UTILIZACIÓN DE UN SISTEMA BASADO EN REGLAS DIFUSAS PARA LA APLICACIÓN DE OPERADORES EN UN ALGORITMO COOPERATIVO-COMPETITIVO

M. Dolores Pérez Godoy A. Jesús Rivera Rivas M. José del Jesus Díaz F. José Berlanga Rivera

Departamento de Informática, Universidad de Jaén, {lperez, arivera, mjjesus, berlanga}@ujaen.es

Resumen

En este trabajo se presenta CO^2RBFN, un algoritmo bioinspirado para el diseño de redes neuronales, concretamente Redes de Funciones de Base Radial (RBFNs). El método de aprendizaje está basado en la programación evolutiva con un enfoque cooperativo-competitivo en el que cada individuo representa una neurona y la población al completo la red. Destaca en el mismo la utilización de un Sistema Basado en Reglas Difusas (SBRD) que representa conocimiento experto sobre los operadores a aplicar a una neurona en base a sus características. El algoritmo se ha aplicado a la resolución de problemas de clasificación.

Palabras Clave:RBFNs, SBRDs, Clasificación, Cooperativo-Competitivo.

1 INTRODUCCIÓN

Las Redes de Funciones de Base Radial (RBFNs) constituyen uno de los paradigmas más populares dentro del campo de las Redes Neuronales. Este modelo de cómputo ha demostrado su solvencia a la hora de abordar problemas como aproximación de funciones [7], clasificación [4] o predicción de series temporales [20]. Las Funciones de Base Radial (RBFs) se usaron inicialmente en interpolación numérica y aproximación funcional [17]. A finales de los 80 tienen lugar las primeras investigaciones de redes neuronales basadas en RBFs [3][11].

Son muchas las características que despiertan el interés por este tipo de redes entre las cuales cabe destacar: su topología simple con tan solo una capa oculta, su capacidad de ser un aproximador universal [13] o la analogía entre la salida de estas redes y los campos receptivos localizados, encontrados en estructuras biológicas cerebrales.

Básicamente una salida de una RBFN ofrece una suma ponderada, por unos pesos, de las respuestas de cada una de las neuronas/RBFs que conforman la red. Una salida o respuesta de una RBF tiene un carácter local y viene dada en función de un centro y un radio. Así pues, el objetivo a la hora de diseñar una RBFN es determinar el centro y el radio que caracteriza cada RBF así como su peso para cada salida de la red.

Las técnicas empleadas en el diseño de RBFNs son muy diversas. El algoritmo típico de diseño de RBFNs tiene dos etapas. En la primera etapa se determinan los centros y los radios de las RBFs, mientras que en la segunda etapa se calculan sus pesos. Para determinar los centros y los radios se pueden emplear técnicas de clustering [14]. En la segunda etapa y para calcular los pesos se pueden utilizar algoritmos como el LMS [21], SVD [6], etc.

El diseño de una RBFN también se puede abordar desde el punto de vista de la computación evolutiva [1]. En la mayoría de estas propuestas evolutivas, un individuo representa una RBFN completa. De esta manera, los operadores evolutivos actúan sobre los individuos añadiendo RBFs, eliminándolas o modificándolas.

Sin embargo y según el trabajo de Potter [16] la computación evolutiva tradicional presenta ciertos problemas relacionados con la evaluación de subcomponentes independientes en el diseño de ciertos modelos. Así se propone la Coevolución Cooperativa [16] para extender el modelo evolutivo tradicional y conseguir un entorno de diseño en el que los individuos de la población representen una parte de la solución evolucionando en paralelo. Ahora los individuos no sólo compiten por sobrevivir sino que deben cooperar para alcanzar una solución.

Los autores ya han desarrollado un importante trabajo en diseño híbrido de RBFNs para la aproximación de funciones y predicción de series temporales [18].

Manteniendo el enfoque coopertivo-competitivo pero cambiando operadores, evaluación de individuos y esquema evolutivo, actualmente trabajamos en el desarrollo de una nueva propuesta para el diseño de RBFNs orientadas a resolver problemas de clasificación [15]. En el trabajo actual se propone un marco de evolución cooperativo-competitivo donde además se utiliza un sistema basado en reglas difusas para decidir la aplicación de los operadores.

