Optimización de CoEvRBF para aumentar su eficiencia en tareas de clasificación

M. Dolores Pérez-Godoy[1] Antonio J. Rivera[1] M. José del Jesus[1] Ignacio Rojas[2]
[1] Departamento de Informática
Universidad de Jaén
{lperez, arivera, mjjesus}@ujaen.es
[2] Depto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
irojas@atc.ugr.es

Resumen

Este trabajo describe una optimización del funcionamiento de CoEvRBFN, que persigue aumentar la eficiencia del algoritmo tanto en su fase de diseño de redes, como en la obtención de los resultados. Esta optimización se consigue con cambios en dos fases principales del algoritmo de aprendizaje, como son la fase de entrenamiento de la red y la de aplicación de operadores sobre los individuos de la población. Así, la fase de entrenamiento de la red se ha simplificado considerablemente ya que se ha pasado de entrenar pesos, radios y centros a entrenar sólo pesos con el tradicional algoritmo LMS. En cuanto a los operadores se ha eliminado el operador NULL que existía previamente y se ha introducido un nuevo operador de mutación que tiene en cuenta el trabajo que cada individuo realiza. Tras aplicar el nuevo algoritmo, a típicos problemas de clasificación, se han obtenido mejores resultados que con la versión anterior.

1. Introducción

Las Redes de Funciones de Base Radial (RBFNs) es 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 [23]. Las Funciones de Base Radial (RBFs) se usaron inicialmente en interpolación numérica y aproximación funcional [18]. A finales de los 80 tienen lugar las primeras investigaciones de redes neuronales basadas en RBFs [3][10].

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 [14] 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 radios. Para determinar los centros y los radios se pueden emplear técnicas de clustering [15]. En la segunda etapa y para calcular los pesos se pueden utilizar algoritmos como el LMS [24], 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 [17] 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 [17] 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 [19]. En este trabajo se propone un marco de coevolución cooperativa donde además se utiliza un sistema basado en reglas difusas para decidir la aplicación de los operadores y un algoritmo de minimización local para afinar las soluciones. Posteriormente se ha realizado una importante adaptación del algoritmo para aplicarlo a tareas de clasificación [16]. En el trabajo actual se presenta una optimización de ese algoritmo, simplificando la parte de entrenamiento del mismo e insertando un nuevo operador que muta los individuos a partir de la información del trabajo que éstos realizan.

La organización del resto de este trabajo se describe a continuación. En la sección II se introducen las RBFNs y se explica cómo se aplican al problema de clasificación de patrones. En la sección III 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 V.

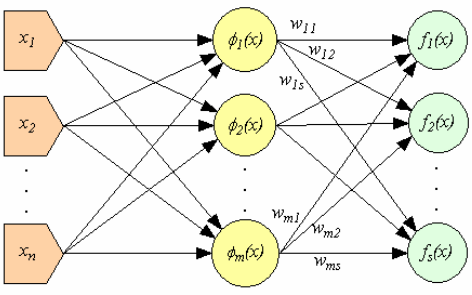

Figura 1. Topología de una RBFN

2. Redes de Funciones de Base Radial y su aplicación al problema 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:

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

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 ofrecida 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][8][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 suelen tener problemas de alto costo computacional y convergencia prematura a mínimos locales. Como se ha comentado anteriormente estos problemas pueden mitigarse utilizando técnicas coevolutivas-cooperativas 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 [19][21][23] que implementan este tipo de paradigma, debido sobre

todo a la dificultad que entraña definir los parámetros de coevolución y cooperación entre los individuos.

3. CoEvRBFN: un algoritmo híbrido coevolutivo para el diseño de RBFNs

Se propone una aproximación coevolutiva híbrida 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 hay una competición por la supervivencia, dado que si el trabajo de un individuo no es bueno dicho individuo se eliminará.

