Revista Iberoamericana de Automática e Informática Industrial
CEA comité español de automática
RIAI
ISSN: 1697-7912. Vol. 4, Núm. 2, Abril 2007, pp. 32-42
http://riai.isa.upv.es

IDENTIFICACIÓN DE SISTEMAS DINÁMICOS UTILIZANDO REDES NEURONALES RBF

Ricardo Valverde Gil*, Diego Gachet Páez**

**School of Engineering, San Francisco State University*
1600 Holloway Ave. San Francisco, California, 94044. Email:valverde@sfsu.edu

*** Escuela Superior Politécnica, Universidad Europea de Madrid*
Villaviciosa de Odón, 28670, Madrid, España, E-mail: gachet@uem.es

Resumen: La identificación de sistemas complejos y no-lineales ocupa un lugar importante en las arquitecturas de neurocontrol, como por ejemplo el control inverso, control adaptativo directo e indirecto, etc. Es habitual en esos enfoques utilizar redes neuronales "feedforward" con memoria en la entrada (Tapped Delay) o bien redes recurrentes (modelos de Elman o Jordan) entrenadas off-line para capturar la dinámica del sistema (directa o inversa) y utilizarla en el lazo de control. En este artículo presentamos un esquema de identificación basado en redes del tipo RBF (Radial Basis Function) que se entrena on-line y que dinámicamente modifica su estructura (número de nodos o elementos en la capa oculta) permitiendo una implementación en tiempo real del identificador en el lazo de control.

Palabras Clave: Identificación, sistemas no-lineales, redes neuronales, estimación de parámetros.

1. INTRODUCCION

Las redes neuronales artificiales constituyen una excelente herramienta para el aprendizaje de relaciones complejas a partir de un conjunto de ejemplos. Gracias a esto se ha generado un gran interés en la utilización de modelos de redes neuronales para la identificación de sistemas dinámicos con no-linealidades desconocidas. Además, debido a sus capacidades de aproximación así como a sus inherentes características de adaptabilidad, las redes neuronales artificiales presentan una importante alternativa en el modelado de sistemas no-lineales. Por último la posibilidad de implementación en paralelo y su respuesta relativamente rápida constituyen un incentivo para la investigación futura utilizando este enfoque en problemas que involucren sistemas dinámicos no-lineales.
La mayoría de sistemas pueden ser representados en forma discreta por un conjunto de ecuaciones.

$$\begin{aligned} x(k+1) &= \Gamma\left[\ x(k), u(k)\right] \\ y(k) &= \Psi\left[x(k)\right] \end{aligned} \tag{1}$$

donde x(.),y(.),u(.) son secuencias de tiempo discreto. Cuando las funciones Γ y Ψ son desconocidas estamos ante un problema de identificación.

Si el sistema descrito por (1) es lineal e invariante en el tiempo las ecuaciones que describen el sistema se pueden expresar como:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2}$$

En este caso A, B, C son matrices y el sistema puede ser parametrizado por el conjunto (A, B, C). Existe una abundante literatura sobre identificación de sistemas lineales. Para este tipo de sistemas, si el orden es conocido, se puede escoger la estructura del modelo y nos queda un problema de estimación de

parámetros. Este procedimiento no es aplicable a la identificación de sistemas no-lineales donde la estructura del modelo tiene que ser justificada.

En general un sistema no-lineal descrito por (1) puede ser representado en términos de entrada-salida como sigue:

$$y(k+1) = f[y(k), y(k-1), ..., y(k-n+1); \\ u(k), u(k-1), ..., u(k-m+1)\ ] \quad (3)$$

El problema de la identificación consiste en escoger un modelo apropiado y ajustar sus parámetros de acuerdo a alguna ley adaptativa de forma que la respuesta del modelo a una señal de entrada (o bien un conjunto de señales de entrada) se aproxime a la respuesta del sistema real a esa misma entrada.

