REVISTA DE MÉTODOS CUANTITATIVOS PARA LA ECONOMÍA Y LA EMPRESA (3). Páginas 40–62.
Junio de 2007. ISSN: 1886-516X. D.L: SE-2927-06.
URL: http://www.upo.es/RevMetCuant/art10.pdf

Modelo no lineal basado en redes neuronales de unidades producto para clasificación. Una aplicación a la determinación del riesgo en tarjetas de crédito

Martínez-Estudillo, F. J.
Departamento de Gestión y Métodos Cuantitativos, ETEA Córdoba (España)
Correo electrónico: fjmestud@etea.com

Hervás-Martínez, C.
Departamento de Informática y Análisis Numérico, Universidad de Córdoba
Correo electrónico: chervas@uco.es

Torres-Jiménez, M.
Departamento de Gestión y Métodos Cuantitativos, ETEA Córdoba (España)
Correo electrónico: mtorres@etea.com

Martínez-Estudillo, A. C.
Departamento de Gestión y Métodos Cuantitativos, ETEA Córdoba (España)
Correo electrónico: acme@etea.com

RESUMEN

El principal objetivo de este trabajo es mostrar un tipo de redes neuronales denominadas *redes neuronales basadas en unidades producto* (RNUP) como un modelo no lineal que puede ser utilizado para la resolución de problemas de clasificación en aprendizaje. Proponemos un método evolutivo en el que simultáneamente se diseña la estructura de la red y se calculan los correspondientes pesos. La metodología que presentamos se basa, por tanto, en la combinación del modelo no lineal RNUP y del algoritmo evolutivo; se aplica a la resolución de un problema de clasificación de índole económica, surgido del mundo de las finanzas. Para evaluar el rendimiento de los modelos de clasificación obtenidos, comparamos nuestra propuesta con varias técnicas clásicas, como la regresión logística o el análisis discriminante, y con el clásico modelo de perceptrón multicapa de redes neuronales basado en unidades sigmoides y el algoritmo de aprendizaje de retropropagación (MLPBP).

Palabras clave: clasificación; redes neuronales de unidades producto; redes neuronales evolutivas.
Clasificación JEL: C45; C63.
2000MSC: 68T10; 62M45.

Artículo recibido el 11 de mayo de 2007 y aceptado el 5 de junio de 2007.

Non-linear model for classification based on product-unit neural networks. An application to determine credit card risk

ABSTRACT

The main aim of this work is to show a neural network model called *product unit neural network* (PUNN), which is a non-linear model to solve classification problems. We propose an evolutionary algorithm to simultaneously design the topology of the network and estimate its corresponding weights. The methodology proposed combines a non-linear model and an evolutionary algorithm and it is applied to solve a real economic problem that occurs in the financial management. To evaluate the performance of the classification models obtained, we compare our approach with several classic statistical techniques such us logistic regression and linear discriminat analysis, and with the multilayer perceptron neural network model based on sigmoidal units trained by means of Back-Propagation algorithm (MLPBP).

Keywords: classification; product unit neural networks; evolutionary neural networks.
JEL classification: C45; C63.
2000MSC: 68T10; 62M45.

I. INTRODUCCIÓN

Las redes neuronales artificiales han surgido en los últimos años, demostrando ser una herramienta eficiente para la resolución de problemas en el ámbito de la economía y las finanzas. Cuestiones como la predicción, la clasificación del riesgo y la selección de información dispersa en los mercados se plantean como elementos claves para una eficiente gestión financiera. Los datos que con frecuencia aparecen en los mercados financieros están afectados por ruido y suelen presentar un comportamiento no lineal. Son numerosos los trabajos que recientemente han abordado la resolución de problemas de predicción y de clasificación en el ámbito económico financiero con redes neuronales artificiales.

Probablemente, el mayor número de aplicaciones de las redes neuronales a la economía se hallan en el ámbito de la predicción de series temporales en mercados de capitales. Los modelos de redes neuronales, unidos en algunos casos a los algoritmos genéticos, han sido usados por ejemplo para la predicción del nivel o el signo de los retornos de índices bursátiles, tanto para índices más grandes y estables, véanse [1] y [2], como en mercados más volátiles, como los asiáticos en [3]. La predicción de la producción de automóviles o de la tasa de impago de bonos corporativos de alto riesgo y la evolución de los márgenes de retorno de estos bonos son otros ejemplos interesantes del uso de la herramienta para la predicción.

También en problemas de clasificación podemos encontrar numerosos ejemplos en los que las redes neuronales han mostrado su rendimiento, mejorando con frecuencia la precisión alcanzada por técnicas clásicas como el análisis discriminante o la regresión logística. La evaluación del riesgo de crédito [4], el análisis de las condiciones y señales que pueden hacer recomendable una intervención bancaria [5], la predicción de la bancarrota [6], aviso previo de insolvencia para las compañías de seguros [7], valoración de propiedades en el condado de San Diego o una clasificación de las empresas españolas en varias categorías [8] son algunos ejemplos claros en este sentido.

En el presente trabajo no nos marcamos como objetivo hacer una revisión exhaustiva de la aportación realizada por las redes neuronales a la economía. Un estudio más extenso y detallado de las numerosas aplicaciones de los modelos de redes neuronales a la resolución de problemas económico-financieros puede consultarse, por ejemplo, en [5,9]. Nuestro propósito es mostrar un tipo de redes neuronales denominadas redes neuronales basadas en unidades producto que, aunque fueron

introducidas por Durbin y Rumelhart en 1989 [1,10], no han sido muy utilizadas por los investigadores en el área de ciencias de la computación y, como consecuencia, tampoco en otras áreas como la economía y las finanzas en las que sin embargo sí han encontrado aplicación las redes neuronales en su formulación clásica determinada por el perceptrón multicapa (MLP).

Las redes unidades producto están basadas en nodos multiplicativos en vez de en nodos de tipo aditivo. La combinación lineal de las variables de entrada en las clásicas redes MLP es ahora sustituida por un producto de estas variables elevadas a exponentes reales. Desafortunadamente, como veremos en el siguiente epígrafe, la superficie de error asociada a las redes de unidades producto es muy rugosa, con numerosos óptimos locales y también con regiones llanas. Este hecho es el que motiva el uso de algoritmos evolutivos como herramienta para optimizar la función de error asociada al problema.

Aunque son numerosas las aplicaciones de las redes neuronales a la economía, nos centraremos en una de las aplicaciones en la resolución de problemas de clasificación. Concretamente, estudiaremos el rendimiento de las redes de unidades producto en un problema de clasificación del impago de tarjeta de crédito surgido del mundo de las finanzas: la base de datos "German card". Los resultados obtenidos por el modelo RNUP serán comparados con los que obtienen las redes MLP clásicas, basadas en unidades de tipo sigmoide, y con modelos clásicos como el análisis discriminante o la regresión logística.

