajo ha sido financiado por el MCYT, proyecto TIC2002-04036-C05-02 y con fondos FEDER.

probándose que una función continua se puede descomponer en dos funciones mutuamente excluyentes, una de tipo radial y la otra parte de tipo cresta (basada en la proyección); aunque es difícil separar la parte radial de una función y a continuación proceder a estimar la aproximación funcional basada en una proyección y no quedar atrapados en óptimos locales en el procedimiento de minimización de los errores cometidos [7]. En general, no hay ningún resultado de tipo teórico que muestre que cualquier función puede ser descompuesta en dos partes excluyentes asociadas a otras topologías de proyección diferentes a las anteriores.

El propósito de este trabajo es mostrar las primeras investigaciones sobre la utilización de modelos híbridos asociados a dos tipos específicos de funciones como aproximadores funcionales de tipo proyección: las funciones de base unidades producto y las funciones de base unidades sigmoide, enfocados a la resolución de problemas de regresión. Por otra parte, mientras que en la mayoría de las investigaciones desarrolladas para determinar modelos híbridos se utilizan métodos basados en el gradiente para la optimización de los parámetros del modelo [7], en nuestro trabajo utilizamos algoritmos de computación evolutiva, dada la complejidad de las funciones de error obtenidas cuando consideramos como objetivo la minimización de los errores cuadráticos cometidos al ajustar los modelos de regresión a los datos de entrenamiento generados mediante las funciones de prueba. Aunque somos conscientes de las mejoras que podrían obtenerse incorporando al algoritmo evolutivo un algoritmo de búsqueda local que permitiera afinar la precisión y mejorar la velocidad de convergencia, hemos optado en este primer estado de la investigación por no hibridar el algoritmo evolutivo. El modelo propuesto ha sido evaluado en cuatro funciones de prueba: las tres utilizadas por Friedman y una nueva, especialmente compleja, definida por nosotros. Los resultados obtenidos muestran un rendimiento comparable al de otras técnicas de regresión consolidadas.

2. Características de las funciones base de tipo sigmoide y de tipo producto

En primer lugar vamos a describir brevemente alguna de las características de los dos tipos de funciones de base, las funciones potenciales y las funciones sigmoides, que vamos a utilizar en los modelos híbridos. Las primeras tienen como funciones de base,

$$B_k(\mathbf{x},\mathbf{w}_k)=\prod_{i=1}^{p} x_i^{w_{ki}} \text{ , k= 1, …, } m_2 \text{ ;} \qquad (1)$$

mientras que las segundas son de la forma

$$B_j(\mathbf{x},\mathbf{u_j})=\frac{1}{1+\exp\left(-u_{j0}-\sum_{i=1}^{p} u_{ji}x_i\right)},$$

para j=1,…,m (2)

Nos planteamos a continuación las características que debe de tener un problema de regresión para que se ajuste mejor con unidades de tipo producto versus unidades de tipo sigmoide (el hecho teórico de que ambas funciones de base sean aproximadores universales no implica que toda función continua sea fácilmente ajustable a través de una combinación lineal de ellos). La cuestión principal es si a partir de los datos, desconociendo la función que los generó, podemos determinar algunas propiedades de la función que permitan elegir un modelo u otro, o bien la mezcla "inteligente" de diferentes modelos. La respuesta a esta cuestión es compleja y no será respondida en este trabajo que pretende ser una primera aproximación a este problema desde la hibridación de modelos. Desde un punto de vista teórico señalaremos dos aspectos importantes relacionados con las familias de funciones que vamos a hibridar: propiedades de acotación y propiedades relacionadas con la capacidad de generalización del modelo híbrido propuesto.

