Sistema Multiagente para el diseño de Redes de Neuronas de Base Radial óptimas

J.M. Valls Ferrán, J.M. Molina López, I.M. Galván León

Dpto. de Informática, Universidad Carlos III de Madrid,
Av. De la Universidad, 30 28911, Leganés, Madrid, ESPAÑA
jvalls@inf.uc3m.es

Resumen

Las Redes de Neuronas de Base Radial (RNBR) se comportan muy bien en la aproximación de funciones, siendo su convergencia extremadamente rápida comparada con las redes de neuronas de tipo perceptrón multicapa. Sin embargo, el diseño de una RNBR para resolver un problema dado, no es sencillo ni inmediato, siendo el número de neuronas de la capa oculta de una Red de Base Radial (RBR) un factor crítico en el comportamiento de este tipo de redes. En este trabajo, el diseño de una RNBR está basado en la cooperación de n+1 agentes: n agentes, cada uno de ellos formado por una RBR, que denominamos *agentes RBR* y 1 *agente árbitro*. Estos n+1 agentes se organizan formando un Sistema Multiagente. El proceso de entrenamiento se distribuye entre los n agentes RBR, cada uno de los cuales tiene un número diferente de neuronas. Cada agente RBR se entrena durante un número determinado de ciclos (una etapa), cuando el agente árbitro le envía un mensaje. Todo el proceso está gobernado por el árbitro que decide cuál es el mejor agente RBR en cada etapa. El resultado de los experimentos muestra una importante reducción del número de ciclos de entrenamiento utilizando la estrategia multiagente propuesta, en lugar de una estrategia secuencial.

Palabras clave: Redes de Neuronas de Bases Radial, Optimización neuronas ocultas, Sistemas Multiagente.

1. Introducción

Las RNBR (Moody y Darken, 1989; Poggio y Girosi, 1990) se originan con la utilización de funciones de base radial, como las funciones Gaussianas, para la solución del problema de interpolación para funciones de varias variables. Estas redes se componen de una única capa oculta con funciones de activación local distribuidas en diferentes clases en el espacio de los datos de entrada. Esto tiene una gran importancia ya que las RNBR construyen aproximaciones locales entrada-salida, lo que permite obtener una convergencia muy rápida del algoritmo de aprendizaje (Moody y Darken, 1989; Galván et. al., 1995).

A pesar de estas ventajas, las RNBR no han sido ampliamente utilizadas en la resolución de problemas. Una de las principales razones radica en el hecho de que el diseño de una RNBR para resolver un determinado problema no es directo. El diseño de una buena RNBR implica determinar su topología, esto es, el número de unidades de base radial o unidades ocultas, lo cual es un factor crítico en el rendimiento de estas redes, puesto que al variar el número de neuronas ocultas, se obtienen comportamientos y aproximaciones muy diferentes.

En la actualidad, la determinación de topologías de redes de neuronas es una labor básicamente manual: depende en gran medida de la experiencia del experto, y en la mayoría de los casos, se emplea un tedioso procedimiento de prueba y error. Sin embargo, en los últimos años, ha aumentado el

interés de la comunidad científica por el desarrollo de métodos automáticos capaces de diseñar la topología de las redes de neuronas.

El mayor esfuerzo en este área se ha dirigido a redes de tipo perceptrón multicapa; sin embargo, en la literatura se encuentran algunos trabajos referidos a RNBR. Por ejemplo, en (Platt, 1991) el número de neuronas ocultas se inicializa a uno, y se crean nuevas neuronas cuando los patrones no pertenecen a ninguna de las clases existentes. Así se puede encontrar el número óptimo de neuronas ocultas. Hay una variación de este método en (Kadirkamanathan y Niranjan, 1993; Yingwei et al., 1997) donde se incorporan estrategias de poda para eliminar neuronas ocultas. Existe otra línea de investigación en este campo, donde se utilizan técnicas evolutivas, como algoritmos genéticos, para determinar el número óptimo de unidades ocultas (Gruau, 1993; Whitehead y Choate, 1994; Whitehead y Choate, 1996).