La organización del resto de este trabajo se describe a continuación. En la sección 2 se introducen las RBFNs y se explica cómo se aplican a problemas de clasificación de patrones. En la sección 3 se presenta el algoritmo propuesto y sus resultados se muestran en la sección 4. Por último se explican las conclusiones alcanzadas y las líneas de trabajo futuro en la sección 5.

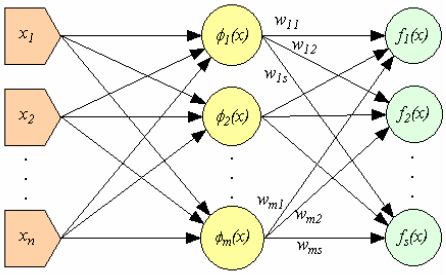

Figura 1: Topología de una RBFN

2 REDES DE FUNCIONES DE BASE RADIAL Y SU APLICACIÓN A PROBLEMAS DE CLASIFICACIÓN

Una Red de Funciones de Base Radial es un tipo de red neuronal hacia delante con tres capas: la capa de entrada con *n* nodos, una capa oculta con *m* neuronas o RBFs, y una capa de salida con uno o varios nodos, ver Figura 1. Las *m* neuronas/RBFs de la capa oculta ofrecen una activación simétrica radial $\phi_i: R^n \rightarrow R$, que puede tomar diferentes formas aunque la más común es la función gausiana que viene dada por la expresión: $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c}_i\|/d_i)^2})$, donde $\vec{c}_i \in R^n$ es el centro de la función base ϕ_i, $d_i \in R$ es el radio y como $\|\ \|$ suele utilizarse la norma euclídea en R^n. Los nodos de salida implementan la siguiente ecuación:

$$f_j(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \qquad (1)$$

En un escenario de clasificación la RBFN debe establecer una relación entre un espacio de entrada X^n hacia un conjunto finito de clases *Y* con *k* clases. De esta manera un típico conjunto de entrenamiento *S* sería:

$$S = \{(\vec{x}_u, y_u) \mid x_u \in X^n,\ y \in Y,\quad u = 1,..., p\} \qquad (2)$$

donde $\vec{x}_u$ es el vector de características e y_u es la clase a la que pertenece. Normalmente, en un problema de clasificación, el número de salidas de la RBFN se corresponde con el número de clases (*k*). Para entrenar la RBFN la pertenencia a la y_u se codifica en un vector binario $\vec{z}_u \in \{0,1\}^k$ a través de la relación $\vec{z}_u^i = 1$ sii $y_u = i$, y $\vec{z}_u^i = 0$ *en otro caso*. La clase obtenida por la red, ante una determinada entrada, será la salida de la red con un mayor grado de activación.

En la bibliografía especializada se pueden encontrar distintos métodos evolutivos [4][9][12] para el diseño de RBFNs orientadas al problema de clasificación. Sin embargo, la mayoría de las aproximaciones existentes trabajan con métodos evolutivos típicos donde un individuo representa una red completa, por lo que pueden tener problemas de alto costo computacional y convergencia prematura a mínimos locales. Como se ha comentado anteriormente estos problemas pueden mitigarse utilizando técnicas cooperativas-competitivas donde un individuo representa una neurona o RBF tal y como se propone en este trabajo. Hasta ahora en la bibliografía no son muchos los trabajos [15][18][19][20] que implementan este tipo de paradigma, debido sobre todo a la dificultad que entraña definir los parámetros de cooperación y competición entre los individuos.

3 CO^{2}RBFN: UN ALGORITMO HÍBRIDO COOPERATIVO-COMPETITIVO PARA EL DISEÑO DE RBFNs

Se propone un enfoque cooperativo-competitivo híbrido dentro de un marco evolutivo, para la resolución de problemas de clasificación. En esta propuesta cada individuo de la población representa una función base y la población entera es la responsable de la solución final. Se tiene un entorno en el que los individuos cooperan para alcanzar la solución definitiva, no obstante, también compiten por la supervivencia, dado que si el trabajo de un individuo no es bueno dicho individuo será eliminado.