En primer lugar, entendemos que una de las propiedades que se llevan mejor con las unidades producto es que la función subyacente a los datos sea no acotada. Si consideramos como dominio de definición todo el espacio R^n, las funciones MLP y RBF están acotadas superior e inferiormente. Sin embargo una función basada en unidades PU no está acotada:

Si $f_{PU}(\mathbf{x})=\sum_{j=1}^{m}\beta_j\prod_{i=1}^{p} x_i^{w_{ji}}$ entonces se tiene que:

$$\lim_{\|x\|\to+\infty} f_{PU}(\mathbf{x})=\lim_{\|x\|\to+\infty}\sum_{j=1}^{m}\beta_j\prod_{i=1}^{p} x_i^{w_{ji}}=\infty \qquad (3)$$

para determinadas elecciones de los parámetros del modelo. Sin embargo, para las unidades sigmoides, MLP, y las RBF

$$\left|f_{MLP}(\mathbf{x})\right| = \left|\sum_{j=1}^{m} \beta_j \frac{1}{1+e^{-\langle \mathbf{x},\mathbf{w}\rangle}}\right| \leq \sum_{j=1}^{m} \left|\beta_j\right| \qquad (4)$$

$$\left|f_{RBF}(\mathbf{x})\right| = \left|\sum_{j=1}^{m} \beta_j e^{-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{r_j}}\right| \leq \sum_{j=1}^{m} \left|\beta_j\right| \qquad (5)$$

Algo parecido se puede afirmar en el caso de las funciones derivadas parciales de las funciones MLP con respecto a cada variable. Estas funciones son acotadas.

$$\left|\frac{\partial f_{MLP}(\mathbf{x})}{\partial x_i}\right| = \left|\sum_{j=1}^{m} \beta_j \frac{w_i e^{-\langle \mathbf{x},\mathbf{w}\rangle}}{(1+e^{-\langle \mathbf{x},\mathbf{w}\rangle})^2}\right| \leq$$

$$\sum_{j=1}^{m} \left|\beta_j\right| \left|\frac{w_i e^{-\langle \mathbf{x},\mathbf{w}\rangle}}{(1+e^{-\langle \mathbf{x},\mathbf{w}\rangle})^2}\right| \leq |w_i| \sum_{j=1}^{m} \left|\beta_j\right| \qquad (6)$$

Normalmente, los datos que se tienen como información para la resolución de un problema de regresión pertenecen a un conjunto compacto, que es en donde se realiza el modelado. En este conjunto, siempre es posible aplicar cualquiera de los modelos MLP o RBF. Sin embargo, la capacidad de generalización para datos fuera del dominio compacto en el que se encuentran los datos de entrenamiento no sería buena en el caso de funciones no acotadas.

De esta forma una característica que puede influir en la decisión sobre qué modelo de red es el adecuado es el comportamiento en los bordes del dominio. Una hipótesis sería la siguiente: si existe gran diferencia entre el valor de los datos en los bordes del dominio y los valores dentro del dominio, posiblemente una estructura de modelado a través de redes de tipo PU se ajuste mejor. (Esto podría estar relacionado con el comportamiento de las derivadas parciales de la función).

El segundo aspecto es la capacidad de generalización del modelo híbrido propuesto. Al considerar una familia de funciones formada por la unión de las funciones de base sigmoide y las funciones basadas en unidades producto se mejora la capacidad de reducir el riesgo empírico, es decir, el error obtenido con los datos de entrenamiento, sin embargo, se plantea la duda de la capacidad de generalización de la nueva familia de funciones. En general, a mayor complejidad de la familia de funciones menor suele ser la capacidad de generalización de los modelos obtenidos. La capacidad de generalización está relacionada con la dimensión de Vapnik-Chervonenkis (VC), [8]. Concretamente, a mayor complejidad mayor valor de la dimensión VC y menor capacidad de generalización. A este respecto es interesante señalar que Schmith [9] obtiene cotas superiores de la dimensión VC para redes con unidades mixtas de tipo sigmoide y producto similares a las cotas conocidas para las redes neuronales con unidades sólo de tipo sigmoide o de tipo producto. Este hecho garantiza una capacidad de generalización de la familia mixta similar a la de las redes de base sigmoide (MLP) o a las redes basadas en unidades producto.

3. Métodos adaptativos para la estimación de funciones

En este trabajo mostramos las primeras investigaciones en la estimación de funciones mediante combinaciones lineales de funciones de base sigmoide y potencial. Estamos interesados en la estimación de una función de p variables predictoras **x**, a partir de un conjunto de n puntos o medidas $z_i = (\mathbf{x_i}, y_i)$, $i = 1,2,...,n$ en un espacio n-dimensional muestral, donde $y = f(\mathbf{x}) + \varepsilon$, con ε una variable aleatoria de error con media 0 y cuya distribución puede depender de **x**, siendo además desconocida la distribución de **x** en el conjunto de entrenamiento.