En este artículo se propone un nuevo sistema para determinar la RNBR óptima capaz de resolver un determinado problema. El sistema determina el número de neuronas ocultas que deben incorporarse en la RNBR para obtener la topología capaz de alcanzar el mínimo error con el menor número posible de ciclos de aprendizaje.

El sistema propuesto se compone de agentes RBR y de un agente árbitro que forman un sistema multiagente (Lux et al, 1995; Molina et al, 1997; Wesson et al, 1998). Los agentes RBR son RNBR con diferente topología, es decir, con diferente número de neuronas ocultas. Inicialmente, todos los agentes RBR se entrenan durante un número prefijado de ciclos de aprendizaje (una etapa), enviando a continuación un mensaje al agente árbitro con información sobre la evolución del entrenamiento. El agente árbitro decide qué agente RBR ha evolucionado mejor y le envía una señal para que continúe su entrenamiento durante una etapa más. Cuando finaliza esta etapa, comunica al agente árbitro su situación, y éste vuelve a seleccionar el mejor agente RBR. Este proceso se repite, hasta que un agente alcanza el objetivo, que consiste en conseguir un error menor que un valor prefijado. De este modo, sólo un agente RBR se está entrenando en un momento dado. Este sistema permite reducir el número de ciclos de entrenamiento total, comparado con un entrenamiento simultáneo de todas las RNBR.

El sistema multiagente propuesto es una arquitectura general que podría ser útil para realizar varias tareas simultánteamente; por ejemplo, los agentes podrían llevar a cabo diferentes tareas de entrenamiento (cada una para un problema distinto), o tareas de test: mientras algunos agentes se están entrenando, otros pueden realizar tareas de test. Esta es la ventaja de utilizar una metodología de sistemas multiagente en lugar de una metodología más convencional. En este trabajo, sólamente se ha considerado la tarea de entrenamiento para un problema específico.

La estructura del resto del artículo es la siguiente: La sección 2 presenta las RNBR, especificando las ecuaciones para calcular la salida, y el mecanismo para determinar los parámetros de la red. En la sección 3 se describe la arquitectura del sistema multiagente y del protocolo de comunicación utilizado. Los resultados experimentales se muestran en la sección 4, y algunas conclusiones derivadas de este trabajo se presentan en la sección 5.

2. Redes de Neuronas de Base Radial

Las RNBR se componen de tres capas (figura 1). La capa de entrada es simplemente un conjunto de unidades sensoriales que transmiten hacia las neuronas ocultas la entrada exterior. La segunda capa es la capa oculta; las neuronas de esta capa aplican una transformación no lineal al espacio de entrada. La tercera y última capa realiza una combinación lineal de las activaciones de las unidades de la capa oculta.

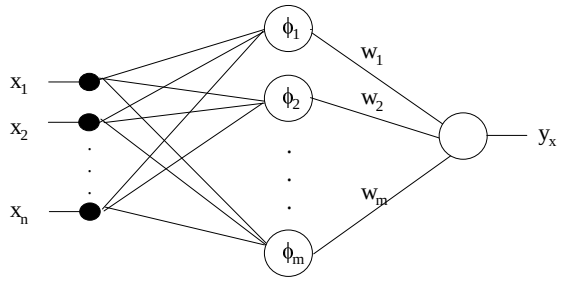

Fig. 1. Arquitectura de Redes de Base Radial.

Para una red con m neuronas ocultas, la ecuación que describe su salida es la siguiente:

$$\tilde{y}_x = w_0 + \sum_{i=1}^{m} w_i \cdot \phi_i(x)$$

donde $(w_i)_{i=1,\ldots,m}$ son los pesos de la red, los cuales se asocian a las conexiones de la capa oculta con la capa de salida; w_0 es el umbral asociado a la neurona de salida; $x=(x_1,\ldots,x_n)$ es la entrada externa de la red, y $(\phi_i)_{i=1,\ldots m}$ son las funciones de base

radial o funciones de activación de las neuronas de la capa oculta.