Este escenario tiene en cuenta la respuesta local de las neuronas y la interpretabilidad de esta clase de redes, lo cual establece una importante guía de diseño. Para medir la asignación de crédito de un individuo, se proponen tres factores que evalúan el papel de cada RBF en la red. El algoritmo usa un Sistema Basado en Reglas Difusas para decidir la probabilidad de aplicación de los operadores a las RBFs.

El algoritmo puede trabajar tanto con atributos lineales (reales y enteros) como nominales. La distancia utilizada es la distancia HDVM [22]:

$$HVDM(x, y) = \sqrt{\sum_{a=1}^{n} d_a^2(x_a, y_a)} \qquad (3)$$

donde *n* es el número de atributos. La función $d_a^2(x,y)$ devuelve la distancia entre los valores *x* e *y* del atributo *a* y se define como:

$$d_a(x,y) = \begin{cases} 1 \quad si\ x\ o\ y\ son\ \ desconocidos \\ VDM_a(x,y) \quad si\ a\ \ es\ \ \text{nominal} \\ \text{distancia euclídea} \quad si\ \ a\ \ es\ \ lineal \end{cases} \tag{4}$$

Las etapas principales de CO^{2}RBFN se explican en las siguientes subsecciones. A continuación se muestra su pseudocódigo:

```
1. Inicialización de las RBFs
2. Entrenamiento de las RBFs
3. Evaluación de las RBFs
4. Aplicación de los operadores
   a las RBFs
5. Sustitución de las RBFs que
   han sido eliminadas
6. Selección de las mejores
   RBFs
7. Salto al paso 2 si no se
   verifica la condición de
   Parada
```

Algoritmo 1. Principales etapas de CO^{2}RBFN

3.1. INICIALIZACIÓN DE LA RED

El proceso para definir la red inicial es simple. Las neuronas se colocan aleatoriamente intentado cubrir las diferentes clases de los patrones del conjunto de entrenamiento. El número de RBFs está especificado como un parámetro (es el tamaño de la población, m).
El centro de cada RBF, $\vec{c}_i$, se inicializa a partir de un patrón del conjunto de entrenamiento, teniendo en cuenta que las RBFs se distribuyan equitativamente entre las diferentes clases existentes. El radio, d_i, se inicializa a la mitad de la media de las distancias entre los centros. Finalmente los pesos, w_{ij} , se ponen a cero.

3.2. ENTRENAMIENTO DE LAS RBFs

Durante esta etapa se entrenan los pesos de las RBFs. El propósito de entrenar los pesos es el de explotar la información local extraída del comportamiento de las RBFs. Dicho entrenamiento se realiza mediante la técnica LMS [21].

3.3. EVALUACIÓN DE LAS RBFs

Se requiere un mecanismo de asignación de crédito para poder evaluar el papel de cada función base dentro del entorno cooperativo-competitivo. Para este propósito se consideran tres parámetros, a_i, e_i, o_i para cada RBF ϕ_i.
La contribución, a_i, de la RBF ϕ_i, $i=1\ldots m$, a la salida de la RBFN, se determina considerando su peso, w_i, y el número de patrones de entrenamiento dentro de su radio, pi_i . Se penaliza una RBF con poco peso y pocos patrones dentro de su radio:

$$a_i = \begin{cases} |w_i| & if\ \ pi_i > q \\ |w_i| * (pi_i / q) & en\ otro\ caso \end{cases} \tag{5}$$

donde q es la media de los valores pi_i menos su desviación típica.
La medida de error, e_i, para cada RBF ϕ_i, se obtiene contando los patrones que estando dentro de su radio están mal clasificados:

$$e_i = \frac{pibc_i}{pi_i} \tag{6}$$

donde $pibc_i$ y pi_i son el número de patrones mal clasificados y el número total de patrones, dentro de su radio, respectivamente.
El solapamiento de la RBF, ϕ_i, con otras RBFs se cuantifica usando el parámetro o_i. Este parámetro se calcula partiendo de la base de la metodología *fitness sharing* [5], que intenta mantener la diversidad en la población. El factor se expresa como:

$$o_i = \sum_{j=1}^{m} o_{ij} \tag{7}$$

donde o_{ij} mide el solapamiento entre las RBF ϕ_i y ϕ_j, $j=1\ldots m$.