Vamos a utilizar métodos de estimación no paramétrica adaptativos de forma tal que podamos estimar la función como una combinación lineal de funciones de base, en la forma:

$$f(\mathbf{x}) = \sum_{j=1}^{M} \beta_j B_j(\mathbf{x}, \mathbf{w_j}), \qquad (7)$$

donde $\mathbf{x} = (x_1, x_2, ..., x_p)$ es el vector de las variables de entrada, β_j son los coeficientes de la combinación lineal que se van a estimar a partir de los datos, $B_j(\mathbf{x}, \mathbf{w_j})$ son las funciones de base, siendo $B_0(\mathbf{x}, \mathbf{p_0}) = 1$ para poder introducir sesgo en el modelo, $\mathbf{w_j} = (w_{j1}, w_{j2}, ..., w_{jp})$ son los

parámetros asociados a las funciones de base y M es el parámetro de regularización del modelo, el cual esta asociado al número de funciones de base que son necesarias y suficientes para minimizar alguna función del error definida al respecto.

El modelo híbrido que proponemos para estimar la función está dado por una combinación lineal de las funciones de base sigmoides y producto:

$$f(\mathbf{x}) = \sum_{j=1}^{m_1} \alpha_j B_j(\mathbf{x}, \mathbf{u_j}) + \sum_{k=1}^{m_2} \beta_k B_k(\mathbf{x}, \mathbf{w}_k) \quad (8)$$

Dado que ambos tipos de funciones de base son aproximadores universales [10], [11], el método consiste en encontrar un número suficiente de funciones de base (su arquitectura) para proporcionar una función de tipo aproximador universal a la función a estimar, de forma tal que, , podamos para cada $\varepsilon > 0$ encontrar un valor de m_1, m_2 y estimadores de los parámetros α_j, β_k, $\hat{\mathbf{u}}_\mathbf{j}$ y $\hat{\mathbf{w}}_k$ para $j = 1, 2, ..., m_1$ y $k = 1, 2, ..., m_2$ tal que:

$$\left\| f(\mathbf{x}) - \sum_{j=1}^{m_1} \alpha_j B_j(\mathbf{x}, \mathbf{u_j}) + \sum_{k=1}^{m_2} \beta_k B_k(\mathbf{x}, \mathbf{w}_k) \right\| < e \quad (9)$$

Estamos por tanto ante un problema de optimización similar a los modelos de regresión mediante proyecciones apropiadas "projection pursuit" con la característica especial de que las funciones consideradas son exclusivamente de dos tipos. Para resolver este problema NP-complejo de obtención de la arquitectura y de estimar los coeficientes del modelo utilizaremos algoritmos de computación evolutiva del tipo de los descritos en los trabajos de Fogel [12], Angeline et al. [13] y N. García, C. Hervás et al [14], donde tan sólo citaremos el esquema fundamental insistiendo en la diferencias fundamentales con los algoritmos citados

4. Algoritmo evolutivo

La idea básica del algoritmo es la utilización de operadores de selección, replicación y 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, a la búsqueda de la estructura de las funciones de base sigmoide y de base potencial que mejor se ajuste a la nube de puntos del conjunto de entrenamiento, determinando los valores de m_1 y m_2 asociados al número de funciones de base consideradas. Por otra parte, la evolución de los pesos de la red se corresponde con la evolución de los vectores $\mathbf{u_j}$ y $\mathbf{w_k}$ que determinan los coeficientes presentes en cada función de base, y de los coeficientes α_j, β_k, que son los coeficientes de la combinación lineal de las funciones de base. El algoritmo se estructura en los siguientes pasos:

1.- Generar la población
2.- Repetir hasta que se cumpla la condición de parada
- Calcular la aptitud para cada individuo
- Ordenar de mayor a menor según la aptitud
- Copiar el mejor en la siguiente generación
- Replicar el r % mejor de la población por el r % peor
- Aplicar mutación paramétrica al r % mejor
- Aplicar mutación estructural al (100-r)% restante

Los porcentajes se han determinado mediante prueba y error, aunque en la actualidad se están haciendo diseño de experimentos del tipo ANOVA para determinar la robustez y los valores óptimos de estos parámetros.

