Clasificación mediante la evolución de Modelos Híbridos de Redes Neuronales

Cesar Hervás[1], Francisco J. Martínez[2], Pedro A. Gutiérrez[1], Juan C. Fernández[1], A. Tallón[1]

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 clasificación 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 cinco bases de datos extraídas del repositorio de la UCI, muestran resultados muy prometedores en esta dirección.**

Palabras clave—**Redes neuronales, funciones de base, algoritmos evolutivos.**

I. INTRODUCCIÓN

La resolución de problemas de clasificación 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 convierten 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 en el diseño de redes neuronales donde se utilizan diferentes tipos de unidades de base en la capa oculta.

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

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 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, 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 nuestro conocimiento, 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 clasificación. 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 del número de patrones mal clasificados al predecir la clase de pertenencia de los patrones de generalización, a partir de los valores numéricos obtenidos para las variables observadas en el conjunto de entrenamiento. Con el objetivo de evaluar el rendimiento (en cuanto a eficacia y simplicidad) de los modelos de clasificación propuestos los hemos evaluado sobre cinco bases de datos obtenidas de la UCI [18]: *Pima Indians Diabetes*, *Balance Scale*, *Card Credit Australian Heart Disease y Glass*, para determinar la estructura que mejor se ajusta a la tipología (número y tipo de funciones de base) del problema de clasificación propuesto. Los resultados obtenidos muestran un rendimiento comparable al de otras técnicas de clasificación consolidadas.

II. ALGUNAS 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,

$$B_k(\mathbf{x},\mathbf{w}_k)=\prod_{i=1}^{p} x_i^{w_{ki}}\ ,\ \mathrm{k}=1,\ldots,\mathrm{m}_2\ ; \qquad (1)$$

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 $\mathrm{j}=1,\ldots,\mathrm{m}_1$ (2)

Nos planteamos a continuación las características que debe de tener un problema de clasificación para que se ajuste mejor con unidades de tipo producto versus unidades de tipo sigmoide. 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 de entropía cruzada subyacente al problema de clasificación sea no acotada. Si consideramos como dominio de definición todo el espacio $\mathbb{R}^n$, las funciones MLP y RBF están acotadas superior e inferiormente. Sin embargo una función basada en unidades PU no está acotada:

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

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

para determinadas elecciones de los parámetros del modelo.

Sin embargo, para las unidades sigmoides y las RBF

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

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

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.

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

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

Normalmente, los datos del conjunto de entrenamiento que se tienen como información para la resolución de un problema de clasificación pertenecen a un conjunto compacto, que es donde se aprenden los parámetros del modelo de clasificación. 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 del conjunto de entrenamiento. 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.

III. MÉTODOS ADAPTATIVOS PARA LA ESTIMACIÓN DE CLASES DE PERTENENCIA

En este trabajo mostramos las primeras investigaciones en la estimación de la clase de pertenencia de un conjunto de patrones mediante combinaciones lineales de funciones de base sigmoide y potencial. Para ello se establece un número de neuronas en la capa de salida igual al número de clases del problema a tratar, partiendo de un conjunto de medidas x_i, $i=1,2,...,k$,que son tomadas en un único patrón que debe ser clasificado en una de las l clases, basándonos en dichas medidas. Los patrones de entrenamiento tienen la forma $D=\{(\mathbf{x}_n,\mathbf{y}_n);n=1,2,...,N\}$, donde $\mathbf{x}_n=(x_{1n},...,x_{pn})$ es el vector con las variables de entrada e $\mathbf{y}_n$ es la clase del patrón n-ésimo.

Adoptaremos la técnica común de utilizar un vector "1-de-1" para representar la clase del patrón. Estos vectores tienen la forma $\mathbf{y}=(y^{(1)},y^{(2)},...,y^{(J)})$, siendo $y^{(j)}$ igual a 1 si el ejemplo pertenece a la clase j, e igual a cero en caso contrario. Basándonos en este conjunto de entrenamiento, el objetivo es encontrar una red neuronal R que clasifique bien a los patrones del conjunto de generalización. Una clasificación incorrecta se produce si la red R asigna a un patrón la clase j cuando en realidad pertenece a la clase $i \neq j$.