Las funciones de base radial normalmente son traslaciones de una función prototipo, ϕ, tal como:

$$\phi_i(x) = \phi\left(\frac{\|x - c_i\|}{d_i}\right)$$

donde $c_i = (c_{i1},...,c_{in}) \in \Re^n$ es el centro de la función ϕ_i, d_i es la desviación, anchura o factor de escala para el radio $\|x - c_i\|$ y $\|\cdot\|$ es la norma Euclidea. La clase de funciones prototipo más general es la función gaussiana:

$$\phi(r) = e^{-r^2/2}$$

El aprendizaje de una RNBR implica la determinación de los centros $(c_i)_{i=1,...,m}$, las desviaciones $(d_i)_{i=1,...,m}$, y los pesos $(w_i)_{i=1,...,m}$. En este trabajo, el aprendizaje de la red se lleva a cabo descomponiéndolo en dos fases: En la primera fase se determinan los centros c_i y las desviaciones d_i de un modo no supervisado, mientras que en la segunda fase se realiza la optimización de los pesos w_i por entrenamiento supervisado.

Durante la primera fase, se utiliza el algoritmo de las K-medias para clasificar el espacio de entrada en m clases; los centros de las m clases serán los parámetros c_i de la RNBR. Los coeficientes de desviación, d_i, se calculan como la raíz cuadrada del producto de las distancias desde el centro c_i a sus dos vecinos más cercanos. Durante la fase supervisada, se determinan los pesos de la red w_i, minimizando el error cuadrático medio:

$$E = \frac{1}{2N}\sum_{x=1}^{N}(y_x - \tilde{y}_x)^2$$

donde y_x y $\tilde{y}_x$ son la salida deseada y la salida de la red para el patrón de entrada x, respectivamente.

3. Arquitectura del Sistema Multiagente

El sistema multiagente propuesto en este artículo se ha diseñado de forma que pueda encontrar la topología óptima de la RNBR capaz de resolver un determinado problema. El sistema se compone de dos tipos de agentes: los agentes RBR y el agente árbitro. Los agentes RBR son RNBR con distinta topología, esto es, con un número diferente de neuronas ocultas. No existe comunicación entre ellos, sólo se comunican con el agente árbitro que dispone de información referente al comportamiento de todos los agentes RBR. Utilizando este conocimiento, el árbitro decide cuál es el agente RBR que mejor se comporta al final de cada etapa, es decir, el más apropiado para resolver el problema. A continuación se describen las tareas que realiza cada tipo de agente y su arquitectura.

Las responsabilidades del agente árbitro son las siguientes:

- Recolección de datos sobre los agentes RBR
- Monitorización del trabajo del sistema para asegurar que se alcancen los objetivos globales.
- Selección del agente RBR que tiene el mejor rendimiento, esto es, el que tiene el mejor valor de crédito asociado, como se explica en la sección 3.1.1.

Por otra parte, los agentes RBR deben llevar a cabo las siguientes tareas:

- Establecimiento de la comunicación con el agente árbitro.
- Ejecución de un número fijo de ciclos de aprendizaje cuando se lo indica el árbitro. Este número de ciclos prefijado se denomina *etapa*.
- Evaluación del valor de crédito asociado a su RNBR cuando se alcanza el final de la etapa.
- Comprobación del cumplimiento de la meta u objetivo final.
- Decisión sobre la detención del proceso de entrenamiento.

3.1 Arquitectura de los Agentes RBR

Como se ve en la figura 2, los agentes RBR tienen tres módulos diferentes: Módulo de comunicaciones, módulo de control y módulo de bajo nivel. A continuación se describe cada uno de ellos.

a) Módulo de Comunicaciones

Este módulo realiza las comunicaciones con el agente árbitro utilizando un sencillo protocolo que se describe en la sección 3.3.