Dado que el sistema real es desconocido, debemos asumir que corresponde a un tipo determinado, y que un modelo parametrizado basado en (3) puede en teoría representar el comportamiento de entrada-salida de cualquier sistema de ese tipo. De esta forma la identificación se reduce a un problema de estimación de parámetros. En nuestro caso el objetivo es aproximar la función f() (dinámica del sistema) utilizando redes neuronales, por lo que la identificación consiste en ajustar los parámetros de la red neuronal de manera que aproxime la función f().

La habilidad de las redes neuronales para aproximar un amplio conjunto de funciones no-lineales las convierten en candidatos ideales para modelar la dinámica de sistemas no-lineales.

Es fácil encontrar ejemplos (Narendra y Parthasaraty, 1990; Kuschewski *et al*, 1993; Chi *et al*, 1990; Chen *et al*, 1990) de utilización de redes "feedforward" con memoria Tapped Delay entrenadas off-line mediante un algoritmo de retropropagación (backpropagation) actuando como identificadores (Le Cun, 1985; Parker, 1985; Rumelhart *et al*, 1986; Werbos, 1974). Este enfoque no permite la identificación de sistemas con parámetros variables en el tiempo debido a que la red es incapaz de ajustar sus pesos on-line. Para evitar este problema, proponemos la utilización de una red neuronal RBF (Radial Basis Function) dinámica, es decir, de tamaño variable que puede ser entrenada on-line e implementarse en tiempo real (Poggio y Girosi, 1989).

2. ESTRUCTURA DE LA RED NEURONAL

La red RBF que utilizamos consiste en una estructura de procesamiento con dos capas, como se muestra en la figura 1. Existen $\mathbf{n_I}$ nodos de entrada, $\mathbf{n_H}$ nodos ocultos y $\mathbf{n_O}$ nodos de salida.

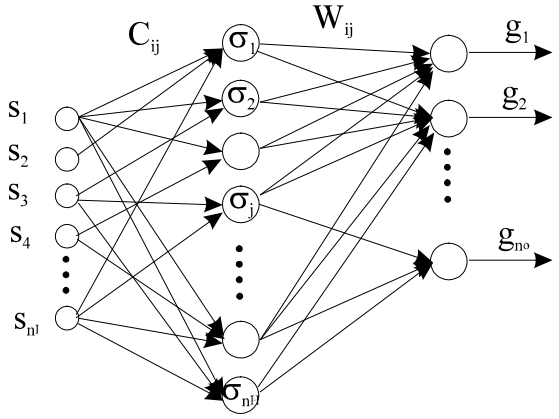


Figura. 1. Estructura de la Red Neuronal

Los pesos $\mathbf{c_{ij}}$ asociados a cada nodo en la capa oculta constituyen el *centro* de ese nodo en el espacio de entrada. Durante la propagación hacia adelante, cada nodo en esta capa calcula la distancia Euclidea entre el centro y el vector de entradas $(s_1, s_2, ...s_{n_I})$ y el resultado se pasa a través de una función de activación no-lineal, en nuestro caso una función gausiana, a fin de obtener el valor de activación de cada nodo.

$$a_j = f(\|\vec{s} - \vec{c}_j\|, \sigma_j) \quad 1 \le j \le n_H \\ f(x, \sigma) = e^{-\frac{x^2}{\sigma^2}} \quad (4)$$

Estos valores se normalizan para toda la capa oculta y se obtiene el valor de salida para cada nodo en esta capa.

$$h_j = \frac{a_j}{\sum_{k=1}^{n_H} a_k} \quad (5)$$

Cada nodo en la capa de salida calcula su valor como la suma ponderada ($\mathbf{w_{ij}}$) de los valores dados por la capa oculta.

$$g_i(\vec{s}) = \sum_{j=1}^{n_H} w_{ij} \cdot h_j, \quad 1 \le i \le n_O \quad (6)$$

