Asepelt España

Comunicaciones

XIV Reunión

USO DE METAHEURÍSTICOS PARA EL APRENDIZAJE DE REDES NEURONALES: UNA COMPARATIVA

Joaquín Pacheco - jpacheco@ubu.es
Alberto Aragón - aaragon@ubu.es
Universidad de Burgos

Anales de Economía Aplicada

Oviedo 22 23 24 Junio 2000

Este documento ha sido extraído del CD Rom "Anales de Economía Aplicada. XIV Reunión ASEPELT-España. Oviedo, 22 y 23 de Junio de 2000".
ISBN: 84-699-2357-9

USO DE METAHEURÍSTICOS PARA EL APRENDIZAJE DE REDES NEURONALES: UNA COMPARATIVA

Joaquín Pacheco y Alberto Aragón
Dpto.Economía Aplicada. Universidad de Burgos
Fac. C Económicas y Empresariales
C/Parralillos s/n
BURGOS 09006

Resumen.- Muchas de las modelos de redes neuronales existentes, como el conocido Perceptrón Multicapa o la más reciente Red de Expansión usan métodos basados en el gradiente para su aprendizaje (ej:algoritmo de propagación hacia atrás). Estos métodos tienen el defecto de converger a óptimos locales no globales en la mayoría de los casos. Además tienen el problema de no poder usarse con funciones de transferencia no derivables. En este trabajo se proponen métodos de aprendizaje basados en Técnicas Metaheurísticas como Temple Simulado, Búsqueda Tabú o, la más reciente, Scatter Search. Dichas técnicas han mostrado en los últimos años una gran capacidad para obtener soluciones de calidad tanto en problemas discretos como continuos. En este trabajo se realiza un amplio análisis de cada una de ellas por separado para este problema así como una comparativa entre ellas y el algoritmo de propagación hacia atrás.

Palabras Clave: Redes Neuronales, Aprendizaje, Temple Simulado, Búsqueda Tabú, Scatter Search

Area Temática Preferible: G1

1.- EL PROBLEMA

El perceptrón Multicapa es un modelo de red neuronal consistente en varias capas de neuronas: una de entrada, 1 o más capas ocultas (por lo general no excede de 2) y una de salida. La información se transmite de cada nodo a los nodos de la capa siguiente, desde la entrada a la salida.

La información que sale de un nodo a cada nodo de la capa siguiente se pondera por un peso w_{ji}; de forma que la información total que llega a cada nodo j de la nueva capa es $\sum_i w_{ji} \cdot x_i$, siendo x_i el valor de salida de los nodos i de la capa anterior. Este valor de entrada , para los nodos de las capas ocultas y la de salida, se transforma mediante una función de transferencia f produciendo la salida del nodo j $x_j = f(\sum_i w_{ji} \cdot x_i)$.

Las funciones de transferencia más usadas son la sigmoide $y = \dfrac{1}{1+e^{-x}}$, y la identidad o lineal y = x.

Para el aprendizaje de una red se usan vectores de entrenamiento, consistentes cada uno en un vector de entrada y un vector de salida. Para unos determinados pesos w_{ji} y cada vector de entrenamiento, la red lee el vector de entrada y se compara la salida producida por la red (salida observada) con la con los valores del vector de salida (salida esperada). El problema consiste en determinar los valores de los pesos w_{ji} de forma que la suma de los errores al cuadrado (E) en el conjunto de vectores de entrenamiento y valores de salida sea mínima. Las soluciones son en realidad vectores de pesos $\mathbf{w} = (w_{ji})$.

Los algoritmos basados en el gradiante como el método de *Propagación hacia atrás* (presentado por Werbos, (1.974), y posteriormente por Parker (1.985), LeCun (1.986) y Rumelhart y otros (1.986)) básicamente usan la siguiente fórmula de actualización de pesos

$$w_{ji} = w_{ji} - \eta \frac{\partial E}{\partial w_{ji}}$$

donde es η conocido como *parámetro de aprendizaje*. Es un proceso iterativo que finaliza cuando no hay mejora en E.

Precisamente una de las limitaciones de estos métodos, es la convergencia a óptimos locales no globales y su dependencia de la solución inicial. Además no son aplicables a modelos de redes que usan funciones de transferencia no derivables.