El trabajo se estructura de la siguiente forma: la sección II revisará de forma no exhaustiva diferentes modelos lineales y no lineales que han sido usados en la resolución de problemas de clasificación; en la sección III se presenta el modelo no lineal basado en unidades producto, mostrándose la estructura funcional, la representación en red neuronal adaptada a un problema de clasificación y un análisis de las fortalezas y debilidades del modelo; la sección IV desarrolla el modelo evolutivo que proponemos para la optimización del modelo; la sección V expondrá los resultados del modelo en el problema de clasificación elegido y la comparación con métodos clásicos como el análisis discriminante y la regresión logística por una parte, y la comparación con las redes MLP entrenadas con retropropagación por otra.

II. MODELOS LINEALES Y NO LINEALES EN CLASIFICACIÓN

Es conocido que el método más simple para asignar un patrón a la clase correspondiente es

a través de una función lineal de las variables de entrada. Este proceso es bastante estable, es decir, la varianza del resultado obtenido con respecto a diferentes conjuntos de datos de entrenamiento es pequeña; aunque el error cometido es mayor que si consideramos modelos no lineales que ajustarán mejor los datos de entrenamiento aunque con una mayor varianza con respecto a la variación del conjunto de datos de entrenamiento[1]. Con frecuencia, en un problema con datos reales, no es posible realizar la suposición de linealidad entre las variables. La hipótesis de linealidad resulta bastante restrictiva como punto de partida. A continuación, iniciamos un recorrido breve por algunos métodos que van más allá de la linealidad, introduciendo la no linealidad en el modelo de diferentes formas y en diferente grado.

El primer método que consideramos es el clásico análisis discriminante [11], en el que se supone que las variables de entrada siguen distribuciones normales. Cuando las matrices de covarianzas de las distribuciones normales asociadas a cada clase son las mismas, la regla de decisión determinada por el método está basada en fronteras lineales en las variables independientes. Si esta hipótesis no se cumple, la regla de decisión da como resultado fronteras de tipo cuadrático. El análisis discriminante ha sido usado en numerosas aplicaciones, por lo que en la parte experimental será uno de los métodos frente a los que compararemos el rendimiento de nuestro modelo.

Otro método clásico usado en multitud de aplicaciones en clasificación binaria es la regresión logística. El método trata de estimar las probabilidades a posteriori de pertenencia de cada uno de los patrones de un conjunto de entrenamiento a uno de los dos valores que toma la variable dependiente mediante relaciones lineales entre las variables predictoras [12], bajo la hipótesis de que la función de probabilidad asociada a una de las clases es de tipo sigmoide. La estimación de los parámetros se hace por máxima verosimilitud. Para obtener el óptimo, el procedimiento habitual es un método iterativo de tipo Newton-Raphson denominado Iteratively Reweighted Least Squared (IRLS) [11]. Sobre este método, es interesante destacar su relación con los modelos clásicos de redes neuronales basados en unidades de tipo sigmoidal. Un estudio completo de la relación entre ambos modelos puede verse en [13] y [14].

En problemas reales, con frecuencia se comprueba que el análisis discriminante

[1] Este hecho es conocido como el *dilema bias-varianza* dentro de la comunidad estadística y en el lenguaje de redes neuronales suele plantearse en términos de precisión y complejidad del modelo, asociada a su capacidad de generalización, es decir, de predecir correctamente el valor asignado a aquellos datos nuevos que no pertenezcan al conjunto de entrenamiento.

lineal o cuadrático, o la regresión logística no son capaces de determinar las fronteras de decisión de un problema de clasificación cuando dichas fronteras de decisión alcanzan un alto nivel de no linealidad. Una técnica alternativa es no suponer nada acerca de las distribuciones de probabilidad de los datos y estimar directamente las diferentes clases del problema a partir de los datos de entrenamiento disponibles.

Una técnica tradicional es sustituir las variables de entrada por funciones no lineales convenientemente elegidas y construidas a partir de ellas, denominadas funciones de base, y posteriormente considerar un modelo lineal en el nuevo espacio formado por las variables transformadas. Por ejemplo, un desarrollo en serie de Taylor hasta el orden 2, incluyendo por tanto términos cuadráticos y las interacciones entre la variables, puede ser considerado como un primer intento para pasar de un modelo lineal a uno no lineal.

Métodos como el de aprendizaje siguiendo la proyección[2] [15], modelos aditivos generalizados[3] [16] y PolyMARS [17], un método híbrido basado en funciones de tipo spline (MARS)[4] [18], diseñado especificadamente para resolver problemas de clasificación, pueden ser vistos desde esta perspectiva. En la misma línea se encuentran las propuestas de Bose [19], que determinan las probabilidades condicionales de cada clase a partir de funciones splines cúbicas (CUS), o su posterior afinamiento usando splines lineales y sus productos tensoriales para resolver problemas de clasificación más complejos [20]. La mayor dificultad de estos métodos consiste en determinar de antemano la tipología de las funciones de base y su número.

Por último, las redes neuronales artificiales [21] han sido aplicadas en un amplio rango de problemas de clasificación. Aunque se han utilizado diferentes arquitecturas, la más frecuente ha sido la estructura de red con una sola capa oculta y funciones de activación de las neuronas de tipo sigmoide.

III. REDES NEURONALES BASADAS EN UNIDADES PRODUCTO

A continuación presentamos las redes neuronales basadas en unidades producto. Comenzaremos definiendo la estructura funcional del modelo no lineal y su representación mediante una red neuronal con unas determinadas características. Nos ocuparemos posteriormente de comentar diferentes aspectos del modelo, haciendo un

[2] Projection pursuit learning.
[3] Generalized additive models.
[4] MultiAdaptive Regression Splines.

recorrido por los métodos de entrenamiento que han sido utilizados hasta el momento para calcular los pesos de una red de este tipo, mostrando a su vez sus limitaciones.

A. ESTRUCTURA FUNCIONAL

Consideremos la familia de funciones definida por $F = \bigcup_{m\in\mathrm{N}} F_m$, donde:

$$F_m = \left\{ f : A \subset \mathbf{R}^k \to \mathbf{R}: f(x_1, x_2, ..., x_k) = \beta_0 + \sum_{j=1}^{m} \beta_j \left(\prod_{i=1}^{k} x_i^{w_{ji}}\right) \right\}$$

y el dominio de definición de f es el conjunto de $\mathbf{R}^k$ dado por $A = \left\{(x_1, x_2, ..., x_k) \in \mathbf{R}^k : 0 < x_i \right\}$.

Observemos que 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 F_m tienen la siguiente expresión: $f(x_1, x_2) = \beta_0 + \sum_{j=1}^{m} \beta_j x_1^{w_{j1}} x_2^{w_{j2}}$.