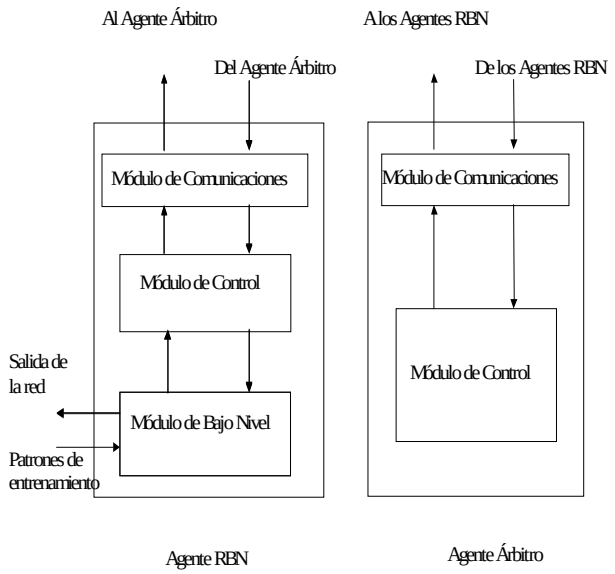

Fig. 2: Arquitectura del Sistema Multiagente: Agente RBR y Agente Árbitro

b) Módulo de Control

El módulo de control almacena información sobre el estado del entrenamiento de la RNBR, información que es útil para medir la adecuación de la red para resolver el problema dado:

- El número de ciclos de aprendizaje realizados hasta el momento actual.
- El error cuadrático medio alcanzado hasta el momento actual.
- La derivada del error cuadrático medio respecto del número de ciclos de entrenamiento.
- El error cuadrático medio deseado para resolver el problema. Este es el que llamaremos *error objetivo*.
- El valor del crédito asociado a la red.

Por otro lado, el módulo de control lleva a cabo las siguientes operaciones:

- Comenzar una nueva etapa de entrenamiento.
- Detener la etapa de entrenamiento si se alcanza el error objetivo.
- Calcular el valor del crédito, como se describe en el apartado 3.1.1.

c) Módulo de Bajo Nivel (Módulo RNBR)

El módulo de bajo nivel, es la red de neuronas de base radial en sentido estricto. Recibe los patrones de entrenamiento como datos de entrada, realiza una etapa de entrenamiento de la red y produce la salida de la red. Como se vio en el apartado 2, el entrenamiento de una RNBR se realiza en dos fases: en la primera fase, se inicializan los centros y las desviaciones, y en la segunda se actualizan los pesos. Pues bien, en las etapas de entrenamiento de los agentes RBR, únicamente se actualizan los pesos. La primera fase del entrenamiento se lleva a cabo una sola vez al inicio de todo el proceso.

3.1.1 Función de Evaluación del Crédito

Uno de los aspectos más importantes del Sistema Multiagente propuesto en este trabajo es la asignación del valor de crédito a cada agente RBR. El agente árbitro utiliza este valor para decidir qué agente debe comenzar una nueva etapa de entrenamiento. Por tanto, el resultado que proporciona el Sistema Multiagente depende en gran medida del valor de crédito elegido.

Se ha creído conveniente utilizar el número de ciclos de entrenamiento (n) y el error cuadrático medio (E) para evaluar el valor del crédito porque son los parámetros que más fielmente reflejan la calidad de una RNBR. En este trabajo, la función de crédito seleccionada es la siguiente:

$$C = \frac{1}{nE}$$

En la elección de la función, se han tenido en cuenta las siguientes consideraciones:

- Cuanto menor es el error, mejor se comportará la red. Por tanto, *C* debe ser inversamente proporcional a *E*.
- Cuanto antes se consiga un error pequeño, mejor. Por tanto, *C* debe ser inversamente proporcional al número de ciclos de entrenamiento *n*.

En cualquier caso, si la derivada del error cuadrático medio es muy pequeña (la convergencia de la red se ha alcanzado), y dicho error es mayor que el error objetivo, el valor de crédito asignado al agente es *-1*. Esto realmente significa que se descarta a este agente.