El algoritmo se inicia generando aleatoriamente N_R redes. Se comienza eligiendo el número total de nodos ocultos a partir de una distribución uniforme en el intervalo $(0, M]$, donde $m_1 + m_2 = M$ 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 $(0, p]$, siendo p 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 $[-U, U]$ 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 permanezca constante durante la evolución. Las mutaciones realizadas en el algoritmo son de dos tipos: *paramétricas y estructurales*. 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 $T(R)$ del modelo de red, y está definida mediante:

$$T(R) = 1 - A(R), \quad 0 \le T(R) \le 1 \tag{10}$$

donde la aptitud $A(R)$ de una red R se calcula como una función decreciente de la raíz del error cuadrático medio $E(R)$ a partir de la expresión

$$A(R) = \frac{1}{1 + E(R)}, \quad 0 < A(R) \le 1 \tag{11}$$

Las mutaciones paramétricas consisten en añadir a cada uno de los coeficientes w_{ki} una variable aleatoria de media cero y de desviación típica $\delta_1(t)T(R)$, mientras que a los coeficientes α_j, u_{ji} y β_k se le añade una variable aleatoria de media cero y de desviación típica $\delta_2(t)T(R)$, donde $\delta_1(t) \geq \delta_2(t), \quad \forall t$. Las mutaciones de los pesos w_{ki} correspondientes a los exponentes de las variables de la función que queremos modelar, han de ser menores que las mutaciones realizadas en los pesos α_j, β_k y u_{ji}, correspondientes a los coeficientes de las funciones de base y a la parte sigmoide del modelo, 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 para determinar si se acepta o no el cambio.

Los parámetros δ_1 y δ_2 que con la temperatura determinan las varianzas de la distribuciones normales, van cambiando durante el proceso de evolución para realizar un aprendizaje adaptativo; de esta forma evitamos quedar atrapados en mínimos locales y aceleramos el proceso de evolución cuando las condiciones del proceso lo permitan.

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 las capas de entrada y salida, afectando de esta forma a la topología de la red. Hemos utilizado 5 tipos de mutaciones, cuatro de ellas descritas con detalle en [13]: añadir nodos (AN), eliminar nodos (EN), añadir conexiones (AC) y eliminar conexiones (EC), a las que se añade 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.

5. Experimentación

Para poder realizar comparaciones entre los modelos de redes neuronales con diferentes funciones de base hemos considerado cuatro funciones tipo. Las tres primeras, f_1, f_2 y f_3, (ver Tabla 1), han sido propuestas en [7] y han sido ampliamente utilizadas en la bibliografía de modelado [2, 3, 4]. La cuarta p_1 la hemos propuesto para ampliar la tipología de funciones a estimar.

El propósito fundamental de las cuatro simulaciones funcionales es probar la eficacia y robustez de los modelos de redes implementados y analizar para qué tipo de función de prueba es preferible un tipo de red neuronal a utilizar como estimador.

El algoritmo evolutivo fue ejecutado 30 veces bajo los mismos parámetros (ver Tabla 3) para las cuatro funciones propuestas, a excepción del número de generaciones 3000, 1000, 1200 y 800 y el diseño inicial de los modelos (5:6:1), (4:3:1), (4:4:1) y (4:4:1), todos ellos con sesgo, para f_1, f_2, f_3 y p_1 respectivamente.

El conjunto de entrenamiento $D = \{(\mathbf{x_l}, y_l)\}$ está formado por 250 puntos obtenidos generando las variables independientes $\mathbf{x}$ siguiendo distribuciones uniformes de parámetros definidos en la Tabla 2, y una variable dependiente, y;

mientras que el conjunto de generalización está formado por 750 puntos sin ruido. Las variables independientes son escaladas linealmente en el intervalo $[0,1,\ 0,9]$ para todos los modelos y las dependientes en $[1,2]$ en el caso de PU y $[0,1,\ 0,9]$ en el caso de unidades sigmoides y en el modelo mixto.