A continuación se van a proponer algoritmos de aprendizaje basadas en la estrategia Metaheurística denominada *Scatter Search* y se va a realizar un análisis de su aplicación. También se analizarán métodos basados en otros Metaheurísticos, como *Temple Simulado* o *Búsqueda Tabú*, así como métodos de mejora por entornos.

Para los primeros análisis y pruebas usaremos un modelo con f. de transferencia identidad con un conjunto de 500 vectores de entrenamiento que siguen la relación

$$\underline{Z = 3X + 2Y + 4}$$

La red constará de una capa de entrada con 3 nodos (X, Y y termino independiente 1), una capa oculta con 5 nodos y una de salida con un nodo (Z).

Esta relación es muy sencilla y en estos casos, cuando existe una relación lineal, el algoritmo de propagación hacia atrás suele converger al óptimo con relativa facilidad. Sin embargo entendemos que nos servirá para realizar análisis previos de los métodos propuestos y sus limitaciones.

Posteriormente realizaremos estudiaremos los resultados con relaciones no lineales entre la entrada y la salida.

Este trabajo se estructura de la siguiente manera en la siguiente sección se expondrá un sencillo método de movimientos entre soluciones por entornos, así como diferentes métodos de mejora a los que da lugar; en las tres secciones siguientes se propondrán tres metaheurísticos pare el aprendizaje de redes neuronales basados respectivamente, en procesos de *Scatter Search*, *Temple Simulado* y *Búsqueda Tabú*, con especial incapié en el primero; en la sección 6 se realizará un análisis comparativo de los métodos expuestos en este trabajo, así como el *algoritmo de propagación hacia atrás*; en la sección 7 se repetirán estos mismos análisis pero usando una con función de transferencia sigmoide y diferentes relacione funcionales a identificar; finalmente en la sección 8 se establecerán las concusiones.

2.- MEJORA POR ENTORNOS

En este apartado vamos a analizar las posibilidades de mejora de una determinada solución por medio de cambios a soluciones vecinas que se encuentren en un entorno suyo. Para obtener una solución vecina sencillamente se genera un valor aleatorio entorno a cada componente de la solución actual con un determinado radio.

Más concretamente, sea $\boldsymbol{w}$ una solución cualquiera de componentes w_l y *radio* un número real suficientemente pequeño, se definen los siguientes procedimientos

Procedimiento Perturbar_Peso(***w***, *radio*, ***w'***)

Generar una solución ***w'*** *cuyos componentes* w_l*' toman valores aleatorios en el intervalo (*w_l *- radio* w_l *+ radio)*

Como se verá en apartados siguientes esta forma de obtener soluciones vecinas en realidad es la usada por los procedimientos Temple Simulado, Búsqueda Tabú así como por el procedimiento de mejora propuesto para Scatter Search. Lo que se pretende en este apartado es analizar y profundizar en este método de búsqueda y mejora por entornos para un mejor funcionamiento de los Metaheurísticos antes mencionados.

Para ello sea $\boldsymbol{w}_0$ una solución cualquiera el parámetros entero *n_vecinos*, el parámetro real *radio* se define el siguiente

Procedimiento Mejor_Vecino($\boldsymbol{w}_0$, *radio, n_vecinos,* ***w'***)
min = ¥
Para i = 1 hasta n_vecinos hacer
  Perturbar_Peso($\boldsymbol{w}_0$ *, radio,* ***w''****)*
  *Si E(****w'****) < min hacer:* ***w'*** *=* ***w''*** *y min = E(****w''****)*

Así mismo, sea ***w*** una solución inicial se define el siguiente

Procedimiento Mejora_Radio_Variable(***w***, *radio0*, *alfa*, *n_vecinos*, *max_int*)
Hacer radio = radio0, n_int = 0;

Repetir

  *Ejecutar Mejor_Vecino(****w****, radio, n_vecinos,* ***w'****)*
  *Si E(w') < E(w) hacer: w = w', radio = radio*alfa y n_int = 0*
  en caso contrario hacer: radio = radio/alfa y n_int = n_int+1
hasta n_int = max_int

3.- SCATTER SEARCH

Scatter Seach es una estrategia poblacional-evolutiva caracterizada básicamente por el uso de un *Conjunto de Referencia* (*RefSet*) de soluciones. Dicho conjunto esta formado por dos Subconjuntos: Subconjunto de Calidad (*RefSet1*) formado por las mejores soluciones y Subconjunto de Diversidad (*RefSet2*) formado por las soluciones más diferentes con el resto de *RefSet.* En cada paso o ciclo se generan nuevas soluciones a partir de las del Conjunto de Referencia que actualizan éste. Amplios y recientes tutoriales sobre *Scatter Search* se pueden encontrar en Glover y otros (1.999) y Laguna (1.999).