Por otra parte, es interesante observar que las funciones definidas en el modelo anterior pueden considerarse como una generalización de las superficies de respuesta. Observemos, por ejemplo, que si en el modelo para dos variables se eligen convenientemente los exponentes $w_{ji} \in \{0,1,2\}$ se obtiene una superficie de respuesta cuadrática del tipo:

$$f(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 .$$

y en general, para *k* variables, del tipo:

$$f(x_1, x_2, ..., x_k) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i<j=1}^{k} \beta_{ij} x_i x_j .$$

Merece atención el problema que surge con este tipo de redes cuando recibe valores negativos de las variables de entrada. En este caso, la salida de la red es un número complejo. Los problemas que vamos a abordar con este tipo de modelos tienen siempre valores reales, por lo que hemos de evitar el trabajo con números complejos. Durbin y Rumelhart [10] sugieren descartar la parte imaginaria y usar solo la parte real de la posible salida compleja de la red. Esta manipulación puede tener como consecuencia un aumento de la dimensión de Vapnik-Chervonenkis (dimensión VC) del modelo, lo que implica la disminución de su capacidad de generalización, es decir, de predecir correctamente datos que no pertenecen al conjunto de entrenamiento (véase [22] para más detalles). Para evitar este problema, restingiremos el dominio al

subconjunto A de $\mathbf{R}^k$ dado por $\left\{\mathbf{x}=(x_1,x_2,...,x_k)\in\mathbf{R}^k : x_i>0, i=1,2,...,k\right\}$.

B. REPRESENTACIÓN DE LA ESTRUCTURA FUNCIONAL EN UNA ESTRUCTURA DE RED NEURONAL

Las redes de tipo multiplicativo están basadas en nodos de la forma:

$$y_j=\prod_{i=1}^{k}x_i^{w_{ji}}$$

donde k es el número de variables de entrada. Cuando los exponentes en la expresión anterior son iguales a 0 ó 1, se obtienen las redes neuronales conocidas como sigma-pi. En el caso de las unidades producto, los exponentes w_{ji} son números reales y además no están fijados de antemano. La estructura funcional definida en el epígrafe anterior puede ser representada como una red neuronal con las siguientes características: una capa de entrada con k nodos, tantos como variables de entrada, una capa oculta con m nodos y una capa de salida con 1 nodo. No existen conexiones entre los nodos de una misma capa, ni conexiones directas entre los nodos de la capa de entrada y de salida. La función de activación del nodo *j* de la capa oculta está dada por $B_j(\mathbf{x},\mathbf{w}_j)=\prod_{i=1}^{k}x_i^{w_{ji}}$ donde w_{ji} es el valor del peso que conecta el nodo de entrada *i* con el nodo *j* de la capa intermedia. La función de activación del nodo de salida está dada por $\beta_0+\sum_{j=1}^{m}\beta_j B(\mathbf{x},\mathbf{w_j})$, donde $\mathbf{w_j}=(w_{j1},w_{j2},...,w_{jk})$.

De forma análoga, es posible definir la estructura de una red producto con *J* salidas. La función para cada salida está definida por $f_l(\mathbf{x};\boldsymbol{\theta}_l)$:

$$f_l(\mathbf{x};\boldsymbol{\theta}_l)=\beta_0^l+\sum_{j=1}^{m}\beta_j^l B_j(\mathbf{x},\mathbf{w}_j),\ l=1,2,...,J. \qquad (1)$$

donde $\boldsymbol{\theta}_l=(\boldsymbol{\beta}^l,\mathbf{w}_1,...,\mathbf{w}_m)$ y $\boldsymbol{\beta}^l=(\beta_0^l,\beta_1^l,...,\beta_m^l)$.

Es importante observar que la expresión $f(x_1,x_2,...,x_k)=\beta_0+\sum_{j=1}^{m}\beta_j(\prod_{i=1}^{k}x_i^{w_{ji}})$ es funcionalmente equivalente a la expresión:

$$f(x_1,x_2,...,x_k)=\beta_0+\sum_{j=1}^{m}\beta_j\exp\left(\sum_{j=1}^{k}w_{ji}\ln(x_i)\right).$$

A partir de esta ecuación, una red basada en unidades producto puede verse de forma equivalente a una red en la que se aplica una transformación logarítmica a las variables de entrada x_i y los nodos de la capa oculta son aditivos con función de

transferencia exponencial $\exp(t) = e^t$ [5]. Esta arquitectura equivalente resulta computacionalmente más eficiente.

C. VENTAJAS Y DESVENTAJAS DE LAS REDES DE UNIDADES PRODUCTO

Las redes neuronales basadas en unidades producto han demostrado buenos resultados en el modelado de datos en los que existen interacciones de diferentes órdenes entre las variables independientes del problema. Se ha comprobado la eficiencia en el modelado de determinado tipo de datos usando un número de nodos en la capa intermedia, inferior al que se necesitarían si se consideraran redes neuronales estándar. Durbin y Rumelhart [10] demostraron que la capacidad de información de una única unidad de tipo producto (medida como la capacidad para el aprendizaje de patrones booleanos aleatorios[6]) es aproximadamente igual a $3N$, comparado con el valor de $2N$ que corresponde a una unidad de tipo aditivo, siendo N el número de entradas de la unidad. Junto a esto, es posible obtener cotas superiores para redes basadas en unidades producto similares a las conocidas para las redes de tipo sigmoide [22]. Por último, como consecuencia del Teorema de Stone–Weierstrass, se ha demostrado que las redes de unidades producto son aproximadores universales [23] (obsérvese que las funciones polinómicas de varias variables son un subconjunto de los modelos basados en unidades producto). Sin embargo, las redes neuronales basadas en unidades producto presentan un inconveniente importante: la superficie de error asociada es especialmente compleja, con numerosos óptimos locales y regiones planas, y por tanto con mayor probabilidad de quedar atrapado en alguno [24, 25]. La estructura potencial del modelo provoca que pequeños cambios en los exponentes tengan como consecuencia un cambio significativo en los valores de la función y en la función de error. Así, los algoritmos de entrenamiento de redes basados en el gradiente quedan con frecuencia, y de forma especial en este tipo de redes neuronales, atrapados en óptimos locales. Por ejemplo, está estudiado el hecho de que el clásico algoritmo de retropropagación no funciona bien en este tipo de redes [26]. Janson y Frenzel [26] diseñan un algoritmo genético para evolucionar los pesos de una red basada en unidades producto con una estructura

[5] Obsérvese la diferencia que hay desde el punto de vista funcional con el modelo definido por las redes MLP dado por:

$$f_l(\mathbf{x}, \boldsymbol{\theta}) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l \sigma\left(\sum_{j=1}^{n} w_{ji} x_i - \theta_j \right), \; l = 1, 2, ..., J$$

donde σ es una función sigmoide.