$$f_1(\mathbf{x}) = 10sen(x_1x_2) + 20(x_3 - 0,5)^2 + 10x_4 + 5x_5 + \varepsilon_1, \quad x_i \in U(0,1)$$

$$f_2(\mathbf{x}) = (x_1^2 + (x_2x_3 - (\frac{1}{x_2x_4}))^2)^{1/2} + \varepsilon_2$$

$x_1 \in U(0,100)$, $x_i \in U(40\pi, 260\pi)$,

$x_3 \in U(0,1)$, $x_4 \in U(1,11)$

$$f_3(\mathbf{x}) = \tan^{-1}\left(\frac{x_2x_3 - (1/x_2x_4)}{x_1}\right) + \varepsilon_3$$

$x_i \in U(0,1)$

$$p_1(\mathbf{x}) = 2x_1^2\sqrt{x_2^3}\,e^{-x_3-x_4} - 2x_3^2\sqrt{x_4^3}\,e^{-x_1-x_2} + sen(x_1x_2x_3x_4) + \varepsilon_4,\ x_i \in U(0,1)$$

Tabla 1 Funciones de prueba utilizadas

Este diseño lo hemos realizado para poder comparar con diseños similares donde el tamaño del conjunto de entrenamiento es la cuarta parte del conjunto de generalización en datos simulados. Las variables aleatorias de error ε_k se generan siguiendo distribuciones normales $N(0,\sigma_k^2)$, donde la desviación típica $\sigma_1 = 1$ y las restantes $\sigma_2, \sigma_3, \sigma_4$ se obtienen de forma tal que se verifique la relación 3:1 entre la señal y el ruido para las funciones f_2, f_3 y p_1.

Población		Mut. Estruc: $[\Delta_m, \Delta_M]$		Mut. Para.	
Tamaño, N_R	1000	AN	[1, 2]	$\delta_1(0)$	1
Nº máx de FB,	8	EN	[1, 3]	$\delta_2(0)$	5
[-*L*, *L*]	[-5, 5]	AC	[1, 2*p*]	*r*	10
[-*U*, *U*]	[0, 5]	EC	[1, 2*p*]		

Tabla 2. Valores de los parámetros del algoritmo evolutivo

6. Resultados

Para poder realizar contrastes de diferencias de medias de los valores de la función de error MSE obtenidos por la mejor solución en cada una de las 30 ejecuciones del algoritmo; para cada modelo de red con unidades de base diferentes, y para cada función de prueba; hemos considerado un análisis de la varianza, ANOVA I, donde el único factor considerado es el tipo de unidades de base utilizado con los siguientes niveles: tipo unidad producto, PU, tipo mezcla de unidad producto-sigmoide, PUSU y tipo de unidad sigmoide, SU. Teniendo en cuenta los resultados se observa: Para la función f_1 de Friedman (Tabla 3), teniendo en cuenta el error MSE en el conjunto de generalización, MSE_G, es preferible, en media, el modelo PUSU (media MSE_G= 0,998), pero al no existir diferencias significativas en las medias entre los modelos PU y PUSU y habiéndolas en cuanto al número medio de coeficientes de estos dos modelos decidimos que, en media, es mejor utilizar una tipología de red basada en unidades producto, aunque los mejores modelos se obtienen con una red híbrida PUSU, donde el mejor modelo tiene un valor de MSE_G= 0,259 y de NRMS= 0,5090/4,8434= 0,1051, lo que muestra un muy buen ajuste, dado que un valor de NRMS= 1 indicaría un ajuste realizado con la media de los valores del conjunto de generalización.

Para la función f_2, la eficiencia medida como menor valor de MSE_G indica que en media, es preferible de nuevo modelos PUSU (media MSE_G= 2467,91), existiendo además diferencias significativas con las medias obtenidas con los otros dos modelos, por lo que en este caso los modelos PUSU son preferibles. El mejor modelo lo encontramos con esta tipología con un valor de MSE_G= 1049,27 y de NRMS= 32,39/ 364,552= 0,089

Respecto de la función f_3, el menor valor medio del error MSEG es 0,0340 obtenido con modelos de redes PUSU, pero de nuevo según el análisis ANOVA I realizado no existen diferencias significativas en las medias de los valores de MSEG, pero en cambio si existen diferencias significativas en los valores medios del número de parámetros del modelo PU respecto de los otros dos, por lo que es preferible este tipo de modelos de unidades de base, aunque el mejor modelo se obtiene tanto con unidades PU como PUSU con

valores de error MSE_G= 0,252 y de NRMS= 0,529.