3.1.- Los Algoritmos

Se ha diseñado una versión estática y una versión dinámica de esta estrategia para este problema. La descripción en seudocódigo de forma muy general es la siguiente

Procedimiento Scatter Search Estático
Generar un conjunto inicial de soluciones aleatorias

A partir de estas soluciones obtener RefSet inicial
Repetir
  Construir subconjuntos de 2 y 3 elementos de RefSet

Obtener nuevas soluciones a partir de dichos subconjuntos
Actualizar RefSet con las nuevas soluciones, (una vez obtenidas todas) (1)
Hasta que RefSet se estabilice (i.e. no haya nuevas soluciones)

El Procedimiento Scatter Search Dinámico es similar cambiando la sentencia (1) por la siguiente
Cada vez que se obtiene una nueva solución actualizar RefSet(1')

En otras palabras: en la versión estática se generan las nuevas soluciones y, una vez obtenidas, con las ya existentes en *RefSet* se actualiza este; en la dinámica, en el momento que se genera una nueva solución se comprueba si mejora el valor de la función objetivo *E* de alguna de las existentes en *RefSet1* o aporta más diversidad que alguna de *RefSet2*, en cuyo caso se incorpora sin esperar a generar las demás.

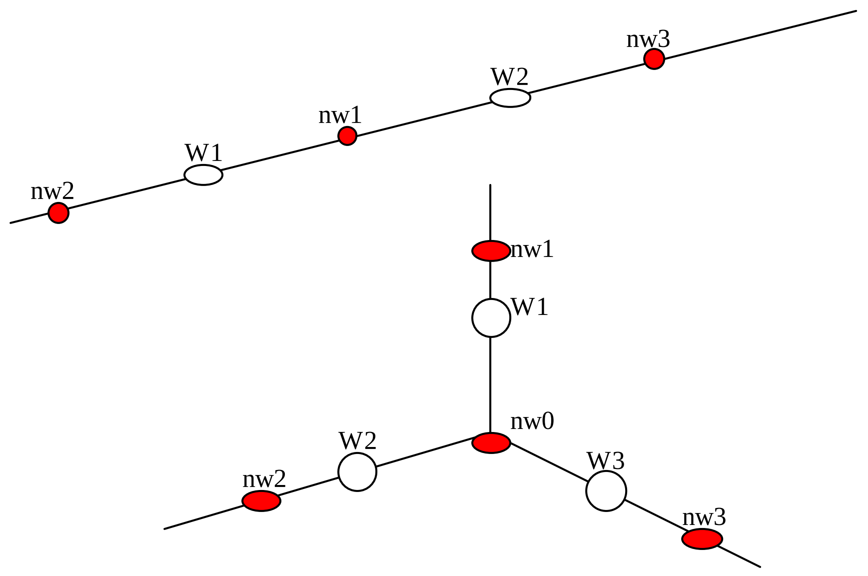
Para medir la diversidad de una solución ***w*** respecto *RefSet* se usa la siguiente función

$$dmin(\boldsymbol{w}) = \min \{ d(\boldsymbol{w}, \boldsymbol{w'}) \ / \ \boldsymbol{w'} \in R\};$$

donde $d(\boldsymbol{w}, \boldsymbol{w'}) = \Sigma_i |w_i - w_i'|$. ($w_i$ cada componente del vector de pesos ***w***).

Los subconjuntos de 2 elementos se forman con todos los pares de *RefSet*, y los de 3 elementos se forman al añadir a todos los subconjuntos de 2 el mejor elemento de *RefSet1* no presente.