[6] Random boolean patterns.

predefinida. El mayor problema de este tipo de algoritmos es cómo obtener la arquitectura óptima de antemano. Ismael y Engelbrecht [24, 25] aplicaron cuatro métodos diferentes de optimización para entrenar redes de unidades producto: algoritmo genético estándar, método de optimización mediante cúmulo de partículas (PSO) y LEAPFROG. Los resultados obtenidos muestran que estos algoritmos son eficientes en el entrenamiento de redes basadas en unidades producto, obteniéndose los mejores resultados con el algoritmo genético y con PSO. Es importante señalar, sin embargo, que las funciones usadas en la experimentación corresponden a casos sencillos, funciones polinómicas de baja dimensionalidad. En un trabajo posterior, [27] utilizan un algoritmo de poda donde evolucionan tanto la estructura como los pesos de la red de unidades producto. Leerink *et al.* [28] prueban diferentes métodos de optimización local y global para este tipo de redes. Los resultados obtenidos muestran que los algoritmos de búsqueda local, como el de retropropagación, suelen quedar atrapados en mínimos locales, y los de búsqueda global, como el de búsqueda aleatoria o enfriamiento simulado, no son eficientes para redes grandes. Sugieren alguna heurística para mejorar la retropropagación y la combinación de métodos de búsqueda global y local. En definitiva, los trabajos realizados sobre redes de unidades producto no han afrontado el problema simultáneo del diseño de la estructura de la red y la estimación de los pesos de las conexiones utilizando métodos clásicos o técnicas evolutivas. No obstante, las unidades producto han sido utilizadas en problemas de regresión [23,29,30,31,32].

D. APLICACIÓN DEL MODELO A PROBLEMAS DE CLASIFICACIÓN

Como señalamos en la introducción, una cuestión de especial importancia en el ámbito financiero es la de disponer de técnicas eficientes que permitan decidir si una inversión o un préstamo lleva aparejado un riesgo aceptable o no para la entidad correspondiente. La cuantificación de este riesgo en función de las características del solicitante (edad, cuantía del préstamo, sexo, salario, años de antigüedad en el empleo, estado civil, número de hijos…) es una tarea esencial para una entidad financiera. De la misma forma, una organización puede estar interesada en tener un criterio afinado que permita discriminar entre riesgo bajo, medio y alto, para controlar la exposición a diferentes tipos de riesgo para un problema determinado. Los problemas descritos son en realidad problemas concretos de clasificación[7] de patrones que pasamos a formular a

[7] Las técnicas de clasificación han sido aplicadas especialmente en el ámbito del diagnóstico de enfermedades. Las redes neuronales han obtenido resultados muy buenos en el diagnóstico de

continuación en su forma general.

En un problema de clasificación se recopila información acerca de varias características o variables x_i, $i = 1,2,...,k$, de cada individuo u objeto de la muestra analizada para, posteriormente, asignar dichos individuos a una de las J clases previamente establecidas, según el valor de las mencionadas variables. Se asume que J es finito, y las características x_i son observaciones aleatorias de estas clases.

Se dispone de una muestra de entrenamiento $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1,2,...,N\}$, donde $\mathbf{x}_n = (x_{1n},...,x_{kn})$ es el vector aleatorio de las medidas de las características analizadas, que tomará valores en $\Omega \subset \mathbf{R}^k$, e $\mathbf{y}_n$ es la clase a la que pertenece el *n*-ésimo individuo. Adoptamos la técnica común de representación de las diferentes clases mediante un vector de codificación "1-de-*J*" $\mathbf{y} = \left(y^{(1)}, y^{(2)}...,y^{(J)}\right)$, de manera que $y^{(l)} = 1$ si $\mathbf{x}$ corresponde a un ejemplo perteneciente a la clase l y, de otro modo $y^{(l)} = 0$. Basándonos en la muestra de entrenamiento, se pretende encontrar una función de clasificación $C : \Omega \to \{1,2,...,J\}$ que nos permita clasificar a los individuos. En otras palabras, dada una partición de Ω, $D_1, D_2,..., D_J$, donde D_l pertenece a la clase l, $l = 1,2,...,J$, y los valores de las características correspondientes a D_l, será clasificado como la clase l-ésima. El error de clasificación ocurre cuando la regla de decisión Ω asigna a un individuo (según su vector de características) a la clase j cuando realmente pertenece a la clase $l \neq j$. Se define el porcentaje de clasificación correcta por $CCR = \frac{1}{N}\sum_{n=1}^{N} I(C(\mathbf{x}_n) = \mathbf{y}_n)$, donde $I(\bullet)$ es la función de pérdida 0-1. Un buen clasificador tratará de conseguir el mayor CCR posible para un problema determinado.

Para abordar el problema de clasificación a partir de nuestro modelo de unidades producto, vamos a normalizar las salidas de la red para que sumen la unidad y puedan interpretarse como las probabilidades de pertenencia a cada clase. Consideramos la transformación[8]:

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{l=1}^{J} \exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}, \; l = 1,2,...,J.$$

---

enfermedades como el cáncer. Otra aplicación importante de las redes neuronales es el reconocimiento de caracteres.

[8] Esta transformación es conocida en la literatura de redes neuronales como transformación soft-max.

La transformación soft-max produce salidas positivas que suman 1 y que, por tanto, pueden ser interpretadas como probabilidades condicionadas de pertenencia a una clase:

$$p\left(y^{(l)}=1\middle|\mathbf{x},\boldsymbol{\theta}_j\right)=g_l\left(\mathbf{x},\boldsymbol{\theta}_l\right),\ l=1,2,...,J\ .$$

La red neuronal correspondiente, después de aplicar la transformación anterior, puede verse en la Figura 1.

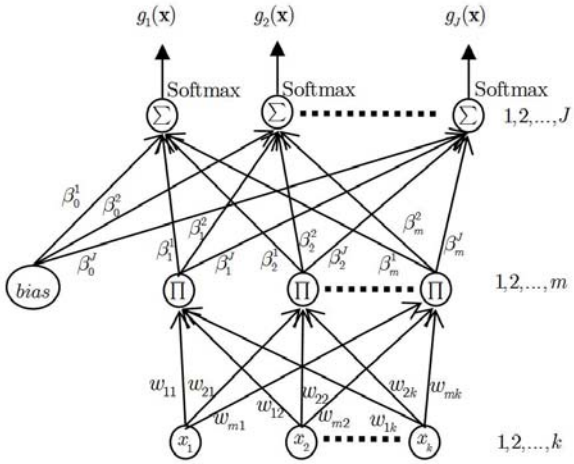

Fig. 1. Modelo de red neuronal de unidades producto para clasificación.

La regla de asignación $C(\mathbf{x})$ para un problema de clasificación es por tanto $C(\mathbf{x})=\hat{l}$ donde $\hat{l}=\arg\max_l g_l(\mathbf{x},\hat{\boldsymbol{\theta}})$ para $l=1,2,...,J$. En otras palabras, cada patrón será asignado a la clase correspondiente a la salida con máxima probabilidad. Por otra parte, la condición de normalización $\sum_{l=1}^{J} p\left(y^{(l)}=1\middle|\mathbf{x},\boldsymbol{\theta}_l\right)=1$ hace que una de las probabilidades no necesite ser estimada, con la consiguiente reducción del número de parámetros a estimar. Sin pérdida de generalidad, supondremos que $f_J(\mathbf{x},\boldsymbol{\theta}_J)=0$.