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

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

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 $B_0(\mathbf{x},\mathbf{w_0})=1$ 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 de clasificación 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:

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

El método consiste en encontrar un número suficiente de los diferentes tipos de funciones de base (su arquitectura) para cada clase existente en el problema (salidas de la red neuronal), de forma que nos proporcione un grado de pertenencia a una determinada clase, y de esta forma podamos 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 se minimice una función asociada a los errores de clasificación sobre el conjunto de entrenamiento.

Para resolver este problema 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.

IV. 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 optimicen la clasificación sobre un conjunto de generalización a partir del modelo de clasificación estimado sobre el conjunto de entrenamiento. Para ello es necesario, en primer lugar determinar los valores de m_1 y m_2 asociados al número de funciones de base consideradas; considerando a la vez la evolución de los pesos de la red, la cual 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 y para la población inicial se cumple que $m_1 = m_2$. 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).

Las mutaciones paramétricas utilizan la regla de razón de éxito 1/5 de Rechenberg [15] y consisten en añadir a cada uno de los coeficientes α_j, β_k, una variable aleatoria de media cero y de desviación típica $\alpha_1(t)T(R)$, mientras que a los coeficientes u_{ji} y w_{ki} se le añade una variable aleatoria de media cero y de desviación típica $\alpha_2(t)T(R)$, donde $\alpha_1(t) \ll \alpha_2(t), \ \forall t$. Las mutaciones de los pesos u_{ji} y 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, correspondientes a los coeficientes de las funciones de base, 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 α_1 y α_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 que están descritas con detalle en [5]: 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.

V. FUNCIONES DE APTITUD

La severidad de las mutaciones depende de la temperatura $T(R)$ del modelo de red, y está definida mediante:

$$T(R) = 1 - A(R), \quad 0 \le T(R) \le 1$$

donde en una primera aproximación la aptitud $CCR(R)$ de una red R se calcula como el ratio de patrones correctamente clasificados $CCR(R)$ a partir de la expresión

$$CCR(R) = \frac{\sum_{n=1}^{N} I(R(x_n) = y_n)}{N}, \quad 0 < CCR(R) \le 1,$$

siendo $I(g)$ una función que devuelve un 1 si g es verdadero y un 0 en caso contrario, y $R(x_n)$ la clase en la que la red neuronal clasifica el patrón n. Un buen clasificador maximizaría el ratio $CCR(R)$ o minimizaría el error correspondiente, es decir, $1 - CCR(R)$.

Pero en este trabajo adoptamos un enfoque diferente, ya planteado en trabajos anteriores de clasificación [16][17] para modelos sólo con unidades producto con buenos resultados, al interpretar las salidas de las neuronas de la capa de salida con un enfoque probabilístico que considera

la función de activación *soft*max en cada una de las neuronas de la capa de salida, la cual viene dada por:

$$g_j\left(\mathbf{x},\boldsymbol{\theta}_j\right)=\frac{\exp f_j\left(\mathbf{x},\boldsymbol{\theta}_j\right)}{\sum_{l=1}^{J}\exp f_l\left(\mathbf{x},\boldsymbol{\theta}_l\right)},\ l=1,2,...,J$$

Siendo J el número de clases del problema, $f_j\left(\mathbf{x},\boldsymbol{\theta}_j\right)$ la salida de la neurona j para el patrón **x** y $g_j\left(\mathbf{x},\boldsymbol{\theta}_j\right)$ la probabilidad de que el patrón **x** pertenezca a la clase j. La transformación *soft*max produce estimaciones de las probabilidades de pertenencia de un patrón del conjunto de entrenamiento a cada una de las clases. Teniendo en cuenta esta consideración, se puede comprobar que la clase predicha por la red neuronal es la correspondiente a la neurona de la capa de salida cuyo valor de salida es mayor. De esta forma la regla optima $C(\mathbf{x})$ es la siguiente:

$$C(\mathbf{x})=\hat{l},\ \textit{donde}\ \hat{l}=\arg\max_l g_l(\mathbf{x},\hat{\boldsymbol{\theta}}),\ l=1,2,...,J$$