La forma de generar nuevos soluciones está inspirada en Kelly y otros (1.996). Para conjuntos de 2, w1 y w2 se generan las siguientes nuevas: nw1 = (w1+w2)/2, (convexa); nw2 = w1-d y nw3 = w2+d, (no-convexas), donde d = (w2-w1)/2. Para conjuntos de 3, w1, w2 y w3 se generan las siguientes nuevas nw0 = (w1+w2+w3)/3 (convexa)., nwi = wi+(wi-nw0)/2, i=1,..3, (no-convexas). Según muestra la gráfica



Formas de generar nuevas soluciones con subconjuntos de 2 y 3

Se define *TamRef* = |*RefSet*|, análogamente *TamRef1* = |*RefSet1*| y *TamRef2* = |*RefSet2*|. Inicialmente se toma *TamRef1* = *TamRef2* = 5.

Esta primera aproximación no contempla 2 elementos fundamentales que aparecen en los tutoriales sobre *Scatter Search* antes mencionados: Un procedimiento de mejora, que se aplica a cada nueva solución generada; y un procedimiento de diversificación que se aplica cuando *RefSet* se ha estabilizado para dar generar nuevas soluciones de partida. Más adelante incorporaremos y analizaremos dichos elementos.

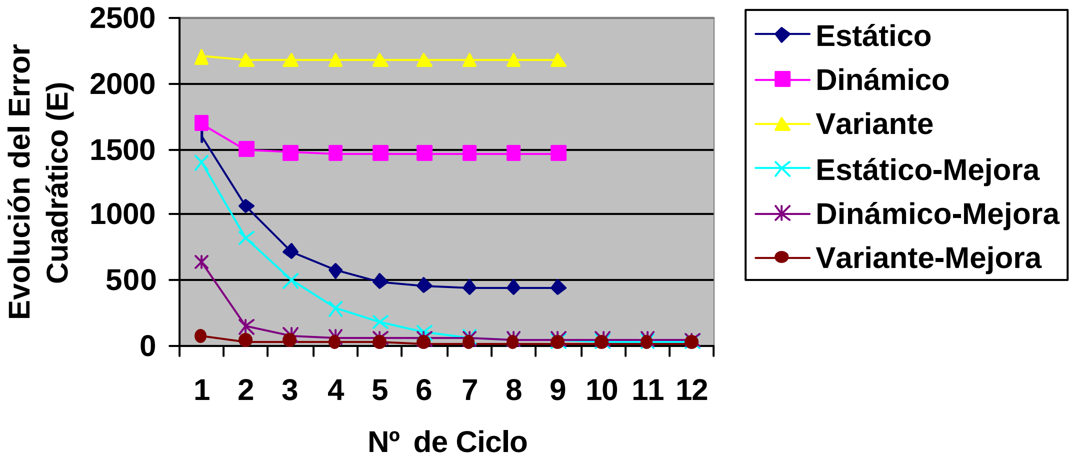
Además de estas dos estrategias Estática y Dinámica analizaremos una variante de la Dinámica que aparece propuesta en los tutoriales antes mencionados. Esta variante consiste en, una vez obtenido el conjunto de referencia inicial, considerar *RefSet1* = *RefSet*, y *RefSet2* = ∅ (i.e. *TamRef1* = 10, en este caso y *TamRef2* = 0). En otras palabras, solo se consideraran incorporaciones a *RefSet* que mejoren la función objetivo de alguno de sus elementos, y no se incorporarán soluciones por diversidad.

3.3.- Incorporación de un método de mejora

A continuación se propone incorporar uno de los elementos de Scatter Search que se ha mencionado anteriormente: un procedimiento de mejora que se aplica a cada solución nueva que se genera.

En nuestro caso el procedimiento de mejora es la ejecución del procediemiento Mejora_Radio_Variable inicialmente con *radio* = 0.1, *alfa* = 1, *n_vecinos* = 1, *max_int*=1. Es decir se comprueba si alrededor de cada nueva solución ***w*** que se genere existe alguna otra solución que la puede mejorar. Para ello se genera una solución ***w'*** cercana, si esta mejora a ***w*** la sustituye. El proceso se repite hasta que no haya más mejoras. De esta forma se espera conseguir soluciones ligeramente mejores. Por otra parte, como solo se genera una solución ***w'*** en torno a ***w*** se espera que el tiempo de computación no aumente excesivamente.