3.2 Arquitectura del Agente Árbitro

Como se ve en la figura 2, el agente árbitro consta de dos módulos:

a) Módulo de Comunicaciones

Este módulo lleva a cabo la comunicación con los agentes RBR. Utiliza el protocolo que se explica en 3.3.

b) Módulo de Control

- Almacena la información sobre los valores de crédito de cada agente RBR.

- Decide cuál es el mejor de los agentes RBR.

3.3 Protocolo de Comunicación

La gestión del conjunto de informaciones y acciones, denominados actos, en el nivel de comunicaciones se realiza mediante una estructura de agenda. En esta agenda se almacenarán todos los actos que involucran la cooperación entre agentes.

Los tipos de actos de la agenda se pueden dividir en acciones a ejecutar, de información o de negociación:

- Ejecución, este tipo de actos representan la ejecución de una tarea que debe ser realizada por el propio agente o por otros agentes.

- Información, este tipo de actos representan datos de alto nivel que informan de la situación del propio agente, de las tareas que está llevando a cabo, de la situación de otros agentes y de las tareas que estos llevan a cabo.

- Negociación, este tipo de actos tiene como cometido la adecuación entre las necesidades de un agente y las posibilidades de otro. Van desde la negativa a cooperar hasta la aceptación de la ejecución.

Los tipos concretos de los actos aquí utilizados son:

- *INFORM*, información, recoge la situación interna del agente y de las tareas internas que está realizando para ser enviadas a otro agente que las necesite.

- *REQUEST.ORDER*, ejecución, esta acción provoca el envío de un mensaje a otro agente para que ejecute una tarea sin negociación.

- *SUPPLY_INFO*, información, recoge la situación en la que se encuentran otros agentes o informaciones que otros agentes creen que deben ser consideradas en el agente.

- *REQUEST_TO_DO.ORDER*, ejecución, petición desde otro agente de realizar una tarea sin posibilidad de negociar. Este tipo de acciones son interpretadas directamente y enviadas al nivel inferior.

- *ACCEPT*, negociación, envío a otro agente del mensaje de aceptación de una acción que ha enviado previamente.
- *REJECT*, negociación, envío a otro agente del mensaje de negativa de realización de una acción que ha enviado previamente.

Fig. 3: Protocolo de Comunicaciones entre los diferenes agentes

El Agente Árbitro (M.A.: Manager Agent) inicia el proceso. Su módulo de comunicaciones genera un acto REQUEST.ORDER (Init, GP) , y como consecuencia se envía un mensaje a cada uno de los agentes RBR (R.A.: RBR Agent). GP es una estructura de datos que contiene los parámetros generales del sistema: Epsilon (Error Objetivo), MaxCycles (Máximo número de ciclos de entrenamiento), N (número de ciclos de entrenamiento en cada etapa de entrenamiento).

Cada R.A. reconoce este mensaje generando un acto ACCEPT (N_j), el cual, cuando se procesado, envía un mensaje de respuesta al agente árbitro, y

comienza su etapa de entrenamiento inicial: se calculan los centros y desviaciones de todas sus neuronas. Si el R.A: no está preparado para realizar esta tarea, genera un acto REJECT (N_j) que, cuando es procesado, envía un mensaje al M.A.

El M.A. espera hasta tener todas las respuestas de los R.A. De ahora en adelante, la comunicación está establecida entre el M.A. y los R.A. que aceptaron el primer mensaje.

Cuando los R.A. finalizan su etapa de entrenamiento inicial, su módulo de control calcula el valor de crédito, y lo envía al módulo de comunicación que genera un acto INFORM (N_i, C) que produce un mensaje dirigido al M.A. El módulo de control del M.A. escribe el valor de C en una tabla indexada con el número del R.A. Esta tabla se llama tabla de créditos.