Al tratarse de probabilidades de pertenencia a una clase está claro que no es necesario calcularlas todas, ya que la última probabilidad $g_J\left(\mathbf{x},\boldsymbol{\theta}_J\right)$ se puede calcular en función del resto como $1-\sum_{j=1}^{J-1}g_j(\mathbf{x},\boldsymbol{\theta}_j)$. De esta forma, podemos simplificar el modelo considerando la salida de una de las neuronas de la capa de salida constante e igual a 0, es decir, $f_J\left(\mathbf{x},\boldsymbol{\theta}_J\right)$=0. Esta neurona no será entrenada, para así reducir el número de parámetros a estimar y la carga computacional del entrenamiento.

La medida que utilizamos en este trabajo para evaluar la aptitud de un modelo de clasificación es la función de error *cross-entropy* (entropía cruzada), para cada observación y viene dada por la siguiente expresión para J clases:

$$l(\boldsymbol{\theta})=-\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{J}y_n^{(l)}\log g_l\left(\mathbf{x}_n,\boldsymbol{\theta}_l\right)=$$
$$=\frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J}y_n^{(l)}f_l(\mathbf{x}_n,\boldsymbol{\theta}_l)+\log\sum_{l=1}^{J}\exp f_l(\mathbf{x}_n,\boldsymbol{\theta}_l)\right]$$

donde $\boldsymbol{\theta}=(\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_J)$. La superficie de error asociada al tipo de modelos de clasificación propuesto es muy irregular, con numerosos óptimos locales, siendo indefinida, en general, la matriz Hessiana de la función $l(\boldsymbol{\theta})$, es por ello por lo que utilizamos una heurística como es la computación evolutiva.

La ventaja del uso de la función de error $l(\boldsymbol{\theta})$ en lugar de $(1-CCR)$ es que es una función continua, lo que permite que el entrenamiento pueda converger siempre hacia soluciones más óptimas, y que el algoritmo evolutivo converja más pausadamente.

VI. EXPERIMENTACIÓN

Para poder realizar comparaciones entre los modelos de redes neuronales con diferentes funciones de base hemos considerado cinco bases de datos obtenidas de la UCI [18] (ver TABLA I), *Pima Indians Diabetes, Balance Scale, Card Credit Australian, Heart Disease y Glass.*

TABLA I
BASES DE DATOS UTILIZADAS PARA CLASIFICACIÓN

	Pima	**Balance**	**Card**	**Heart**	**Glass**
#Patrones total	768	625	690	270	214
#Patrones training	576	469	518	202	161
#Patrones test	192	156	172	68	53
#Variables entrada	8	4	47	13	9
#Clases	2	3	2	2	6

El propósito fundamental de la clasificación de estos cinco problemas es probar la eficacia y robustez de los modelos de redes implementados y analizar para qué tipo de base de datos de prueba es preferible un tipo de red neuronal (tipos de funciones de base) a utilizar como clasificador.

El proceso que se ha seguido para realizar el entrenamiento de los modelos ha sido un *hold-out* de validación cruzada, donde el tamaño del conjunto de entrenamiento es aproximadamente 3*n*/4 y el de generalización *n*/4, siendo *n* el número total de patrones.

El algoritmo evolutivo fue ejecutado 10 veces para cada experimento bajo los mismos parámetros (ver TABLA II) para las cinco bases de datos propuestas, a excepción del número de generaciones y número máximo de neuronas en capa oculta, dependiendo de las unidades de base utilizadas para el experimento (ver Tabla III)

TABLA II
VALORES COMUNES DE LOS PARÁMETROS DEL ALGORITMO EVOLUTIVO

POBLACIÓN		**MUT. ESTRUCTURAL**		**MUT. PARAMÉTRICA**	
Tamaño, N_R	1000	AN	[1, 2]	$\alpha_1(0)$	0.5
		EN	[1, 2]	$\alpha_2(0)$	1
[-L, L]	[-5, 5]	AC	30%(H),5%(O)	r	10
[-U, U]	[-5, 5]	EC	30%(H),5%(O)		

El algoritmo evolutivo se detendrá cuando transcurran 20 generaciones sin mejorar la media de aptitud del 10% de la población y 20 generaciones si

mejorar la aptitud del mejor individuo de la población.