Este escenario de coevolución tiene en cuenta la respuesta local de las neuronas y la interpretabilidad de esta clase de redes, de forma que se establece una importante guía de diseño en nuestro algoritmo. Para medir la asignación de crédito de un individuo, se proponen tres factores para evaluar el papel de una RBF en la red. El algoritmo usa un Sistema Basado en Reglas Difusas para decidir la probabilidad de aplicación de los operadores a ciertas RBFs. Las etapas principales de CoEvRBFN, se explican en la siguientes subsecciones, a continuación se muestra su pseudocódigo:

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

Algoritmo 1. Principales etapas de CoEvRBFN

3.1. Inicialización de la Red

Para definir la red inicial se usa un proceso 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 la RBFN

Durante esta etapa se entrenan los pesos de las RBFs mediante la técnica LMS [24]. A diferencia del trabajo anterior [16], en esta propuesta no se entrenan los centros y los radios de las RBFs de tal forma que se consigue mayor eficiencia en cuanto al tiempo de ejecución del algoritmo.

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 evolutivo. 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, $i=1...m$, 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:

$$a_i = \begin{cases} |w_i| & if\ pi_i > q \\ |w_i| * (pi_i / q) & en\ otro\ caso \end{cases} \quad (3)$$

donde q es la media de los valores pi_i dividida por cuatro.
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.:

$$e_i = \frac{pibc_i}{pi_i} \quad (4)$$

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:

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

$$o_{ij} = \begin{cases} \left(1 - \left\|\phi_i - \phi_j\right\| / d_i\right) & if \left\|\phi_i - \phi_j\right\| < d_i \\ 0 & en\ otro\ caso \end{cases} \quad (5)$$

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

3.4. Aplicación de los operadores a las RBFs

En este trabajo se proponen tres operadores que se pueden aplicar a las RBFs. Se introduce un nuevo operador de mutación con información con respecto al trabajo anterior.

- Operador ELIMINA: es un operador que elimina una RBF.
- Operador MUTA SIN INFORMACIÓN: 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 20% sobre el radio antiguo. En cuanto a la modificación del centro, se alteran algunas de sus coordenadas en la misma proporción que en la mutación del radio. El número de coordenadas del centro que se van a mutar se obtiene aleatoriamente entre un 1% y un 25 % del total de características del conjunto de entrenamiento.
- Operador MUTA CON INFORMACIÓN: este operador también puede modificar el radio y centro de la RBF. El centro se modifica como sigue:

$$c_{ij}^{'} = c_{ij} \pm h \quad \forall j = 1...n \quad (6)$$

El incremento o decremento del centro antiguo se decide mediante un número aleatorio h $(0.05 \cdot d_i \le h \le 0.2 \cdot d_i)$. 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:

$$\begin{cases} d^{'} = d + h & si\ (mdpco \le 2 * d)\ y\ (npco > 0) \\ d^{'} = d - h & si\ npci * 0.1 \le npnci \end{cases} \quad (7)$$

donde h, es un número aleatorio $(0.05 \cdot d_i \le h \le 0.2 \cdot d_i)$, $npco$ y $npci$, son el número de patrones que pertenecen a la misma clase que la RBF y están fuera y dentro de su radio respectivamente; $npnci$ determina el número de patrones que no pertenecen a la clase de la RBF y sí están dentro de su radio; y $mdpco$ es la distancia mínima entre el centro de la RBF y los patrones que pertenecen a su misma clase pero están fuera de su radio.

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 [9], 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_{mSI} y p_{mCI}, que representan la probabilidad de aplicar los operadores ELIMINA, MUTA SIN INFORMACIÓN y MUTA CON INFORMACIÓN respectivamente. La Figura 2 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 se diseñe un sistema fuzzy 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.

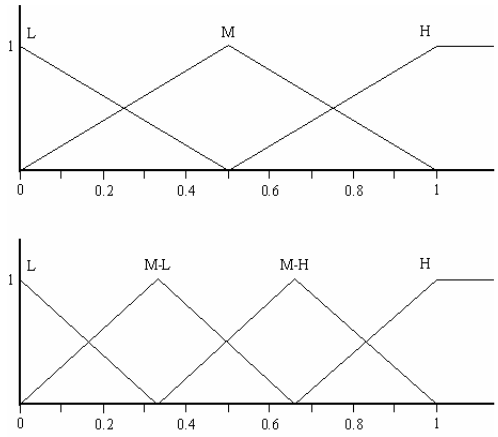

Figura 2. Arriba: etiquetas lingüísticas para las variables de entrada. Abajo: etiquetas lingüísticas de las variables de salida.