En general, la función de entrada-salida dada por la red es un mapeo $G: R^{n_I} \rightarrow R^{n_O}$

$$g_i(\vec{s}) = \frac{\sum_{j=1}^{n_H} w_{ij} \cdot a_j}{\sum_{j=1}^{n_H} a_j} = \frac{\sum_{j=1}^{n_H} w_{ij} \cdot f(\|\vec{s} - \vec{c}_j\|, \sigma_j)}{\sum_{j=1}^{n_H} f(\|\vec{s} - \vec{c}_j\|, \sigma_j)} \\ 1 \le i \le n_O \quad (7)$$

De manera que en un instante *k*, la red calcula una predicción del vector de salida de esta forma:

$$\vec{y}^n(k) = G(\vec{s}(k), \phi) \quad (8)$$

La arquitectura neuronal se utiliza para "copiar" el comportamiento dinámico del sistema que está siendo identificado. Para esto el vector de entrada a la red neuronal en el instante *k*, $\vec{s}(k)$, consiste de las señales de entrada al sistema y los valores retardados tanto de la entrada como de la salida $\vec{u}(k)$, $\vec{u}(k-1)$, ... , $\vec{u}(k-n)$, $\vec{y}(k-1)$, ... , $\vec{y}(k-m)$ (memoria Tapped Delay) (Goodwin y Sin, 1984; Widrow y Stearns, 1985).

El vector de salida $\vec{y}^n(k)$ reemplaza la salida del sistema cuando la red neuronal está ya entrenada. Esto se muestra en las figuras 2 y 3 y se explica con más detalle en la sección siguiente. **Φ** es el vector de parámetros modificables de nuestro modelo neuronal. En general **Φ** está formado por los centros de los nodos ocultos, el ancho de la función gausiana de cada nodo oculto y los pesos de la capa de salida.

$$\phi = [c_{ij}, \sigma_j, w_{ij}] \quad (9)$$

Con el fin de identificar el sistema de forma adecuada debemos encontrar el conjunto de parámetros Φ^* que produce la mejor estimación de la respuesta del sistema para el subconjunto de valores de entrada-salida usado para la validación. En la literatura se encuentran algunos ejemplos de utilización de redes de tipo RBF para la identificación de sistemas dinámicos (Sanner y Slotine, 1991; Sjöber *et al*, 1995). La mayoría de ellos ubican a priori los centros de los nodos de la capa oculta formando una malla que cubre el espacio de entrada. Además fijan un valor adecuado para el ancho **σ** que es el mismo para todos los nodos. De esta forma el vector de parámetros que debe ser optimizado mediante aprendizaje consiste solo en los pesos asociados a la capa de salida **Φ=[w_{ij}]**. Se consiguen buenos resultados utilizando este enfoque, pero existe un gran inconveniente: el número de nodos en la capa oculta crece de manera exponencial al aumentar la dimensión del espacio de entrada.

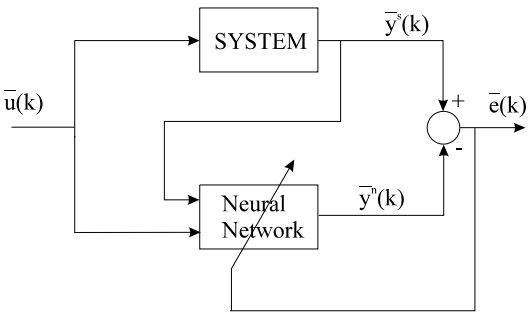


Figura 2. Esquema de identificación Serie/ Paralelo

Hay varias maneras de solucionar este problema (Panchapakesan y Palaniswami, 2002; Peng *et al*, 2004; Zemouri *et al*, 2003). Una de las más interesantes es la ubicación dinámica de los centros de los nodos de la red RBF como se describe en (Obradovic 1996).