E. SOBRE LA FUNCIÓN DE ERROR PARA CLASIFICACIÓN: ENTROPÍA

Para la estimación del modelo, consideraremos la entropía cruzada[9] como función de error[10] [21]. Dados los datos del conjunto de entrenamiento $D=\{(\mathbf{x}_n,\mathbf{y}_n)\}$ con $x_{in}>0$,

[9] Cross-entropy, en inglés.

[10] Otra posibilidad es considerar el vector de salidas normalizadas de la red y calcular para cada patrón la distancia euclídea entre dicho vector y el vector objetivo de carácter binario formado por un 1 en la posición correspondiente a la clase a la que pertenece el patrón y ceros en el resto. Esta medida de error, conocida como error cuadrático medio o mean squared error (MSE) en inglés, es muy utilizada en los problemas de clasificación. Sin embargo, es importante destacar que la entropía resulta más adecuada que el error cuadrático medio para resolver problemas de clasificación (véanse por ejemplo

$\forall i,n$, el proceso de estimación del modelo se basa en el método de máxima verosimilitud en el que se intenta maximizar las probabilidades de pertenencia a la clase correspondiente para cada patrón del conjunto de datos de entrenamiento. Equivalentemente, y para simplificar los cálculos, usualmente se minimiza el valor negativo del producto del logaritmo de las probabilidades de pertenencia. De esta forma, la función de error a minimizar es:

$$l(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{J} y_n^{(l)} \log g_l\left(\mathbf{x}_n, \boldsymbol{\theta}_l\right) = \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \log\sum_{l=1}^{J}\exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l)\right] \qquad (2)$$

donde $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_J)$.

Desde un punto de vista estadístico, con la transformación soft-max y la función de entropía cruzada (2), el modelo puede verse como un modelo de regresión multilogística [11].

La superficie de error asociada a este modelo es generalmente bastante rugosa con numerosos óptimos locales. Además la matriz hessiana de la función de error $l(\boldsymbol{\theta})$ es en general indefinida. Este hecho, unido a la no linealidad de las funciones $f(\mathbf{x},\boldsymbol{\theta})$ y a que el número de funciones de base no esté fijado de antemano, son los motivos que nos han llevado a aplicar las técnicas de computación evolutiva para la optimización de la función de error. No aplicamos por tanto el método de tipo gradiente, denominado IRLS y usado clásicamente para la estimación de un modelo de regresión logística.[11]

IV. ESTIMACIÓN DE LOS PARÁMETROS: ALGORITMO EVOLUTIVO

Establecido el modelo no lineal con el que vamos a trabajar, se ha de fijar el procedimiento para determinar la estructura de red y los valores de los pesos que minimicen la función de error elegida. Frente a los métodos clásicos que fijan de antemano la estructura y determinan los pesos a partir de métodos de tipo gradiente, como la retropropagación, el gradiente conjugado o el método de Newton-Raphson, nosotros vamos a usar técnicas de programación evolutiva.

La evolución simultánea de la estructura de la red (nodos y conexiones) y de los valores de los pesos permite una exploración global del espacio de modelos posibles y

---

los argumentos esgrimidos en [33,34] a favor de la entropía). Junto a los argumentos presentados en las referencias anteriores a favor de la entropía, hay que señalar además que el rango de variación de esta es mucho mayor que en el MSE, al ser potencialmente no acotada (véase a este respecto el análisis en [35]). Este hecho es especialmente importante cuando usamos un esquema de tipo evolutivo en el que es necesario que el proceso no se estanque o bien cuando se utilizan búsquedas locales de tipo gradiente.

evita tener que determinar de antemano la estructura óptima del modelo. Los algoritmos evolutivos y la evolución de la población son por tanto una herramienta adecuada para realizar esta búsqueda en el espacio de conexiones y pesos.

Otra ventaja de los métodos evolutivos es su capacidad para trabajar con situaciones de discontinuidad en los datos o no diferenciabilidad de la correspondiente función de error. A menudo, los métodos clásicos de optimización aplicados en estadística o econometría sobre modelos no lineales fallan o quedan atrapados en óptimos locales cuando se trabaja con problemas reales en los que no se cumplen las hipótesis básicas de continuidad y/o diferenciabilidad.

En el campo de las redes neuronales, han existido diferentes intentos para diseñar la arquitectura de forma automática, como los métodos constructivos y los métodos de poda [36,37]. Sin embargo, estos métodos son propensos a quedar atrapados en óptimos locales.

Los algoritmos evolutivos aplicados a redes neuronales aparecen como una propuesta más eficiente que los métodos constructivos y de poda para diseñar la arquitectura óptima y encontrar los pesos de la red (véase los trabajos siguientes como una muestra de la teoría desarrollada en los últimos años sobre las redes evolutivas [38,39,40,41,42,43,44]). El proceso evolutivo, por tanto, determinará el número de funciones de base del modelo y los correspondientes coeficientes y exponentes.

A. ESTIMACIÓN DE LOS PARÁMETROS

En este apartado proponemos un algoritmo evolutivo para el aprendizaje de los parámetros de una red neuronal, minimizando la función de error de entropía cruzada. Construimos un algoritmo evolutivo para diseñar la estructura, así como para estimar los pesos de las conexiones de la red de unidades producto.

El proceso de búsqueda comienza a partir de una población inicial de redes de unidades producto y, en cada iteración, la población va modificándose utilizando un algoritmo de actualización. La población está sujeta a operadores de mutación y replicación. No se utiliza cruce por sus potenciales problemas en la evolución de redes neuronales [43,44]. Teniendo en cuenta las características del algoritmo propuesto, podemos concluir que se trata de un algoritmo evolutivo [45,46]. La estructura general del algoritmo evolutivo es la siguiente:

---

[11] Obsérvese a este respecto el aumento en la complejidad de la función de error cuando se sustituye el modelo lineal propio de la regresión logística en (2) por una función $f(\mathbf{x},\boldsymbol{\theta})$ de tipo producto.

(1) Generar una población inicial de tamaño N_P.

(2) Repetir hasta que se cumpla la condición de parada.

(a) Calcular la aptitud de cada uno de los individuos de la población.

(b) Ordenar los individuos con respecto a su aptitud.

(c) El mejor individuo se pasa a la nueva población.

(d) El 10% mejor de la población se copian y sustituyen al 10% peor de la población.

Sobre la población intermedia:

(e) Se aplica mutación paramétrica sobre los individuos del 10% mejor de la población.

(f) Se aplica mutación estructural sobre los individuos del 90% restante de la población.

Sea $l(\boldsymbol{\theta})$ la función de error del individuo g de la población. Obsérvese que g es una red de unidades producto y puede ser vista como una función vectorial $g(\mathbf{x},\boldsymbol{\theta}) = (g_1(\mathbf{x},\boldsymbol{\theta}_1),...,g_l(\mathbf{x},\boldsymbol{\theta}_l))$. La aptitud es una función estrictamente decreciente de la función de error $l(\boldsymbol{\theta})$ dada por $A(g) = \frac{1}{1+l(\boldsymbol{\theta})}$.