Por último en la función de prueba propuesta, p_1, el valor medio menor se obtiene con unidades sigmoides, SU, con un valor medio de MSE_G= 0,0699, y como según el análisis ANOVA I realizado, existen diferencias en los valores medios de MSE_G entre este tipo de modelos y los modelos de unidades producto, mientras que no existen con los modelos PUSU. Además si analizamos las diferencias significativas existentes en cuanto a número de conexiones entre los modelos PUSU y SU (Sig= 0,002), de esta forma los modelos PUSU tienen menor número de coeficientes; pero el mejor modelo se obtiene con los modelos SU con valores de error MSE_G= 0,0284 y de NRMS= 0,61.

Fun.	F.Base	Media	SD	Mejor	Peor	Media	SD	Mejor	Peor	Media	SD
f_1	PU	1,560	0,187	1,228	1,870	1,003	0,304	0,399	1,517	*30,83*	*1,76*
	PUSU	1,733	0,774	1,118	3,934	*0,998*	0,775	*0,259*	3,205	33,30	2,10
	SU	2,476	0,607	1,461	3,537	1,634	0,638	0,432	2,615	34,23	3,28
f_2	PU	36097	299	35473	36773	3098	893	2121	6557	*14,40*	*0,89*
	PUSU	37340	1345	35860	42332	*2467*	1220	*1049*	5850	14,83	1,26
	SU	41270	2209	38367	46833	5126	2582	1847	11284	14,57	1,76
f_3	PU	0,2138	0,0032	0,2090	0,2212	0,0367	0,0064	*0,0252*	0,0521	*17,93*	*1,23*
	PUSU	0,2180	0,0038	0,2072	0,2232	*0,0340*	0,0042	0,0275	0,0432	20,03	1,54
	SU	0,2205	0,0026	0,2161	0,2293	0,0342	0,0042	*0,0252*	0,0416	20,87	2,06
p_1	PU	0,8519	0,0076	0,8380	0,8637	0,0964	0,0407	0,0580	0,2402	*19,10*	*1,03*
	PUSU	0,8619	0,0071	0,8506	0,8772	0,0791	0,0251	0,0403	0,1220	19,70	1,29
	SU	0,8709	0,0069	0,8587	0,8862	*0,0699*	0,0248	*0,0284*	0,1127	20,97	1,73

Tabla 3 Resultados estadísticos de los valores de error MSE de entrenamiento y generalización y nº de conexiones obtenidos por los mejores individuos en 30 ejecuciones del algoritmo evolutivo.

En la Tabla 4 pueden verse los resultados comparados del algoritmo propuesto con diferentes técnicas de regresión avanzadas que aparecen en la bibliografía sobre las funciones de Friedman #1,#2 y #3. Concretamente, las técnicas con las que se compara son: Bagging, Adaboost (Drucker, 1997) y máquinas de soporte vectorial (SVM), (Vapnik, 1999). El diseño experimental que hemos considerado ha sido en mismo que el realizado en (Drucker 1997) para obtener los resultados de la Tabla. 4. Los resultados de estas tres metodologías corresponden a la media sobre 10 ejecuciones. De los resultados de la Tabla 4 se desprende que el algoritmo evolutivo propuesto presenta mejores valores medios que el resto técnicas para las funciones de Friedman #1 y #2, mientras que en la función de Friedman #3 las técnicas de Boosting y SVM obtienen mejores resultados que el algoritmo evolutivo.

	Bagging	Boosting	SVM	AE
f1	2,2	1,65	0.67	0,998
f2	11463	11684	5402	2467
f3	0,0312	0,0218	0,026	0,034

Tabla 4. Resultados medios del error MSE de generalización

7. Conclusiones y trabajos futuros

El modelo híbrido propuesto formado por unidades de tipo sigmoide y unidades de tipo producto en una red neuronal puede presentarse como una alternativa viable a las redes neuronales con unidades de un único tipo. Las redes híbridas propuestas se han diseñado con un algoritmo evolutivo construido específicamente para este modelo. La evaluación del modelo y del algoritmo sobre tres funciones de prueba muestran unos resultados comparables a los de otras técnicas de regresión no lineal consolidadas en la teoría de regresión [7],[16],[17]. En las tres funciones de Friedman, los mejores resultados en media han sido obtenidos por los modelos hibridos, aunque el mejor modelo ha correspondido en el caso de la función de Friedman 2 a redes con un único tipo de unidad, producto y sigmoide.