Tabla 1. Base de reglas utilizadas.

	Antecedentes			Consecuentes		
	v_a	v_e	v_o	$p_{elimina}$	p_{mSI}	p_{mCI}
R1	L			M-L	M-H	L
R2	M			L	H	M-L
R3	H			L	H	M-H
R4		L		L	H	M-H
R5		M		M-L	H	M-L
R6		H		M-H	M-H	L
R7			L	L	H	L
R8			M	M-L	H	M-L
R9			H	M-H	M-H	M-H

3.5. Introducción de nuevas RBFs

En este punto el algoritmo sustituye las RBFs eliminadas por otras nuevas. Las nuevas RBFs se colocan en el centro de las zonas peor clasificadas y que están fuera del radio de cualquier RBF. El radio de la nueva RBF se inicializa con la media de los radios de las RBFs presentes en la población.

3.6. Selección de las mejores RBFs

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

4. Resultados Experimentales

Los datos usados en esta sección se han obtenido de UCI Repository of Machine Learning Database: Iris, Wine, Wbcd y Glass. El tamaño de la población es el mismo que el número de clases existentes en la base de datos usada. La estimación de la capacidad de generalización de las RBFNs se obtiene mediante 10-validación cruzada, y el número de ejecuciones se ha fijado en 200. Las Tablas de la 2 a la 5 muestran los resultados obtenidos con el CoEvRBFN anterior, el actualizado y con otros algoritmos de aprendizaje de RBFN descritos en la bibliografía especializada.

Un análisis de los resultados muestra que CoEvRBFN sigue obteniendo RBFNs con una estructura simple (el número de neuronas es igual al número de clases) y los resultados son comparables a los obtenidos con otros métodos. Además su capacidad de generalización sigue siendo igual o superior a la de otros métodos. Con las modificaciones introducidas se ha conseguido mejorar dicha capacidad en algunas ocasiones con un menor coste computacional.

Tabla 2. Resultados obtenidos con la base de datos Iris.

Algoritmo	*Nº RBF´s*	*Porcentaje de Clasificación*
Netlab [11]	4.5	96
Bing Yu [2]	6	97.33
Newrb [13]	9	79.37
Wallace [22]	3	98
Tian [20]	14.1	97.2
Topchy [21]	5	95.6
CoEvRBFN [16]	3	98.3
CoEvRBFN actual	3	99.3

Tabla 3. Resultados obtenidos con la base de datos Wine.

Algoritmo	*Nº RBF´s*	*Porcentaje de Clasificación*
Netlab [11]	3	98.9
Bing Yu [2]	20	96.3
Newrb [13]	58	92.8
Tian [20]	81.9	95.0
CoEvRBFN [16]	3	98.9
CoEvRBFN actual	3	99.3

Tabla 4. Resultados obtenidos con la base de datos Wbcd.

Algoritmo	*Nº RBF´s*	*Porcentaje de Clasificación*
Netlab [11]	2.2	97.1
Wallace [22]	2	96.6
CoEvRBFN [16]	2	98.2
CoEvRBFN actual	2	98.1

Tabla 5. Resultados obtenidos con la base de datos Glass.

Algoritmo	*Nº RBF´s*	*Porcentaje de Clasificación*
Bing Yu [2]	27	86.2
Newrb [13]	87	78.5
CoEvRBFN [16]	7	74.7
CoEvRBFN actual	7	75

5. Conclusiones

En este trabajo se presenta una optimización de CoEvRBFN. Se consigue reducir el costo computacional debido a que ahora sólo se entrenan los pesos de la red y no los centros y los radios como antes. Además, la introducción de dos modalidades de mutación 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 desinformada promueve modificaciones aleatorias de forma que favorezca la exploración del entorno y se huya de óptimos locales.