3.4. APLICACIÓN DE LOS OPERADORES A LAS RBFs

En el algoritmo existen cuatro operadores que se pueden aplicar a las RBFs:

- Operador Elimina: es un operador que elimina una RBF.
- Operador Mutación Aleatoria: es un operador que modifica el centro y el radio de la RBF aleatoriamente. El radio se modifica con una probabilidad inversamente proporcional al número de características en el problema de clasificación (n). La modificación, en caso de llevarse a cabo, puede oscilar en un porcentaje entre un 5% y un 50% sobre el radio antiguo. En cuanto a la modificación del centro, el número de coordenadas que se van a mutar se obtiene aleatoriamente entre un 1% y un 25 % del total de características del conjunto de entrenamiento. Si la coordenada a mutar tiene un valor real se le provoca un incremento o decremento aleatorio de un valor entre un 5% y un 50% del radio. Por el contrario, si tiene un valor nominal se produce un cambio a uno de los valores nominales posibles, con una probabilidad inversamente proporcional a la distancia entre el valor actual y los posibles valores de cambio.
- Operador Mutación Informada: este operador también puede modificar el radio y algunas coordenadas del centro de la RBF usando información de su entorno. El centro se modifica como sigue:

$$c'_{ij} = c_{ij} \pm h \quad \forall j = 1\ldots n \tag{8}$$

El incremento o decremento del centro antiguo se decide mediante un número aleatorio h ($0.05 \cdot di \leq h \leq 0.5 \cdot di$). El centro se modifica para aproximar la neurona al centro de los patrones de entrenamiento que están dentro del radio de la RBF y pertenecen a su misma clase.

El objetivo de modificar el radio es que el mayor número posible de patrones que pertenecen a la misma clase que la RBF, queden dentro de su radio. De esta forma el radio se modifica de la siguiente forma:

$$\begin{cases} d' = d + h & si\ u \leq D + A \\ d' = d - h & en\ otro\ caso \end{cases} \quad (9)$$

donde h, es un número aleatorio $(0.05 \cdot d_i \leq h \leq 0.5 \cdot d_i)$, D es el número de patrones que no pertenecen a la clase de la RBF dividido por el número de patrones que pertenecen a la clase, ambas cantidades medidas dentro de su radio. A, es el número de patrones que pertenecen a la misma clase que la RBF dividido por el número de patrones que no pertenecen a la misma clase, ambas cantidades medidas dentro de dos veces el radio de la RBF.

- Operador Nulo: no se realiza ninguna acción sobre la RBF.

Estos operadores se aplican a la población entera de RBFs. La probabilidad de elegir un operador para una RBF dada viene determinada por un sistema difuso tipo Mamdani [10], cuyas entradas son los parámetos a_i, e_i y o_i, éstos determinan la asignación de crédito de cada RBF. Estas entradas se consideran como variables lingüísticas va_i, ve_i y vo_i , y las salidas son $p_{elimina}$, p_{ma}, p_{mi} y p_n, que representan la probabilidad de aplicar los operadores Elimina, Mutación Aleatoria, Mutación Informada y Nulo, respectivamente. La Figura 1 muestra las funciones de pertenencia para las etiquetas lingüísticas de las entradas y las salidas respectivamente. El número de etiquetas lingüísticas se ha determinado empíricamente, con centros y bases directamente relacionados con su significado. La Tabla 1 muestra las reglas usadas para relacionar los antecedentes y los consecuentes. El bajo número de reglas permite que el sistema difuso diseñado sea muy simple.

En el diseño de reglas se tiene en cuenta el hecho de que una RBF es peor si su contribución (a_i) es baja, su error (e_i) es alto y su solapamiento (o_i) también es alto. En el lado opuesto, una RBF es mejor cuando su contribución es alta y su error y solapamiento son bajos. De esta forma, la probabilidad de eliminar una RBF se incrementa cuando ésta tiene un mal comportamiento; la probabilidad de mutar va aumentando a medida que el comportamiento es mejor. El operador nulo se aplicará cuando el comportamiento de una RBF sea muy bueno.