El número de nodos a añadir en la mutación estructural se ha limitado en el intervalo $[1,2]$, al igual que el número de nodos a eliminar. Se ha utilizado la variante relativa para las mutaciones estructurales de añadir y eliminar enlaces (AE y EE). Los porcentajes de enlaces mutados han sido fijados en un 30% para la capa oculta y un 5% para la capa de salida.

Las variables independientes son escaladas linealmente en el intervalo $[0.1, 0.9]$ para todos los modelos.

TABLA III

CONFIGURACIONES TOPOLÓGICAS MEJORES MODELOS EN CADA TIPOLOGÍA

BASE DATOS	Nº GENE	CONFIG. (I:H:O)	UNID. BASE
Pima	120	(8:4:2)	Sigmoide
		(8:2:2)	UP
		(8:4:2)	Sig-UP
Balance	150	(4:6:3)	Sigmoide
		(4:5:3)	UP
		(4:4:3)	Sig-UP
Card	250	(47:2:2)	Sigmoide
		(47:2:2)	UP
		(47:2:2)	Sig-UP
Heart	250	(13:2:2)	Sigmoide
		(13:4:2)	UP
		(13:2:2)	Sig-UP
Glass	1000	(9:12:6)	Sigmoide
		(9:11:6)	UP
		(9:10:6)	Sig-UP

VII. RESULTADOS

Para poder realizar contrastes de diferencias de medias de los valores de la función de error CCR obtenidos por la mejor solución en cada una de las 10 ejecuciones del algoritmo; para cada modelo de red con unidades de base diferentes, y para cada base de datos; 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.

Los estadísticos asociados al CCR para el conjunto de generalización se presentan en la tabla IV donde se observa que los modelos híbridos no son mejores en media que los simples, pero los mejores modelos obtenidos indican que la hibridación obtiene modelos tan buenos como los mejores obtenidos con las unidades simples y además con un número de parámetros no significativamente superior, en la tabla III observamos que el número de nodos en capa oculta es en general inferior o igual a los propuestos para los modelos simples. Estos mejores modelos se muestran en su formulación completa en la tabla V donde podemos observar que el número de parámetros no es muy superior al necesario si utilizamos funciones discriminantes lineales y en alguna base de datos como Card inferior.

VIII. 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 en cuanto a la obtención de los mejores modelos. Las redes híbridas propuestas se han diseñado con un algoritmo evolutivo construido específicamente para este modelo. La evaluación del modelo y del algoritmo sobre las cinco bases de datos utilizadas muestran unos resultados comparables a los de otras técnicas de clasificación consolidadas en la literatura de machine learning [19]

Se puede afirmar, en principio, que la hibridación de modelos constituye un procedimiento interesante para la resolución de problemas de clasificación 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 las distribuciones de frecuencias asociadas a los patrones de los conjuntos de entrenamiento en problemas de clasificación y de las funciones de entropía cruzada ¿Es posible definir las características que determinen la familia de funciones (reconocedores universales) que es conveniente considerar para clasificarlos con mayor precisión? ¿Podemos determinar algunas propiedades de las funciones de probabilidad a priori 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 la precisión del clasificador en el conjunto de 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.

Introducir en el Algoritmo Evolutivo una función de aptitud con multicriterio para penalizar aquellas redes que poseen una mayor complejidad estructural y de este modo obtener modelos más sencillos e interpretables.

Mejorar la eficiencia del Algoritmo Evolutivo para lograr una mayor rapidez en su ejecución. Un enfoque interesante sería paralelizar el Algoritmo Evolutivo repartiendo la población entre varias máquinas y realizar así el entrenamiento en paralelo.

AGRADECIMIENTOS

Este trabajo ha sido financiado en parte por el proyecto TIN2005-08386-C05-02 de la Comisión Interministerial de Ciencia y Tecnología y fondos FEDER.