Una vez que el M.A. han recibido los mensajes de todos los R.A., selecciona el mejor valor de la tabla de créditos, toma su índice j, y genera un acto REQUEST.ORDER (Train,Nj) que produce un mensaje que es enviado al R.A. número j. Este agente reconoce este mensaje con un acto ACCEPT, y comienza una nueva etapa de entrenamiento.

Mientras tanto, el M.A. permanece dormido hasta que se recibe un mensaje INFORM(N_j,C) del agente j. Entonces, el M.A. escribe el nuevo valor de C en la tabla, y vuelve a seleccionar el mejor valor, toma su índice j, y genera un REQUEST.ORDER(Train,Nj).

La negociación se termina cuando:

1. Todos los valores de crédito son -1, es decir, ninguno de los R.A. es adecuado. En este caso, el módulo de comunicaciones del M.A. genera un acto REQUEST.ORDER(Kill,Fail) que, cuando se procesa, envía un mensaje a cada R.A. para detener el sistema. El resultado es que ninguno de los agentes RBR es capaz de realizar la tarea.

2. Un R.A. (el k-ésimo) alcanza un error E tal que $E_K <= E_{goal}$. En este caso, este agente detiene su etapa de entrenamiento, y genera un acto INFORM(Success,Status,Nk) que produce el respectivo mensaje al M.A. que genera un acto REQUEST.ORDER(Success,Nk), el cual implica que se envía un mensaje a cada uno de los R.A. para parar el sistema, informando que se ha alcanzado el objetivo por parte del agente k-ésimo.

4. Resultados Experimentales

El Sistema Multiagente propuesto en este trabajo, se ha aplicado a la aproximación de la función

$$x(t) = \lambda\ x(t-1)\left[1 - x(t-1)\right]$$

con $\lambda = 3.97$, y $x(0) = 0.5$. Esta función describe una serie temporal fuertemente caótica, llamada serie temporal logística. El objetivo del sistema es encontrar la topología óptima de una RNBR que sea capaz de aproximar de manera adecuada.

Se han realizado varias simulaciones con el Sistema Multiagente utilizando tres conjuntos de datos diferentes: Datos de la serie logística desde t=0 hasta t=60, desde t=0 hasta t=100 y desde t=0 hasta t=140.

Para todas las simulaciones, se han considerado 10 agentes RBR, con una topología de 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50 neuronas ocultas, respectivamente. En la tabla 1, se muestran las RNBR óptimas obtenidas por el sistema propuesto para cada simulación, cuando el objetivo es alcanzar un error cuadrático medio menor que 0.001.

Número de patrones	Error mínimo	RNBR óptima
60	0.001	10 neuronas ocultas
100	0.001	15 neuronas ocultas
140	0.001	15 neuronas ocultas

Tabla 1: RNBR óptimas obtenidas por el Sistema Multiagente para cada simulación

Para validar los resultados obtenidos por el Sistema Multiagente, y comprobar si las redes obtenidas son efectivamente las mejores, se han entrenado todas las RNBR simultáneamente (desde 5 a 50 neuronas). La figura 4 muestra la evolución de los errores cuadráticos medios durante los 100 primeros ciclos de entrenamiento para 60 patrones de entrenamiento. En esta figura se puede observar que la RNBR que alcanza el error objetivo (menor que 0.001) utilizando el menor número de ciclos de entrenamiento es la RNBR con 10 neuronas ocultas, lo cual coincide con la decisión del Sistema Multiagente. Cuando se utilizan 100 y 140 patrones

de entrenamiento, la situación es similar, la mejor red en cada uno de los casos, es la formada por 15 neuronas, como se observa en la figura 5 y 6 respectivamente. Esto muestra que la solución proporcionado por el Sistema Multiagente es la mejor red capaz de resolver el problema dado.