La mutación paramétrica se aplica a cada coeficiente w_{ji}, β_j^l introduciendo ruido gaussiano según la siguiente expresión:

$$\begin{aligned} w_{ji}(t+1) &= w_{ji}(t) + \xi_1(t) \\ \beta_j^l(t+1) &= \beta_j^l(t) + \xi_2(t) \end{aligned} \tag{3}$$

donde $\xi_k(t) \in N(0,\alpha_k(t))$, para cada $k = 1,2$, representa una variable aleatoria normal unidimensional normalmente distribuida de media 0 y varianza $\alpha_k(t)$. Una vez que la mutación es aplicada, la aptitud de cada individuo es recalculada y se aplica un algoritmo típico de enfriamiento simulado [47,48]. De esta forma, si ΔA es la diferencia de la función de actitud entre antes y después de aplicar la mutación, el criterio es: si $\Delta A \geq 0$, la mutación es aceptada, y si $\Delta A < 0$, la mutación es aceptada con una probabilidad $\exp(\Delta A / T(g))$, donde la temperatura $T(g)$ de un individuo g viene dada por $T(g) = 1 - A(g),\ 0 \leq T(g) < 1$.

La varianza $\alpha_k(t)$ se actualiza a lo largo del proceso evolutivo. Hay diferentes métodos para actualizar la varianza. En nuestro caso, utilizamos la regla de 1/5 de éxitos de Rechenberg [49], uno de los métodos más simples. Esta regla establece que la

proporción de mutaciones exitosas ha de ser 1/5. Por tanto, si la proporción de mutaciones exitosas es mayor a 1/5, la desviación de la mutación se incrementa en otro caso disminuye. De esta forma:

$$\alpha_k(t+s) = \begin{cases} (1+\lambda)\alpha_k(t) & si \quad s_g > 1/5 \\ (1-\lambda)\alpha_k(t), & si \quad s_g < 1/5 \\ \alpha_k(t) & si \quad s_g = 1/5 \end{cases} \tag{4}$$

donde $k = 1,2$, s_g es la proporción de mutaciones exitosas sobre s generaciones y $\lambda = 0.1$. El objetivo de esta adaptación es intentar evitar quedar atrapado en mínimos locales así como acelerar el proceso de búsqueda dentro de la evolución cuando las condiciones del entorno de búsqueda actuales sean buenas.

También hay que señalar que el grado de las modificaciones en los exponentes w_{ji} es diferente a la que se realiza sobre los coeficientes β_j^l, donde $\alpha_1(t) << \alpha_2(t)$.

La mutación estructural implica una modificación en la estructura de la red neuronal y posibilita la exploración en diferentes regiones en el espacio de búsqueda, ayudando a mantener la diversidad en la población.

Hay cinco tipos de mutaciones estructurales: eliminar nodos, eliminar conexiones, añadir nodos, añadir conexiones y fusión de nodos. Estas cinco mutaciones son aplicadas secuencialmente a cada red. Las primeras cuatro son similares a las mutaciones expuestas en el modelo GNARL [44]. En la mutación unir nodos, se eligen aleatoriamente dos nodos de la capa oculta, a y b, y se reemplazan por un nuevo nodo c, el cual es combinación de los dos anteriores. Las conexiones comunes entre los dos nodos se mantienen, con un peso dado por:

$$\begin{aligned} \beta_c^l &= \beta_a^l + \beta_b^l \\ w_{jc} &= \frac{w_{ja} + w_{jb}}{2} \end{aligned} \tag{5}$$

Las conexiones no comunes entre los nodos son heredadas por c con una probabilidad de 0.5, manteniéndose el peso. No obstante, los operadores de eliminación y unión se realizan con una mayor probabilidad ($T(g)$ para las mutaciones de eliminación y unión y $T^2(g)$ para las de suma). Si la mutación de eliminación o unión mejoran la aptitud de la red, no se realizan el resto de mutaciones estructurales. Si por probabilidad no se seleccionara ninguna mutación, se elige un tipo aleatoriamente y se aplica a la red.

El criterio de parada se alcanza si se cumple alguna de las siguientes condiciones:

se alcanza un número de generaciones determinado o la varianza de la aptitud del 10% mejor de la población es menor que 10^{-4}.

Los parámetros utilizados en el algoritmo evolutivo son comunes para todos los problemas. Hemos considerado $\alpha_1(0) = 0.5$, $\alpha_2(0) = 1$, $\lambda = 0.1$ y $s = 5$. Los exponentes w_{ji}, y los coeficientes β_j^l se inicializan en el intervalo $[-5,5]$. El máximo número de nodos ocultos es $m = 6$. El tamaño de la población es $N_P = 1000$. El número de nodos que pueden ser añadidos o eliminados en una mutación estructural puede ser $\{1,2\}$. El número de conexiones que pueden ser añadidas o eliminadas de la red se elige dentro del intervalo $[1,c]$, donde c es un tercio de número de conexiones de la red. Se ha considerado un máximo de 400 generaciones.

B. ESCALADO

En las aplicaciones a problemas reales de clasificación que llevaremos a cabo, realizaremos un escalado previo de los datos mediante una transformación lineal en el intervalo $[1,2]$. La cota inferior ha sido elegida para evitar valores de las variables cercanos a cero que podrían generar valores del modelo muy elevados cuando los exponentes son negativos. La cota superior se ha elegido para evitar cambios bruscos en la función de error cuando los exponentes toman un valor elevado.

V. **EXPERIMENTOS**

Para examinar el riesgo de impago en tarjetas de crédito, hemos usado una base de datos utilizada por Baesens y otros [4] denominada German Credit Card. Incluye información sobre 20 variables explicativas o características correspondientes a 1000 personas titulares de tarjetas de crédito que nos permiten clasificarlas en dos categorías excluyentes según exista o no riesgo de impago.

Por lo tanto, la variable dependiente del modelo de clasificación tomará el valor 0 si no hay riesgo de impago, y 1 si lo hay. La base de datos incluye 300 casos de impago.

Las variables categóricas han sido transformadas en variables binarias, una por cada categoría, por lo que en total se han utilizado 61 variables, tal y como se puede observar en la Tabla 1, la cual incluye la descripción de las variables utilizadas y algunos estadísticos descriptivos sobre las mismas.