Para contrastar la eficacia de la incorporación de este método de mejora se han generado 50 conjuntos de soluciones aleatorias iniciales como en el subapartado anterior. A continuación se muestran las gráficas de los resultados medios obtenidos en cuanto a valor de *E* y tiempo de computación.



Las gráficas muestran claramente que la incorporación de este simple procedimiento de mejora reduce mucho el error en las tres versiones, llevándolas a niveles que empiezan a ser aceptables.

3.- TEMPLE SIMULADO

El *Temple Simulado* es una conocida estrategia Metaheurística propuesto por Kirpatrick, Gelatt y Vecci, (1.982) y (1.983) basado en los principios de la Física Termodinámica, más concretamente en el proceso de enfriamiento de determinados materiales. Sea ***w*** una solución inicial, c_0, *alfa* y *radio* tres parámetros reales esta estrategia puede ser adaptada fácilmente al problema que estamos tratando de la siguiente forma

Procedimiento <u>Temple Simulado</u>(***w***, c_0, *alfa*, *radio*)
Hacer $c = c_0$ *, niter = 0 y* ***w**** *=* ***w***
<u>Repetir</u>

*<u>Perturbar_Peso</u>(**w**, radio, **w'**)*

Si E(w') < E(w) o $\left(Random < \exp\left(\frac{E(w)-E(w')}{c}\right)\right)$ *hacer: w = w' y niter = 0;*

en caso contrario hacer niter = niter + 1

*Si E(**w**) < E(**w***) hacer **w*** = **w***

*Hacer c = alfa*c*

hasta niter = max_iter
*Hacer **w** = **w****

Random es una función que genera un número aleatorio con probabilidad uniforme en (0,1), (empleamos la terminología del compilador que usamos). En otras palabras el proceso siempre acepta movimientos o cambios *hacia abajo* pero también acepta movimientos *hacia arriba*, es decir que empeoran, con una determinada probabilidad que depende del parámetro de control *c* (a veces denominado *temperatura*). El valor de *c* va disminuyendo paulatinamente al multiplicarse por *alfa* (< 1).

Se observa como a medida que alfa aumenta el proceso se hace más lenta pero por el contrario aporta soluciones ligeramente mejores. En cualquier caso los resultados no son excesivamente satisfactorios.

4.- BÚSQUEDA TABÚ

La *Búsqueda Tabú* es una estrategia Metaheurística dada a conocer por Glover (1.989) y (1.990), y aunque la mayor parte de sus aplicaciones se han dado en problemas combinatorios o discretos, también existen aplicaciones en problemas no-lineales continuos como el que se trata en este trabajo.

Concretamente en este apartado se va a analizar el algoritmo de Búsqueda Tabú *Preliminar* propuesto por Sexton y otros, (1.998) para este problema. Sea ***w*** una solución inicial, los parámetros enteros *tamaño_lista* y *n_vecinos*, y el parámetro real *radio* el procedimiento puede ser descrito básicamente como sigue

Procedimiento Búsqueda_Tabú(***w***, *n_vecinos*, *tamaño_lista*, *radio*)
*Hacer niter = 0, Lista_Tabu = Æ, nlista = 0, **w*** = **w***
Repetir

Hacer min = ¥,

Para i = 1 hasta n_vecinos hacer

*Perturbar_Peso(**w**, radio, **w’**);*

*Si **w’** es una solución válida y E(**w’**) < min hacer: min = E(**w’**) y **w’’**= **w’***

*Si min < ¥ entonces: Actualizar Lista Tabú y hacer **w** = **w’’**;*

*Si E(**w**) < E(**w***) hacer **w*** = **w** y niter = 0*

en caso contrario hacer niter = n iter+1

hasta niter = max_iter

Una solución ***w’*** será considerada válida si: E(***w’***) < E(***w****), (entonces se dice que cumple el *Criterio de Aspiración*), o ***w’*** no coincide con ninguna de las soluciones de la Lista_Tabú. Como los valores de las soluciones son reales se establecen unos márgenes de tolerancia; es decir una solución ***w’*** es considerada igual a otra $\boldsymbol{w_0}$ de la lista si: E(***w’***) $\in$ (E($\boldsymbol{w_0}$) – *radio_tolerancia* · E($\boldsymbol{w_0}$), E($\boldsymbol{w_0}$) + *radio_tolerancia* · E($\boldsymbol{w_0}$) y además para todos los componentes $\boldsymbol{w'}_i \in$ ($\boldsymbol{w}_{0i}$ - *radio_tolerancia* · | $\boldsymbol{w}_{0i}$| , $\boldsymbol{w}_{0i}$ - *radio_tolerancia* · |$\boldsymbol{w}_{0i}$|).