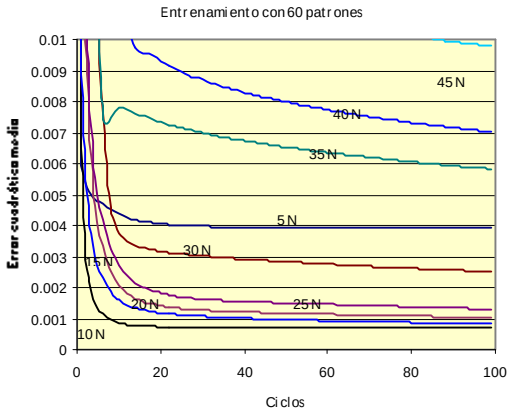

Fig. 4: Evolución del error cuadrático medio para diferentes RNBR (10, 15, 20, ...50 neuronas ocultas) con 60 patrones de entrenamiento

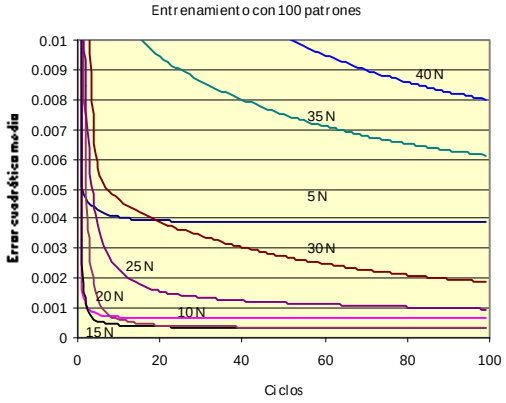

Fig. 5: Evolución del error cuadrático medio para diferentes RNBR (10, 15, 20, ...50 neuronas ocultas) con 100 patrones de entrenamiento

Número de ciclos de entrenamiento del Sistema Multiagente vs. entrenamiento simultáneo.

Se ha comparado el número de ciclos de entrenamiento implicado en el Sistema Multiagente con el número de ciclos que se requieren cuando todas las RNBR se entrenan simultáneamente. Cada experimento está definido por el error cuadrático medio que debe ser alcanzado por las redes.

En la tabla 2 se muestra el número de ciclos requeridos por el Sistema Multiagente y por el entrenamiento simultáneo de todas las RNBR, cuando se utiliza un conjunto de entrenamiento de 60 patrones. También se ha medido el número de ciclos de entrenamiento con 100 patrones (ver tabla 3). Como se puede observar en las tablas 2 y 3, el Sistema Multiagente necesita un menor número de ciclos de entrenamiento. El porcentaje de reducción se incrementa cuando el error mínimo exigido disminuye. Cuando se utiliza un conjunto de entrenamiento con 60 patrones y para un error objetivo de 0.001, la relación entre el número de ciclos con entrenamiento simultáneo y con el sistema multiagente propuesto es de 90/36= 2.5

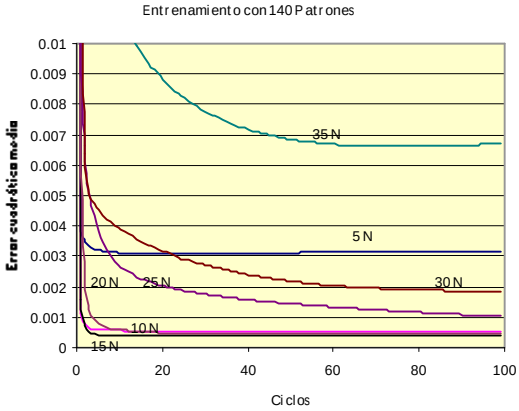

Fig. 6: Evolución del error cuadrático medio para diferentes RNBR (10, 15, 20, ...50 neuronas ocultas) con 140 patrones de entrenamiento

Aprendizaje con 60 patrones de entrenamineto		
Error Objetivo	Número de ciclos	
	Entrenamineto Simultáneo	Sistema Multiagente
0.001000	90	36
0.000950	90	36
0.000900	100	37
0.000850	120	39
0.000800	140	41
0.000750	180	45
0.000700	380	85
0.000650	1920	659

Tabla 2: Sistema Multiagente versus entrenamiento simultáneo con 60 patrones de entrenamiento

Para un error objetivo de 0.0007 , la relación es 380/85=4.47. Algo similar ocurre cuando utilizamos 100 patrones de entrenamiento.