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:hidenn posibilities 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, attern 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] I. Rechenberg, Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution. Stuttgart: Framman-Holzboog 1973.

[16] Hervás, C. Martinez, F.J. y Gutiérrez, P.A. Classification by means of Evolutionary Product-Unit Neural Networks. IEEE World Congress of Computational Intelligence. International Joint Conference on Neural Networks. 2006. Vancouver, Canada.

[17] Martínez, F.J. Hervás, C. Gutiérrez, P.A. Martínez A, Ventura S. Evolutionary Product-Unit Neural Networks for Classification. Intelligent Data Engineering and Automated Learning. 2006. Burgos, Spain.

[18] C. Blake and C. J. Merz, " UCI repository of machine learning data bases," www.ics.uci.edu/ mlearn/MLRepository.thml, 1998.

[19] N. Landwehr, M. Hall, and F. Eibe, "Logistic Model Trees," Machine Learning, vol. 59, pp. 161-205, 2005.

TABLA IV.

RESULTADOS ESTADÍSTICOS DE LOS VALORES DE ERROR CCR DE ENTRENAMIENTO Y GENERALIZACIÓN Y Nº DE CONEXIONES OBTENIDOS POR LOS MEJORES INDIVIDUOS EN 10 EJECUCIONES DEL ALGORITMO EVOLUTIVO

		ENTRENAMIENTO				GENERALIZACIÓN				Nº CON.	
BASE DATOS	**F.B**	**MEDIA**	**SD**	**MEJOR**	**PEOR**	**MEDIA**	**SD**	**MEJOR**	**PEOR**	**MEDIA**	**SD**
PIMA	UP	76,684	0,441	77,256	75,868	77,968	1,204	79,687	76,562	8,600	1,349
	SIG	76,961	0,907	78,819	76,215	80,052	1,534	**82,291 (1)**	78,125	23,300	3,622
	SIG+UP	77,621	0,539	78,645	76,909	79,062	1,587	81,770	77,604	22,500	4,275
BALANCE	UP	97,974	0,230	98,294	97,654	97,564	0,662	98,076	96,794	17,800	3,881
	SIG	91,791	1,069	93,176	89,765	92,051	0,865	92,948	90,384	35,100	3,446
	SIG+UP	97,313	2,163	98,933	91,257	97,371	1,824	**98,076 (2)**	92,307	21,900	2,806
CARD	UP	85,386	0,681	86,293	84,362	88,255	0,534	88,953	87,790	16,900	5,586
	SIG	87,644	1,084	89,961	86,486	87,732	1,699	**90,116 (3)**	84,883	25,300	6,037
	SIG+UP	88,166	0,971	89,189	85,907	87,267	0,796	88,372	86,046	30,100	7,015
HEART	**UP**	88,762	2,152	91,089	84,158	85,000	3,084	**89,705 (4)**	80,882	23,200	4,984
	SIG	87,524	1,162	89,603	85,643	85,441	2,724	88,235	80,882	18,800	2,740
	SIG+UP	87,376	1,480	90,099	85,643	84,852	1,705	86,764	80,882	17,700	1,828
GLASS	UP	79,814	2,196	82,609	76,398	66,415	3,753	73,585	62,264	84,800	7,857
	SIG	80,932	1.436	83,230	78,882	68,113	1,392	69,811	66,038	114,300	6,464
	SIG+UP	79,130	1,470	81,366	77,019	66,604	4,177	**73,585 (5)**	60,377	90,500	9,823

TABLA V.