TABLA 1: DESCRIPCIÓN DE LAS VARIABLES INDEPENDIENTES UTILIZADAS PARA LA CLASIFICACIÓN

Variable	Definición	Tipo	Mediana	Mín.	Máx.
1 a 4	Estado de la cuenta corriente	Categórica, 4 posibles valores: desde menos de 0 marcos (descubierto) hasta más de 200 marcos (1€=1.95583 marcos alemanes), o no tiene cuenta corriente	0	0	1
5	Duración (en meses)	Continua	18	4	72
6 a 10	Historial de crédito	Categórica, 5 categorías: desde no existe historial hasta retrasos en el pago	0	0	1
11 a 21	Propósito	Categórica, 11 posibilidades según tipo de compra: coche, mobiliario, negocios, etc.	0	0	1
22	Cantidad de crédito	Continua	2319.5	250	18424
23 a 27	Cuentas de ahorro	Categórica, 5 posibles valores: desde menos de 100 marcos hasta más de 1000 marcos, o no tiene cuenta de ahorro	0	0	1
28 a 32	Años en el empleo actual	Categórica, 5 valores posibles: desde desempleado hasta más de 7 años	0	0	1
33	Cuotas de pago (% sobre ingresos disponibles)	Continua	3	1	4
34 a 38	Sexo y estado civil	Categórica, 5 posibles combinaciones: entre hombre o mujer, soltero, casado, divorciado, separado o viudo	0	0	1
39 a 41	Otros deudores-avalistas	Categórica, 3 valores: ninguno, co-solicitante, avalista	0	0	1
42	Años en la residencia actual	Continua	3	1	4
43 a 46	Tipo de propiedad	Categórica, 4 categorías: desde propietario de inmueble, seguro de vida, coche u otro, no propiedad o desconocido	0	0	1
47	Edad en años	Continua	33	19	75
48 a 50	Otros pagos pendientes	Categórica, 3 posibles valores: banco, tiendas, ninguna	0	0	1
51 a 53	Régimen de vivienda	Categórica: alquiler, propiedad, renta libre	0	0	1
54	Número de créditos existentes en este banco	Continua	1	1	4
55 a 58	Categoría laboral	Categórica, 4 categorías: desde desempleado o incapacitado hasta directivo o autoempleo	0	0	1
59	Número de personas a su cargo	Continua	1	1	2
60	Teléfono	Categórica: sí o no	-1	-1	1
61	Trabajador extranjero	Categórica: sí o no	1	-1	1

A. COMPARACIÓN CON OTROS MÉTODOS: ANÁLISIS DISCRIMINANTE (AD), REGRESIÓN LOGÍSTICA (RL) Y MLPBP

Para realizar la experimentación, hemos dividido la base de datos en 10 partes del mismo tamaño y hemos aplicado el procedimiento estándar de validación cruzada en las diferentes metodologías consideradas para la comparación (MLPBP, AD Y RL). El algoritmo evolutivo se ha ejecutado 10 veces para cada partición, mientras que los algoritmos de regresión logística (RL) y de análisis discriminante (AD) se ejecutaron una vez para cada partición. En la Tabla 2 se muestran los resultados medios del porcentaje de patrones correctamente clasificados (CCR) y la desviación típica para cada uno de los métodos.

Como puede verse, el algoritmo evolutivo basado en unidades producto obtiene el mejor resultado (mayor CCR medio) entre todas las metodologías consideradas. Se ha realizado un test de comparación de los CCR medios obtenidos con las diferentes metodologías (muestras relacionadas), por el cual se ha detectado la existencia de diferencias significativas entre las redes unidades producto (RNUP) y el resto de metodologías con un nivel de significación del 5%.

TABLA 2: RESULTADOS ESTADÍSTICOS SOBRE EL CONJUNTO DE GENERALIZACIÓN DEL MODELO RNPU Y LOS MÉTODOS MLPBP, AD, RL

Método	**CCR**	**Desv. Típica**
MLPBP	73.2	3.61
RNUP	76.3	4.8
AD	71.7	6.25
RL	75.7	5.3

VI. CONCLUSIONES Y TRABAJOS FUTUROS

En el presente trabajo se ha mostrado un tipo de redes neuronales, denominadas redes neuronales basadas en unidades producto (RNUP), como un modelo no lineal que puede ser utilizado para la resolución de problemas de clasificación en aprendizaje. Estas redes neuronales no han sido muy utilizadas por los investigadores en el área de ciencias de la computación y, como consecuencia, tampoco en otras áreas como la economía y las finanzas en las que, sin embargo, sí han encontrado aplicación las redes neuronales en su formulación clásica determinada por el perceptrón multicapa (MLP). Junto al modelo no lineal, se propone un método evolutivo en el que simultáneamente se diseña la estructura de la red y se calculan los correspondientes pesos. Para evaluar el rendimiento de los modelos de clasificación obtenidos, comparamos nuestra propuesta

con varias técnicas clásicas como la regresión logística o el análisis discriminante y también con el clásico modelo perceptrón multicapa (MLP) de redes neuronales basado en unidades sigmoides. Los resultados obtenidos por el modelo RNUP muestran que la propuesta realizada puede ser una alternativa interesante en la resolución de problemas de clasificación en el ámbito económico a los métodos clásicos estadísticos, como por ejemplo el análisis discriminante y la regresión logística, e incluso a las redes MLP.

La investigación realizada abre diferentes líneas de trabajo. La primera sería realizar una experimentación más completa, incluyendo un mayor número de bases de datos con diferentes características, que permitiera obtener resultados de comparación más consistentes. En segundo lugar, el mismo modelo puede ser aplicado a problemas de regresión y, con algunas modificaciones en la estructura de la red, a problemas de predicción de series temporales. Por último, una tercera línea de trabajo estaría basada en un enfoque multiobjetivo del problema de clasificación o de regresión, en donde se considerara no solo la precisión del modelo, sino también su complejidad. La obtención de modelos más sencillos, con un menor número de parámetros facilitaría la interpretación.

AGRADECIMIENTOS

Este trabajo ha sido financiado en parte por el proyecto TIN 2005-08386-C05-02 de la Comisión Interministerial de Ciencia y Tecnología (MICYT) y fondos FEDER. Deseamos también expresar nuestro agradecimiento a los revisores anónimos que con sus valiosas sugerencias han mejorado sin duda la calidad del presente trabajo.

REFERENCIAS

1. Hawley, D., Johnson, J., y Raina, D., Artificial Neural System: A new tool for financial decision-making. Financial Analysts Journal 23 (1990) 63-72.
2. Refenes, A.P., Neural networks in the capital markets, New York Wiley (1995).
3. Parisi, A., Parisi, F., y Díaz, D., Modelos de Algoritmos Genéticos y Redes Neuronales en la Predicción de Índices Bursátiles Asiáticos. Cuadernos de Economía, 43 (2006) 251-284.
4. Baesens, B., Setiono, R., Mues, C., y Vanthienen, J., Using neural network rule extraction and decision tables for credit -risk evaluation. Management Science, 49 (2003) 312-329.
5. Mcnelis, P.D., Neural Networks in Finance: Gaining Predictive Edge in the Market. Advanced Finance Series Elsevier Academic Press (2005).