La actualización de la *Lista_Tabú* consiste en las siguientes operaciones: Añadir ***w*** a la lista; si el número de elementos presentes supera el valor de *tamaño_lista* eliminar de la lista el más antiguo.

6.- ANALISIS CONJUNTO CON FUNCION DE TRANSFERENCIA SIGMOIDE

Para una mejor comparativa de los métodos propuestos en los apartados anteriores, se han hecho una serie de pruebas con todos ellos conjuntamente: *Scatter Search Dinámico*, *Variante* y *Estático*; *Mejora en Entorno de Radio Variable*, *Temple Simulado*, *Búsqueda Tabú* y *Propagación hacia atrás* (o Gradiente).

Además de los procedimientos anteriores también se ha ejecutado en este análisis una variante propia (ver Pacheco y Delgado (1.997)) del método de *Propagación hacia atrás*. Dicha variante en vez de mantener el *parámetro de aprendizaje* η constante y parar en el momento en que no haya mejora, actúa de la forma siguiente: cada vez que haya una mejora multiplicar η por 2; si en un paso no hay mejora se vuelve a la solución anterior, se divide η entre 2 y se vuelven a actualizar los pesos. El procedimiento termine cuando transcurren un determinado número de intentos, (*max_int*) sin mejora.

El problema anterior, 500 vectores de entrenamiento siguiendo una relación lineal, y la estructura de red anterior con f. de transferencia Identidad, ha servido de ‘piedra de toque’ para los algoritmos propuestos. A continuación analizaremos que resultados dan estos algoritmos para redes con función de transferencia sigmoide y diferentes relaciones entre la entrada y la salida.

Se usarán dos nodos en la capa de entrada (correspondientes a X, e Y), 5 en la capa intermedia y 1 en la de salida (correspondiente a Z).

Los procedimientos se han ejecutado con los siguientes parámetros y características:

- En los 3 procedimientos de *Scatter Search* se ha utilizado como método de mejora, en vez del procedimiento Mejorar_Peso, el procedimiento Mejora_Radio_Variable con los siguientes valores: *radio0* = 0’1, *alfa0* = 10, *n_vecinos* = 1, *max_int* = 2. Además el número de ciclos se ha limitado a 25 en todos los casos y la versión Dinámica se ha ejecutado con control a la no-convexidad, no así la estática.
- El procedimiento Mejora_Radio_Variable ha usado los siguientes valores: *radio0* = 0’1, *alfa0* = 10, *n_vecinos* = 25, *max_int* = 3.

- El procedimiento Temple_Simulado ha usado los siguientes valores: *alfa* = 0'001 y *radio* = 0'001. Además *max_iter* = 1000. Para obtener c_0 la función Temperatura_Inicial ha usado los siguientes valores: *n_vecinos*= 100, y *radio* = 0'001.
- El procedimiento Búsqueda_Tabú ha usado los siguientes valores: *n_vecinos*= 100, *tamaño_lista*= 50, y *radio* = 0'001. Además igual que antes *max_iter* = 125 y *radio_tolerancia* = 0'01%.
- Los procedimientos basados en el Gradiante (*Propagación hacia atrás* y variante antes descrita) usan como valor de η = 0'005 (en el primer caso valor fijo en el segundo variable). En la variante además *max_int* = 5.

Se usarán 100 vectores de entrenamiento. Los valores de estos vectores de entrenamiento se generan de forma aleatoria como en Sexton y otros, (1.998): X en el intervalo [-100, +100], Y en [-10,+10]. Los valores de Z se reescalan posteriormente para que estén comprendidos en el intervalo [0'1, 0'9]. Este reescalamiento se hace teniendo en cuenta el mínimo y máximo valor que pudiera alcanzar Z, dependiendo de X e Y.