MEJORES MODELOS OBTENIDOS PARA REDES CON DIFERENTES UNIDADES DE BASE

BASE DE DATOS	(FUNCIONES DISCRIMINANTES FD CON UNIDADES PRODUCTO P Y SIGMOIDES S)
PIMA (1) SIG Nº CON= 27	FD= 0,234 +5,180*(1/(1+e^-(-0,553-0,965*(x1)+0,155*(x4)+1,141*(x5)-4,184*(x8)))) -4,602*(1/(1+e^-(-5,456+5,087*(x2)-0,076*(x5)+2,933*(x6)+3,132*(x7)))) +0,912*(1/(1+e^-(-4,328+0,448*(x1)+1,935*(x2)+0,042*(x6)+4,404*(x8)))) +5,249*(1/(1+e^-(+2,795-2,249*(x1)-3,306*(x2)+1,705*(x3)+0,268*(x5)-4,791*(x6)+0,356*(x7))))
BALANCE (2) SIG+UP Nº CON=26	FD1= -0,687-1,415*S1+2,381*S2+3,352*S3-7,359*P1 FD2= 8,555 +5,305*S1+0,221*S2 +1,145*S3-2,040*P1 P1= (x1^-4,941*x2^-4,752*x3^4,733*x4^4,793) ; S1= 1/(1+e^-(+1,251-3,822*(x1)-2,846*(x2)+0,033*(x4))); S2= 1/(1+e^-(-0,611-4,690*(x3)-0,287*(x4))); S3= 1/(1+e^-(+0,950+1,314*(x1)+3,783*(x2)-4,991*(x3)-4,699*(x4)))
card (3) SIG Nº CON=30	FD= -3,496 +6,70*(1/(1+e^-(-4,637+0,408*(x3)+3,698*(x23)-1,389*(x30)-0,158*(x37)+4,196*(x39)+1,182*(x40)+1,514*(x47)))) +4,254*(1/(1+e^-(-2,128*(x3)-5,811*(x8)-2,868*(x10)+4,921*(x12)+4,264*(x15)-4,924*(x18)+1,771*(x19) -3,360*(x20)+0,600*(x23)+2,794*(x30)+2,437*(x32)+1,754*(x33)-2,942*(x35)-4,268*(x36)+2,182*(x37)+1,792*(x39) -0,387*(x42)+3,093*(x43)+2,172*(x44))))
Heart (4) UP Nº CON=20	FD= +4,057 -1,853*(x1^-1,607*x4^0,415*x6^0,008*x7^0,510*x13^3,392) -3,231*(x2^0,651*x8^-0,805*x12^0,531) -2,840*(x2^-1,688*x9^1,803*x10^1,218*x13^0,944) -9,283*(x3^3,110*x10^0,522*x12^0,363)
GLASS (5) SIG+UP Nº CON=78	FD1= 2,193+10,125*S1-9,678*S2+1,0875*S3-0,715* P1+1,461* P3-1,558* P4+3,335*P5 FD2= 0,511+1,706*S1+7,474*S4+10,470*S5-11,321*P1 +3,564*P2 +2,338*P3 -4,270*P4+0,535*P5 FD3= -3,337 +10,133*S1 +9,863*S3-0,033*S4-1,343*P1 FD4= 2,883-8,373*S1+7,814*S2-1,696* S3+1,171*S4 -8,691*P1 +2,992*P2 -4,223* P4+2,797*P5 FD5= -0,995+0,248*S1+6,712*S3-4,327*P1+5,494*P3-1,988*P4 P1= (x6^-4,080*x8^5,171); P2= (x3^5,490*x4^2,005*x7^-3,41); P3= (x2^4,215*x5^1,440*x6^-2,024*x7^0,673); P4=(x1^-5,650*x2^-3,137*x4^2,305*x5^8,121*x7^2,653*x8^1,630); P5=(x1^-5,294*x2^-0,578*x4^1,187*x5^1,563*x8^1,313*x9^3,114); S1= 1/(1+e^-(-6,278+1,524*(x2)+9,352*(x3)+0,702*(x6))); S2= 1/(1+e^-(+4,602-5,0390*(x1)-6,078*(x3))); S3= 1/(1+e^-(-1,718-3,117*(x1)+8,162*(x2)-3,051*(x5)+1,349*(x9))); S4= 1/(1+e^-(-0,511+3,837*(x1)-2,750*(x4)-3,472*(x5)+2,403*(x6)-0,072*(x8)+3,936*(x9))) S5= 1/(1+e^-(-0,248+1,012*(x2)-4,322*(x3)-5,045*(x4)-0,741*(x5)+3,961*(x7)))

L