Se puede afirmar en general que la hibridación de modelos constituye un procedimiento interesante para la resolución de problemas de regresión en los que la tipología especial de las funciones objetivo pueda presentar diferentes comportamientos en diversas regiones del dominio de definición. Quedan abiertos a futuras investigaciones diversos aspectos:

A partir de los datos generados por una determinada función continua y desconociendo la función que los generó, ¿es posible definir las características que determinen la familia de funciones (reconocedores universales) que es conveniente considerar para aproximarla? ¿ podemos determinar algunas propiedades de la función que permitan elegir entre diferentes familias de reconocedores universales, o bien entre la mezcla "inteligente" de diferentes modelos?

-¿Es posible incorporar al algoritmo evolutivo procedimientos que permitan en determinados momentos de la evolución determinar el tipo de unidad que hay que incluir en el modelo para mejorar el grado de aproximación y reducir el error de entrenamiento y generalización?

-Por último, parece lógico incorporar al algoritmo evolutivo procedimientos de búsqueda local, basados o no en el gradiente, que permitan mejorar la eficiencia y eficacia del algoritmo evolutivo, controlando el riesgo de sobreentrenamiento que puede traer consigo la hibridación en el algoritmo.

Referencias

[1] Specht, D.F., A General Regression Neural Network. IEEE Transactions on Neural Networks,. 2 (6): p. 568-576, 1991.

[2] Durbin, R., & Rumelhart, D., Products Units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation, 1: p. 133-142, 1989.

[3] Duch, W. and Jankowsky, N., Transfer functions: hidden possibilities for better neural networks, 9th European Symposium on Artificial Neural Networks, (ESANN), Brugge, pp. 81-94, 2001.

[4] Cohen, S. and Intrator, N., Forward and backward selection in regression hybrid network, Third International Workshop on Multiplier Classifier Systems, 2002.

[5] Cohen and Intrator, A Hybrid Projection-based and Radial Basis Function Architecture: Initial Values and Global Optimisation, Pattern Analysis & Applications, 113-120, 2005.

[6] Donoho D. Projection based in approximation and a duality with kernel methods. Ann. Statist., 17, 58-106, 1989.

[7] Friedman J. "Multivariate adaptive regression splines (with discussion)". Ann. Stat. vol 19, pp 1-141, 1991.

[8] Vapnick, N., The nature of Statistical Learning Theory, Springer, 1999.

[9] Schmitt, M., On the Complexity of Computing and Learning with Multiplicative Neural Networks. Neural Computation, 14: p. 241-301, 2001.

[10] Cybenko, G., Aproximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2: p. 302-366, 1989.

[11] Leshno, M., Lin, V. L., Pinkus, A., and Schocken, S., Multilayer feedforward networks with a nonpolynomical activation can approximate any function. Neural Networks, 6: 861-867, 1993.

[12] Fogel, D.B. 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.

[13] Angeline, P.J., Saunders, G. M., & Pollack, J. B., An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5 (1): p. 54-65, 1994.

[14] García-Pedrajas, N., Hervás-Martínez, C.& Muñoz-Pérez, J., Multiobjetive cooperative coevolution of artificial neural networks. Neural Networks, 15(10): p. 1255-1274, 2002.

[15] Drucker H., Improving Regressors using Boosting Techniques, Proceedings of the Fourteenth International Conference in Machine Learning, .Ed. Douglas, H. and Fisher, J., pp. 107-115, 1997.

[16] Breiman L. Bagging predictors. Machine learning, 24, 123-140, 1996

[17] Breiman L., Friedman J. H., Olshen R. A. and Stone C. J. Classification and regression trees. The Wadsworth Statistics/Probabilities series, Belmont, C. A, 1984.