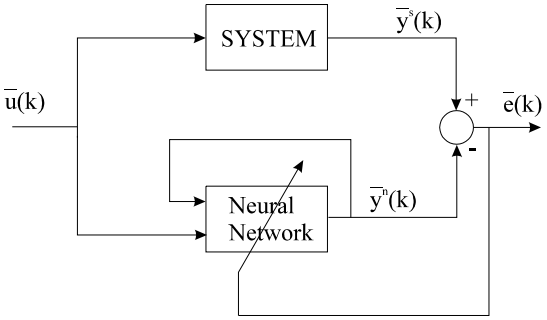


Figura 3. Esquema de Identificación Paralelo

La solución que aquí se propone está basada en un algoritmo que trabaja on-line. Se ubican dinámicamente los nodos en el lugar donde son necesarios en el espacio de entrada. Los valores del ancho σ_j de las funciones de activación son diferentes para cada nodo y se modifican (reducen) en las áreas en las que la identificación del sistema es más difícil (error de predicción alto).

Por lo tanto la estructura de la red se forma dinámicamente y los nodos se ubican solamente allí donde se necesitan de forma que se cumplan las especificaciones de rendimiento requeridas, manteniendo la estructura de la red lo más pequeña posible.

3. ALGORITMO DE ENTRENAMIENTO

El algoritmo de entrenamiento que presentamos tiene la ventaja de poder ser utilizado on-line. Podría igualmente ser utilizado off-line, pero se desperdiciaría una de sus principales virtudes.

De los dos esquemas posibles de identificación descritos en (Narendra y Parthasarathy, 1989) y representados en las figuras 2 y 3, hemos escogido el Serie-Paralelo para entrenar la red. Es decir, tenemos en cuenta los valores previos del sistema en lugar de los valores estimados dados por la red neuronal.

En la fase de funcionamiento escogeremos el esquema a utilizar dependiendo de la aplicación que le demos al modelo. Para propósitos de simulación, el esquema paralelo es la única posibilidad. En cambio si incluimos el modelo en un sistema de control podremos utilizar el esquema serie-paralelo. El error de predicción es mucho menor en el esquema serie-paralelo, pero debe quedar claro que sólo podremos predecir un paso adelante, aunque esto es suficiente para algunos sistemas de control (Compensación FeedForward).

3.1 Adaptación de Pesos en la Capa de Salida.

En el entrenamiento la red calcula el valor de salida estimado $\vec{y}^n(k)$ para cada vector de entrada $\vec{s}(k)$ propagándolo a través de las capas ocultas y de salida, como vio en las ecuaciones anteriores.

Después , el valor estimado se compara con el valor actual de salida proporcionado por el sistema, $\vec{y}^s(k)$, y se utiliza el error para modificar los parámetros de la red de forma que éste se reduzca. En cada iteración los pesos $\mathbf{w_{ij}}$ se ajustan considerando $\mathbf{c_{ij}}$ y $\boldsymbol{\sigma}_\mathbf{j}$ constantes. Más tarde veremos como se asignan estos valores. Para este ajuste utilizamos la regla delta modificada:

$$\begin{aligned} \vec{e}(k) &= \vec{y}^s(k) - \vec{y}^n(k) \\ e_i(k) &= y_i^s(k) - y_i^n(k) \\ \Delta w_{ij} &= \alpha \cdot e_i \cdot h_j \end{aligned} \qquad (10)$$

donde α es la velocidad de aprendizaje (utilizamos valores en el rango 0.5-0.9,) y $\mathbf{h_j}$ es el valor de salida de cada nodo en la capa oculta.

3.2 Creación de Nuevos Nodos.

Como hemos mencionado, en nuestra red neuronal los nodos se ubican dinámicamente donde se necesitan. Cada vez que se recibe un vector de entrada se calcula el valor de activación para cada nodo

$$a_j(s) = e^{-\frac{\|\vec{s}-\vec{c}_j\|^2}{\sigma_j^2}} \qquad (11)$$

Si existe un nodo cuyo valor de activación es mayor que un umbral predefinido $\mathbf{a_{min}}$, podemos decir que esta zona del espacio de entrada está cubierta. Si no hay ningún valor de activación mayor que el umbral $\mathbf{a_{min}}$, podemos decir que esta zona no está cubierta, y en este caso se crea un nuevo nodo cuyo centro se ubica exactamente en el punto definido por el vector de entrada:

$$c_{i,n_H+1} = s_i \qquad (12)$$

Los pesos de salida asociados a los nuevos nodos son aquellos que hacen que la red (con su nueva estructura) de exactamente el valor correcto de salida para esta situación, $\vec{y}^s(k)$, leída del sistema real:

$$w_{i,n_H+1} = y_i^s(k) \cdot (1 + \sum_{j=1}^{n_H} a_j) - y_i^n(k) \cdot (\sum_{j=1}^{n_H} a_j) \qquad (13)$$

Donde $\vec{y}^n(k)$ es el valor dado por la red antes de incluir el nuevo nodo, el parámetro $\mathbf{a_{min}}$, cuyo significado se aclara en la figura 4, se define a priori para toda la fase de entrenamiento de la red.

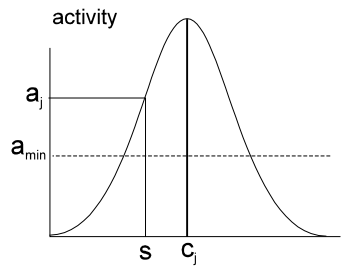


Figura 4. Valor de activación mínimo para un nodo oculto

Este valor no es crítico. Utilizamos valores en el rango 0.5-0.8. Cuanto más alto es, más alta es la capacidad de almacenamiento de información de la red pero también significa muchos más nodos en la capa oculta y un entrenamiento más lento.

La red neuronal se empieza a construir de cero. El primer nodo se ubica en el lugar del primer valor del vector de entrada y a partir de ahí crece.

3.3 Actualización de $\boldsymbol{\sigma}$

Con $\mathbf{a_{min}}$ fijado a priori, la densidad de los nodos en el espacio de entrada se controla con el ancho de la función de activación $\boldsymbol{\sigma}_\mathbf{j}$.

Cuanto más alto el valor de $\boldsymbol{\sigma}_\mathbf{j}$, menor el número de nodos que se necesitan para cubrir el espacio de entrada, pero también se tiene menos capacidad de almacenamiento de información en esa zona.

Si deseamos aproximar una función de variaciones suaves lo podemos hacer con muy pocos nodos, pero si queremos aproximar con precisión una función con cambios bruscos necesitamos muchos nodos.

Cuando tratamos de identificar un sistema dinámico no conocemos sus características, si tiene cambios bruscos o suaves, y en que zonas es más suave o más brusco. Para obtener un alto grado de precisión en la información pero manteniendo un número razonable de nodos utilizamos un $\boldsymbol{\sigma}$ diferente en cada zona del espacio de entrada.

De hecho en nuestra red cada nodo RBF tiene su propio valor de ancho $\boldsymbol{\sigma}_\mathbf{j}$. Tenemos que utilizar un valor bajo de $\boldsymbol{\sigma}$ en las zonas más *difíciles* del espacio de entrada. Para ello modificamos el valor de $\boldsymbol{\sigma}_\mathbf{j}$ en cada ciclo de entrenamiento teniendo en cuenta el error de predicción:

$$\Delta\sigma_j = \begin{cases} -\gamma \cdot \sigma_j \cdot h_j^2 & \text{if } \|\vec{e}\| > 1 \\ -\gamma \cdot \|\vec{e}\| \cdot \sigma_j \cdot h_j^2 & \text{if } E_{th} < \|\vec{e}\| < 1 \\ 0 & \text{if } \|\vec{e}\| < E_{th} \end{cases} \quad (14)$$

Este procedimiento de actualización mantiene valores pequeños de σ_j en zonas donde el error es grande. Cuando el error de predicción cae por debajo de un umbral, E_{th}, que se escoge teniendo en cuenta la precisión requerida para el modelo (rango 0.1-0.3), entonces σ_j no se actualiza. γ es un factor de decrecimiento en el intervalo [0,1]. Utilizamos valores en el rango 0.1-0.4. Este valor no debe ser demasiado alto. La velocidad de aprendizaje de los nodos de salida (w_{ij}) debe ser mucho más alta que la velocidad de creación de nuevos nodos para que se mantenga la estabilidad de la estructura de la red ($\alpha >> \gamma$).