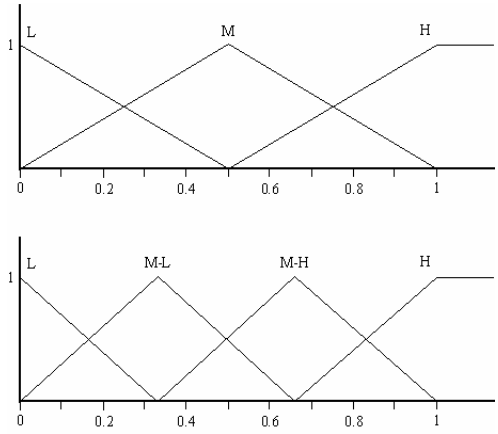

Figura 1. Arriba: etiquetas lingüísticas para las variables de entrada. Abajo: etiquetas lingüísticas de las variables de salida.

Tabla 1: Base de reglas utilizada.

	Antecedentes			Consecuentes			
	v_a	v_e	v_o	$P_{elimina}$	P_{mr}	P_{mi}	p_{nulo}
R1	L			M-H	M-H	L	L
R2	M			M-L	M-H	M-L	M-L
R3	H			L	M-H	M-H	M-H
R4		L		L	M-H	M-H	M-H
R5		M		M-L	M-H	M-L	M-L
R6		H		M-H	M-H	L	L
R7			L	L	M-H	M-H	M-H
R8			M	M-L	M-H	M-L	M-L
R9			H	M-H	M-H	L	L

3.5. INTRODUCCIÓN DE NUEVAS RBFs

En este punto el algoritmo sustituye las RBFs eliminadas por otras nuevas. Existen dos alternativas para situar las nuevas RBFs, ambas con probabilidad de 0.5: en un patrón mal clasificado que esté fuera de cualquier RBF o en un patrón obtenido de forma aleatoria.

3.6. SELECCIÓN DE LAS MEJORES RBFNs

Después de aplicar los operadores de mutación aparecen nuevas RBFs y éstas son comparadas con sus padres para determinar cuáles tienen un mejor comportamiento en la red. Las mejores RBFs serán las elegidas para formar parte de la nueva población.

4 RESULTADOS EXPERIMENTALES

Para la experimentación se han utilizado cuatro bases de datos obtenidas de UCI Repository of Machine Learning Database: Hepatitis, Sonar, Wbcd y Wine.

La Tabla 2 muestra las características de las bases de datos.

El estudio se ha realizado usando ten-fold cross validación, es decir, diez particiones para entrenamiento y test, el 90% de los datos para entrenamiento y el 10% para test. Para cada base de datos se obtiene la media de las diez particiones. El número de generaciones se fija a 200.

Tabla 2: Características de las bases de datos

Bases de Datos	*#Instancias*	*#Atributos Lineales*	*#Atributos Nominales*	*#Clases*
Hepatitis	155	6	13	2
Sonar	208	60	0	2
Wbcd	699	9	0	2
Wine	178	13	0	3

Las tablas de la 3 a la 6 muestran los resultados obtenidos con CO^{2}RBFN y con otros métodos. Los otros métodos con los que se compara pertenecen a distintos paradigmas dentro del campo del aprendizaje automático:

- C4.5: algoritmo que genera reglas de clasificación en forma de árboles de decisión a partir de un conjunto de ejemplos dado. Implementación obtenida de la herramienta Keel [8].
- GP-COACH: propuesta evolutiva que utiliza la Programación Genética para aprender SBRDs compactos y precisos en problemas con alta dimensionalidad y que sigue el enfoque cooperativo-competitivo de representación de reglas [2]. Implementación propia.
- GenéticoRBFN: algoritmo genético para el diseño de RBFNs basado en el esquema Pittsburgh. En dicho esquema un individuo es una red completa. Implementación propia.
- MultiObjetivoRBFN: algoritmo para el diseño de RBFNs con enfoque multiobjetivo, basado en el algoritmo NSGA2 (un individuo representa una red). Implementación propia.
- Perceptrón Back: algoritmo para el diseño de Redes Perceptrón Multicapa que utiliza el algoritmo Backpropagation para el aprendizaje. Implementación obtenida de la herramienta Keel [8].
- Perceptrón Grad: algoritmo para el diseño de Redes Perceptrón Multicapa que utiliza el algoritmo de Gradiente Conjugado para el aprendizaje. Implementación obtenida de la herramienta Keel [8].