Aprendizaje con 100 patrones de entrenamineto		
Error Objetivo	Número de ciclos	
	Entrenamineto Simultáneo	Sistema Multiagente
0.001000	40	31
0.000950	40	31
0.000900	40	31
0.000850	40	31
0.000800	40	31
0.000750	50	32
0.000700	50	32
0.000650	50	32
0.000600	60	33
0.000550	70	34
0.000500	80	35
0.000450	100	42
0.000400	130	46
0.000350	860	270

Tabla 3: Sistema Multiagente versus entrenamiento simultáneo con 100 patrones de entrenamiento

5. Conclusiones

El diseño de RNBR para un problema dado está basado actualmente en la experiencia del desarrollador. Este trabajo propone el uso de diferentes redes de neuronas de base radial y la cooperación entre ellas en un sistema multiagente para detectar de forma automática la RNBR que mejor se adapta al problema dado.

El Sistema Multiagente se compone de 'n+1' agentes: 'n' agentes RBR y el agente árbitro. La realización de un Sistema Multiagente permite aprovechar las ventajas generales de los sistemas distribuidos, como son: la robustez frente a caídas de un agente y la facilidad de distribución de tareas entre servidores. Los resultados experimentales muestran que dicho sistema junto con el protocolo de comunicaciones desarrollado es capaz de encontrar la red de base radial óptima para el problema dado. El problema de encontrar la red óptima puede ser abordado mediante una búsqueda exhaustiva pero el Sistema Multiagente permite reducir el coste computacional. De este modo, el número de ciclos de aprendizaje -coste computacional- requerido para conseguir dicha arquitectura es menor que si se entrenaran simultáneamente el conjunto de redes iniciales, mostrándose así la superioridad de la estrategia multiagente frente a una estrategia convencional.

Referencias

Galván I.M., Zaldívar J.M., Hernández H., Molga E.: The Use of Neural Networks for Fitting Complex Kinetic Data. Computer Chem. Engng. Vol. 20, No 12, pp. 1451-1465. 1996.

Gruau F.: Genetic synthesis of modular neural networks. Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo CA, 318-325, 1995.

Kadirkamanathan V. and Niranjan M.: A Function Estimation Approach to Sequential Learning with Neural Networks. Neural Computation 5, 954-975, 1993.

Lux A., Steiner D.: Understanding Cooperation: an Agent's Perspective. Proc. International Conference on Multiagent Systems, ICMAS-95, AAAI Press. San Francisco (CA), 1995.

Molina J. M., Jiménez F. J., Casar J. R.: Cooperative Management of Netted Surveillance Sensors. IEEE International Conference on Systems, Man and Cybernetics, pp 845-850. Orlando, EEUU. 1997.

Moody J.E. and Darken C.J.: Fast Learning in Networks of Locally-Tuned Processing Units. Neural Computation 1, 281-294, 1989.

Platt J.: A Resource-Allocating Network for Function Implementation. Neural Computation 3, 213-225, 1991.

Poggio T. and Girosi F.: Networks for approximation and learning. Proceedings of the IEEE, 78, 1481-1497, 1990.

Wesson R. et al: Network Structures for Distributed Situation Assessment", Readings in Distributed Artificial Intelligence, Ed. Alan H. Bond and Les Gasser, Morgan Kaufman 1988.

Whitehead B.A. and Choate T.D.: Evolving Space-Filling Curves to Distribute Radial Basis Functions Over an Input Space. IEEE Transactions on Neural Networks 5,1, 15-23, 1994.

Whitehead B.A. and Choate T.D.: Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction. IEEE Transactions on Neural Networks 7, 4, 869-880, 1996.

Yingwei L., Sundararajan N. and Saratchandran P.: A Sequential Learning Scheme for Function Approximation using Minimal Radial Basis Function Neural Networks. Neural Computation 9, 461-478, 1997.