Los resultados conseguidos se igualan e incluso superan a los obtenidos con la versión anterior.

Como línea de trabajo futuro se ampliará el algoritmo para que sea capaz de tratar con características de tipo nominal teniendo en cuenta la interpretabilidad de las bases de datos manejadas.

Agradecimientos

Este trabajo ha sido parcialmente subvencionado por los proyectos españoles del CICYT TIN2004-01419 y TIN2005-04386-C05-03.

Referencias

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

[2] Bing Yu; Xingshi He (2006). " Training Radial Basis Function Networks with Differencial Evolution". IEEE International Conference on Granular Computing, pp. 369-372.

[3] Broomhead, D.; Lowe, D. (1988). "Multivariable functional interpolation and adaptive networks". Complex System. v 2. pp 321-355.

[4] Buchtala, O.; Klimek, M.; Sick, B. (2005)."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. pp. 928 – 947

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

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

[7] González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, J., Fco, A. (2003). "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, November 2003, pp 1478-1495.

[8] Lacerda, E; Carvalho, A.; Braga A.; Ludermir T. (2005) "Evolutionary Radial Functions for Credit Assessment". Applied Intelligence 22. Springer Netherlands. pp 167-181.

[9] Mandani, E.; Assilian, S. (1975)."An experiment in linguistic synthesis with a fuzzy logic controller". Int. J. Man-Machine Stud., v. 7, n. 1, pp. 1-13,

[10] Moody, J.; Darken, C, J. (1989). "Fast learning in networks of locally-tuned processing units". Neural Computing. vol 1. pp. 281-294.

[11] Nabvey,J.; Darken, C.J. Netlab Neural Network Software. http://www.ncrg.aston.ac.uk/netlab.
[12] Neruda, R.; Kudová, P. "Learning methods for radial basis function networks" (2005). Future Generation Computer Systems, Volume 21, Issue 7, pp 1131-1142.
[13] Newrb, Matlab neural networks toolbox.
[14] Park, J.; Sandberg, I. (1991)."Universal approximation using radial-basis function networks". Neural Comput., vol. 3, pp. 246-257.
[15] Pedrycz, W. (1998). "Conditional fuzzy clustering in the design of radial basis function neural networks ". IEEE Transactions on Neural Networks 9 (4), pp. 601-612.
[16] Pérez-Godoy, M; Rivera, A.J.; Jesus, M.J.; Rojas, I. (2007) "CoEvRBFN: an approach to solving the classification problem with a hybrid cooperative-coevolutive algorithm" . IWANN 07.
[17] Potter, M.; De Jong, K. (2000) "Cooperative Coevolution: an architecture for evolving coadapted subcomponents". Evolutionary Computation, 8(1), pp. 1-29.
[18] Powell, M. (1985). "Radial basis functions for multivariable interpolation: A review". In IMA. Conf. on Algorithms for the approximation of functions and data, pp. 143-167.
[19] Rivera, A.J.; Rojas, Ignacio; Ortega, Julio; del Jesús, M.José (2006) "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.
[20] Tian J.; Li, M-Q.; Chen F-Z. (2005). "A three-phase rbfnn learning algorithm for complex classification". International Conf. on Machine Learning and Cybernetics, pp. 4134-4139.
[21] Topchy A.; Lebedko, O.; Miagkikh V.; Kasabov N.;(1998). "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, pp. 253-258.
[22] Wallace, M.; Tsapatsoulis, N.; Kollias, S. (2005). "Intelligent initialization of resource allocating RBF networks". Neural Networks 18 (2), pp. 117-122.
[23] Whitehead, B; Choate, T. (1996) "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, pp.869-880.
[24] Widrow, B.; Lehr, M.A.(1990). "30 Years of adaptive neural networks: perceptron, madaline and backpropagation". Proceedings of the IEEE, v. 78 n. 9, sept. pp. 1415-1442