El análisis de los resultados muestra que CO^{2}RBFN obtiene RBFNs con una estructura simple y los resultados son comparables, incluso superiores, a los obtenidos con otros métodos. La simplicidad de la red obtenida implica una buena interpretabilidad, lo cual es una característica importante en los problemas de clasificación.

Tabla 3: Resultados obtenidos con Hepatitis

Algoritmo	*#nodos/ reglas*	*Test (%)*
C4.5	7.1	89.647
GP-COACH	9.033	80.539
GenéticoRBFN	7.5	86.711
MultiObjetivoRNFN	8	87.479
Perceptrón Back.	30	71.817
Perceptrón Grad.	10	79.461
CO^{2}RBFN	8	**87.399**

Tabla 4: Resultados obtenidos con Sonar

Algoritmo	*#nodos/ reglas*	*Test (%)*
C4.5	14.3	71.071
GP-COACH	13.767	65.651
GenéticoRBFN	7.68	73.305
MultiObjetivoRNFN	6	73.814
Perceptrón Back.	30	67.762
Perceptrón Grad.	10	73.524
CO^{2}RBFN	8	**75.086**

Tabla 5: Resultados obtenidos con Wbcd

Algoritmo	*#nodos/ reglas*	*Test (%)*
C4.5	12.4	94.995
GP-COACH	8.433	94.896
GenéticoRBFN	6.24	96.713
MultiObjetivoRNFN	8	96.969
Perceptrón Back.	30	87.722
Perceptrón Grad.	10	94.985
CO^{2}RBFN	5	**97.083**

Tabla 6: Resultados obtenidos con Wine

Algoritmo	*#nodos/ reglas*	*Test (%)*
C4.5	5.1	94.902
GP-COACH	7.767	93.584
GenéticoRBFN	10.44	95.275
MultiObjetivoRNFN	12	95.643
Perceptrón Back.	30	93.301
Perceptrón Grad.	10	96.111
CO^{2}RBFN	7	**96.739**

5 CONCLUSIONES

Este trabajo presenta un algoritmo bioinspirado para diseñar redes de funciones de base radial. En él se mantiene una población de RBFs que cooperan para lograr una solución final y que compiten por su supervivencia. El comportamiento (asignación de crédito) de cada función base dentro de la red completa se mide en base a tres factores: la contribución, a_i, de la RBF a la salida de la red, el error, e_i, de la RBF y el solapamiento de la RBF con otras RBFs.

Se utilizan cuatro operadores para poder aplicar a una RBF dada en el proceso evolutivo: Elimina, Mutación Aleatoria, Mutación Informada y Nulo. Con las dos modalidades de mutación se consigue un equilibrio adecuado entre las cualidades de explotación y exploración que todo algoritmo evolutivo debe poseer. Por un lado la mutación informada utiliza información del entorno de una neurona para modificar ésta de forma que se adapte óptimamente a su entorno. La mutación aleatoria promueve modificaciones de forma que se favorezca la exploración del entorno y se huya de óptimos locales.

La aplicación de dichos operadores es determinada mediante un Sistema basado en Reglas Difusas. Las entradas a dicho sistema son los parámetros a_i, e_i, y o_i, usados para medir la asignación de crédito y las salidas son las probabilidades de aplicación de los distintos operadores.

El algoritmo propuesto se ha evaluado con bases de datos conocidas y los resultados obtenidos son comparables e incluso superiores a los obtenidos con otros métodos.
Como líneas de trabajo futuro se estudiará la aplicación del algoritmo a bases de datos no balanceadas.

Agradecimientos

Este trabajo ha sido subvencionado por los proyectos españoles TIN2005-04386-C05-03 y TIN2007-60587.