A continuación se muestran los resultados para diferentes relaciones propuestas también de Sexton y otros (1.998). Estas relaciones son las siguientes:

Relación 1: $\underline{Z = X + Y}$

	SSDinamico	**SSDin-Var**	**SSEstático**	**Gradiante**	**Grad-Var**	**Radio-Variabl**	**B.Tabú**	**T.Simu-lado**
Error Mínimo	*1,9584*	*4,1968*	*3,5211*	*4,4175*	*4,9259*	*2,8194*	*1,1619*	*2,9749*
Error Medio	***6,4352***	***6,6783***	***5,7645***	***7,6274***	***8,2412***	***6,7798***	***5,4548***	***7,0062***
ErrorMáxim	*8,8711*	*9,8070*	*7,5777*	*10,6958*	*10,7013*	*9,7693*	*8,0038*	*9,7147*
T.Com Medio	17,09	15,62	31,38	18,41	0,10	9,85	1410,56	50,73

Relación 2: $\underline{Z = X \cdot Y}$

	SSDinamico	**SSDin-Var**	**SSEstático**	**Gradiante**	**Grad-Var**	**Radio-Variabl**	**B.Tabú**	**T.Simu-lado**
Error Mínimo	*9,9879*	*10,0333*	*9,8855*	*10,0707*	*10,3215*	*9,9163*	*10,0782*	*10,1687*
Error Medio	***10,3840***	***10,3703***	***10,1921***	***10,7165***	***11,0452***	***10,4135***	***10,2742***	***10,5359***
ErrorMáxim	*12,2372*	*12,1411*	*10,3511*	*12,8270*	*13,9349*	*12,7239*	*11,4561*	*13,5105*
T.Com Medio	15,05	12,01	31,16	26,82	0,12	7,43	587,55	47,06

Relación 3: $Z = \dfrac{X}{1+|Y|}$

	SSDinamico	**SSDin-Var**	**SSEstático**	**Gradiante**	**Grad-Var**	**Radio-Variabl**	**B.Tabú**	**T.Simu-lado**
Error	*2,1513*	*2,1060*	*2,1252*	*2,4584*	*2,9219*	*2,2362*	*2,2155*	*2,2156*

	SSDinamico	SSDin-Var	SSEstático	Gradiante	Grad-Var	Radio-Variabl	B.Tabú	T.Simu-lado
Mínimo								
Error Medio	***3,0719***	***2,8731***	***2,9561***	***3,7557***	***4,2833***	***3,1027***	***2,8213***	***3,0194***
ErrorMáxim	*4,7057*	*3,7288*	*5,0271*	*9,2556*	*9,3151*	*5,3598*	*3,6570*	*4,6798*
T.Com Medio	19,92	16,36	31,02	27,54	0,07	27,39	662,36	46,29

Relación 4: $\underline{Z = X^2 - Y^3}$

	SSDinamico	SSDin-Var	SSEstático	Gradiante	Grad-Var	Radio-Variabl	B.Tabú	T.Simu-lado
Error Mínimo	*2,9459*	*1,9507*	*2,3022*	*3,4619*	*3,5033*	*3,2728*	*2,8822*	*3,0664*
Error Medio	***3,8114***	***3,5995***	***3,5395***	***4,4709***	***4,9159***	***3,9417***	***3,6436***	***3,8931***
ErrorMáxim	*4,9596*	*5,7492*	*4,7513*	*7,1963*	*8,0915*	*5,0940*	*4,7331*	*4,7522*
T.Com Medio	16,51	11,45	30,66	10,95	0,03	7,81	489,24	41,41

A la vista de los resultados obtenidos en las 4 relaciones funcionales estudiadas, se pueden establecer las siguientes conclusiones:

- Los algoritmos basados en estrategias Metaheurísticas (*Scatter Search*, *Búsqueda Tabú*, *Temple Simulado*) dan soluciones sustancialmente mejores que los basados en el gradiante (*Propagación hacia atrás* y Variante), tanto en resultados medios como en mínimos.
- También el método de *Mejora con Radio Variable* da mejores resultados que la *Propagación hacia atrás*, en tiempos computación razonables.
- Entre las técnicas metaheurísticas la que da mejores soluciones ofrecen son *Scatter Search Estático* y *Búsqueda Tabú*, aunque está última usando tiempos de computación que pueden resultar excesivos.
- El resto de los metaheurísticas (las 2 versiones dinámicas de *Scatter Search* y *Temple Simulado*) dan resultados similares entre sí tanto en valor de la solución como en tiempos.
- Entre estos últimos, dependiendo de la relación funcional analizada, los mejores resulta ser la variante del *Scatter Search Dinámico*, con soluciones muy cercanas a las conseguidas por *Búsqueda Tabú* (mejores en la última relación funcional) y con un tiempo de computación muy inferior. La incorporación a los algoritmos de *Scatter Search* propuestos de un método de diversificación podría hacerlos aún más eficaces.

8.- REFLEXIONES

En este informe se ha analizado un problema concreto: dada una determinada estructura de red neuronal, (con un número determinado de nodos en cada capa, y con una determinada función (-es) de transferencia), y dados unos vectores de entrenamiento determinar el vector de pesos de dichas red que minimice el error cuadrático medio total. No se han analizado otros aspectos que pueden incidir en el funcionamiento de la red: Número de capas y nodos en cada capa, existencia de nodos independientes, elección de vectores de entrenamiento, escalado de estos, o tipo de f. de transferencia elegido.

Los métodos tradicionales basados en el gradiente han demostrado tener limitaciones en su funcionamiento. La primera de ella es la convergencia a mínimos locales no-globales. Frente a estos métodos, el resto de los algoritmos propuestos en este trabajo han dado por lo general mejores resultados con tiempos de computación razonables (exceptuando *Busqueda Tabú*).

Estos últimos métodos analizados (*Metaheurísticos* y *Mejora con Radio Variable*) se basan en (o hacen uso de) la generación de soluciones vecinas aleatoriamente en un entorno de cada solución. Este procedimiento, conceptualmente sencillo y fácil del implementar, permite que estos métodos puedan usarse con cualquier tipo de función de transferencia. Por el contrario los métodos basados en el gradiente, obviamente, solo se pueden usar con funciones derivables.

9.- BIBLIOGRAFÍA

GLOVER, F (1.989): “Tabu Search: Part I.” *ORSA Journal on Computing*. Vol. 1, pp. 190-206.

GLOVER,F (1.990). “Tabu Search: Part II.” *ORSA Journal on Computing*. Vol. 2, pp. 4-32.

GLOVER,F., LAGUNA,M. and MARTÍ,R. (1.999): "Scatter Search". Aparecerá en *Theory and Applications of Evolutionary Computation: Recent Trends*, A.Ghosh and S. Tsutsui (Eds.), Springer-Verlag.

KELLY,J.P., RANGASWAMY,B., and XU,J. (1.996): “A Scatter Search-Based Learning Algorithm for Neural Networks Training”. *Journal of Heuristics*, vol.2, n.2, 129-146.

KIRPATRICK S., GELATT C. D. and VECCHI M. P. (1.982): "Optimization by Simulated Annealing". *IBM Research Report* RC 9355.

KIRPATRICK S., GELATT C. D. and VECCHI M. P. (1.983): "Optimization by Simulated Annealing". *Science*, Vol. 220, pp 671-680.

LAGUNA, M. (1.999): "Scatter Search". Aparecerá en *Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende (eds). Oxford Academic Press.

LECUN,Y. (1.986): “Learning Process in an Asymetric Threshold Network. Disordered Systems and Biological Organization”. *Springer*, Berlin, pg. 233-240.

PARKER,D. (1.985): “Learning logic”. Technical Report TR-87. *Center for Computational Research in Economics and Management Science*, MIT, Cambridge. MA.

RUMELHART,D., HINTON,G. and WILLIAMS,R. (1.986): “Learning internal representations by error propagation”. In RUMELHART,D., McCLEELAND,J. (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Vol.1. MIT Press. Cambridge. MA.

WERBOS,P. (1.974): *Beyond Regression: New tools for prediction and analisys in the behavioral sciences*. Tesis Doctoral. Harvard. Cambridge.