6. Coleman, K.G., Graettinger, T.J., y Lawrence, W.F., Neural Networks for Bankruptcy Prediction: The Power to Solve Financial Problems. AI Review, (1991) 48-50.
7. Brockett, P.W., Cooper, W.W., Golden, L.L., y Pitaktong, U., A neural network method for obtaining an early warning of insurer insolvency. The Journal of Risk and Insurance, 6 (1994) 402-424.
8. Martín-del Brio, B. y Serrano-Cinca, C., Self-organizing Neural networks: The financial State of Spanisch Companies, in Neural networks in the Capital Markets, A.P. Refenes, Editor: Wiley. 341-357 (1995).
9. Herbrich, D., Keilbach, M., Graepel, T., Bollmann-Sdorra, P., y Obermayer, K., Neural Networks in Economics: Background, applications and new developments, in Advances in Computational Economics: Computational techniques for Modelling Learning in Economics, T. Brenner, Editor, Kluwer Academics. 169-196 (2000).
10. Durbin, R. y Rumelhart, D., Products Units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation, 1 (1989) 133-142.
11. Hastie, T., Tibshirani, R.J., y Friedman, J., The Elements of Statistical Learning. Data mining, Inference and Prediction, in Springer. (2001).
12. McCullagh, P. y Nelder, J.A., Generalized Linear Models, 2nd edn., ed. C. Hall, London (1989).
13. Schumacher, M., Robner, R., y Vach, W., Neural networks and logistic regression: Part I. Computational Statistics & Data Analysis, 21 (1996) 661-682.
14. Vach, W., Robner, R., y Schumacher, M., Neural Networks and logistic regression: Part II. Computational Statistics & Data Analysis, 21 (1996) 683-701.
15. Friedman, J. y Stuetzle, W., Proyection pursuit regression. Journal of the American Statistical Association, 76 (376) (1981) 817-823.
16. Hastie, T.J. y Tibshirani, R.J., Generalized Additive Models, London Chapman & Hall (1990).
17. Kooperberg, C., Bose, S., y Stone, C.J., Polychotomous Regression. Journal of the American Statistical Association, 92 (1997) 117-127.
18. Friedman, J., Multivariate adaptive regression splines (with discussion). Ann. Stat., 19 (1991) 1-141.
19. Bose, S., Classification using splines. Computational Statistics & Data Analysis, 22 (1996) 505-525.
20. Bose, S., Multilayer statistical classifiers. Computational Statistics & Data Analysis, 42 (2003) 685-701.
21. Bishop, M., Neural Networks for Pattern Recognition Oxford University Press (1995).

22. Schmitt, M., On the Complexity of Computing and Learning with Multiplicative Neural Networks. Neural Computation, 14 (2001) 241-301.
23. Martinez-Estudillo, A., Martinez-Estudillo, F., Hervas-Martinez, C., y Garcia-Pedrajas, N., Evolutionary product unit based neural networks for regression. Neural Networks, 19 (4) (2006) 477-486.
24. Ismail, A. y Engelbrecht, A.P. Training products units in feedforward neural networks using particle swarm optimisation. in Development and practice of Artificial Intelligence Techniques, Proceeding of the International Conference on Artificial Intelligence Durban, South Africa In V.B. Bajic & D. Sha (Eds). (1999)
25. Ismail, A. y Engelbrecht, A.P. Global optimization algorithms for training product units neural networks. in International Joint Conference on Neural Networks IJCNN`2000 Como, Italy. (2000)
26. Janson, D.J. y Frenzel, J.F., Training product unit neural networks with genetic algorithms. IEEE Expert, 8 (5) (1993) 26-33.
27. Ismail A., E.A.P. Pruning product unit neural networks. in Proceedings of the International Conference on Neural Networks Honolulu, Hawai. (2002)
28. Leerink, L.R., Giles, C.L., Horne, B.G., y Jabri, M.A., Learning with products units. Advances in Neural Networks Processing Systems, 7 (1995) 537-544.
29. Saito, K. y Nakano, R., Extracting Regression Rules From Neural Networks. Neural Networks, 15 (2002) 1279-1288.
30. Saito, K. y Nakano, R. Numeric law discovery using neural networks. in Proc. of the 4th International Conference on Neural Information Processing (ICONIP97). (1997)
31. Engelbrecht, A.P. y Ismail, A., Training product unit neural networks. Stability and Control: Theory and Applications, 2 (1-2) (1999) 59-74.
32. Martinez-Estudillo, A.C., Hervas-Martinez, C., Martinez-Estudillo, F.J., y Garcia-Pedrajas, N., Hybridization of evolutionary algorithms and local search by means of a clustering method. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics, 36 (3) (2006) 534-545.
33. Joost, M. y Schiffmann, W., Speeding up backpropagation algorithms by using Cross-Entropy combined with Pattern Normalization. International Journal of Uncertainty Fuzziness and Knowledge-Based Systems, 6 (2) (1998) 117-126.
34. Bishop, C.M., Pattern Recognition and Machine Learning. Information Science and Statistics, ed. M. Jordan Springer (2006).
35. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., y Nielsen, H., Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics, 16 (5) (2000) 412-424.
36. Reed, R., Pruning algorithms-A survey. IEEE Transactions on Neural Networks, 4 (1993) 740-747.

37. Setiono, R. y Hui, L.C.K., Use of quasinewton method in a feedforward neural-network construction algorithm. IEEE Trans. Neural Networks, 6 (1995) 273-277.
38. Yao, X., Evolving artificial neural network. Proceedings of the IEEE, 9 (87) (1999) 1423-1447.
39. García-Pedrajas, N., Hervás-Martínez, C., y Muñoz-Pérez, J., Multiobjetive cooperative coevolution of artificial neural networks. Neural Networks, 15 (10) (2002) 1255-1274.
40. Yao, X. y Liu, Y., Making use of population information in evolutionary artificial neural networks. IEEE Transactions and System Man and Cybernetics-Part B: Cybernetics, 28 (3) (1998) 417-425.
41. Yan, W., Zhu , Z., y Hu, R. Hybrid genetic /BP algorithm and its application for radar target classification. in Proceedings of the IEEE National Aerospace Electronics Conference Piscataway, NJ, USA IEEE Press. (1997)
42. 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)
43. Yao, X. y Liu, Y., A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks, 8 (3) (1997) 694-713.
44. Angeline, P.J., Saunders, G.M., y Pollack, J.B., An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5 (1) (1994) 54-65.
45. Fogel, D.B., Owens, A.J., y Wals, M.J., Artificial Intelligence Throught Simulated Evolution, New York Wiley (1966).
46. Fogel, D.B., Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, New York IEEE Press (1995).
47. Kirkpatrick, S., Gellat, C.D.J., y Vecchi, M.P., Optimization by simulated annealing. Science, 220 (1983) 671-680.
48. Otten, R.H.J.M. y van Ginneken, L.P.P.P., The annealing algorithm, Boston, MA. Ed. Kluwer (1989).
49. Rechenberg, I., Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution, Stuttgart Framman-Holzboog Verlag (1975).