Referencias

[1] Bäck, T.; Hammel, U.; Schwefel, H. Evolutionary computation: comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, v.1 n.1 April. Pág. 3-17, 1997.

[2] Berlanga, F.J. Del Jesus, M.J.; Herrera, F. A Novel Genetic Cooperative-Competitive Fuzzy Rule Based Learning Method using Genetic Programming for High Dimensional Problems. *3rd International Workshop on Genetic and Evolving Fuzzy Systems (GEFS08)*. WittenBommerholz (Germany) Pág. 101-106, 2008.

[3] Broomhead, D.; Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex System*. v 2. Pág. 321-355, 1988.

[4] Buchtala, O.; Klimek, M.; Sick, B. Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol 35, n. 5, Oct. Pág. 928-947, 2005.

[5] Goldberg, D.; Richardson J. Genetic algorithms with sharing for multimodal function optimization. In Grefenstette (ed.), *Proc. of the 2nd International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates., Pág. 41-49, 1987.

[6] Golub, G.; Van Loan, C. Matrix computations. J. *Hopkins University Press*, 3rd ed., 1996.

[7] González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, J., Fco, A. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, Volume 14, Issue 6, Pág. 1478-1495, 2003.

[8] KEEL: Knowledge Extraction based on Evolutionary Learning (Spanish National Project TIC2002-04036-C05 and TIN2005-08386-C05). http://www.keel.es.

[9] Lacerda, E; Carvalho, A.; Braga A.; Ludermir T. Evolutionary Radial Functions for Credit Assessment. *Applied Intelligence* 22. Springer Netherlands. Pág. 167-181, 2005.

[10] Mandani, E.; Assilian, S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Machine Stud.*, v. 7, n. 1, Pág. 1-13, 1975.

[11] Moody, J.; Darken, C, J. Fast learning in networks of locally-tuned processing units. *Neural Computing*. vol 1. Pág. 281-294, 1989.

[12] Neruda, R.; Kudová, P. Learning methods for radial basis function networks. *Future Generation Computer Systems*, Volume 21, Issue 7, pp 1131-1142, 2005.

[13] Park, J.; Sandberg, I. Universal approximation using radial-basis function networks. *Neural Comput.*, vol. 3, Pág. 246-257, 1991.

[14] Pedrycz, W. Conditional fuzzy clustering in the design of radial basis function neural networks. *IEEE Transactions on Neural Networks* 9 (4), Pág. 601-612, 1998.

[15] Pérez-Godoy, M; Rivera, A.J.; Jesus, M.J.; Rojas, I. CoEvRBFN: an approach to solving the classification problem with a hybrid cooperative-coevolutive algorithm. *9th International Work-Conference on Artificial Neural Networks (IWANN07)*, LNCS 4507, Pág. 324-332, 2007.

[16] Potter, M.; De Jong, K. Cooperative Coevolution: an architecture for evolving coadapted subcomponents". *Evolutionary Computation*, 8(1), Pág. 1-29, 2000.

[17] Powell, M. Radial basis functions for multivariable interpolation: A review. In *IMA. Conf. on Algorithms for the approximation of functions and data*, Pág. 143-167, 1985.

[18] Rivera, A.J.; Rojas, I.; Ortega, J.; del Jesús, M.J.; A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. *Soft Computing*. ISSN 1432-7643. D.O.I: http://dx.doi.org/10.1007/s00500-006-0128-9, 2006.

[19] Topchy A.; Lebedko, O.; Miagkikh V.; Kasabov N.; Adaptive training of radial basis function networks based on co-operative evolution and evolutionary programming. *Prog. in connectionist-based information syst*. N. Kasabov et al (eds), Springer, Pág. 253-258, 1998.

[20] Whitehead, B; Choate, T. Cooperative-competitive genetic evolution of Radial Basis Function centers and widths for time series prediction. *IEEE Trans. on Neural Networks*, Vol.7, No.4, July, Pág.869-880, 1996.

[21] Widrow, B.; Lehr, M.A. 30 Years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE*, v. 78 n. 9, sept. Pág. 1415-1442, 1990.

[22] Wilson, D. Randall, and Tony R. Martinez. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, vol. 6, no. 1, Pág. 1-34, 1997.