Cuando se crea un nodo, se le asigna un nuevo σ_{n_H+1} que es la media de la σ de los nodos cercanos.

$$\sigma_{n_H+1} = \sum_{j=1}^{n_H} h_j \cdot \sigma_j \quad (15)$$

Al primer nodo se le asigna un valor σ muy alto y paulatinamente se reduce.

4. RESULTADOS EXPERIMENTALES

En esta sección presentamos los resultados obtenidos con nuestra red neuronal en la identificación de tres sistemas distintos.
Los parámetros de aprendizaje utilizados en todos los ejemplos mostrados a continuación son $\alpha=0.9$, $a_{min}=0.5$, $E_{th}=0.1$, $\gamma=0.4$.

4.1 Sistema Lineal con una entrada y una salida (SISO)

El primer sistema es lineal de tipo SISO (una entrada y una salida) con función de transferencia:

$$G(s) = \frac{K \cdot ((s+c)^2 + v^2)}{(s+b)((s+a)^2 + w^2)} \quad (16)$$

donde $a=1.0$, $b=2.5$, $c=1.25$, $w = \frac{2\pi}{2.5}$, $v=1.649$.
Utilizando un retenedor de orden cero y un período de muestreo $T=0.08$ s, la ecuación en tiempo discreto es:

$$\begin{aligned} y(k) = {} & A_1 y(k-1) + A_2 y(k-2) + A_3 y(k-3) \\ & + B_1 u(k-1) + B_2 u(k-2) + B_3 u(k-3) \end{aligned} \quad (17)$$

con $A_1=2.627771$, $A_2=-2.333261$, $A_3=0.697676$, $B_1=0.017203$, $B_2=-0.030862$, $B_3=0.014086$.

En este caso utilizamos un modelo neuronal con 4 entradas **u(*k*), y(*k-1*), y(*k-2*), y(*k-3*)**, 1 salida, **y(*k*)**, y una capa oculta cuyo número de nodos se incrementa durante el aprendizaje. No se han utilizado todas las entradas que se obtendrían del conocimiento de la ecuación en tiempo discreto (**u(*k-1*), u(*k-2*)** no se utilizan). Puesto que la ecuación del sistema a identificar normalmente no es conocida quisimos observar el resultado cuando las entradas a la red no encajan exactamente con la ecuación discreta del sistema.

La red se entrenó on-line siguiendo el esquema serie-paralelo y utilizando como señal externa de entrada una secuencia de pulsos de frecuencia y amplitud aleatorias.

A fin de demostrar la habilidad de la red para adaptarse on-line a los cambios en la dinámica del sistema los parámetros del mismo se modificaron una vez que la red estaba entrenada. Los nuevos parámetros de la función de transferencia fueron $a=0.75$, $w = \frac{2\pi}{3.5}$, mientras que b, c y v permanecieron inalterados. Estos nuevos valores dan como resultado una sistema más lento.

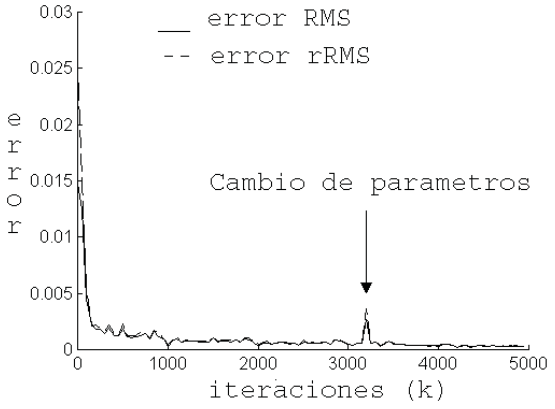


Figura 5. Evolución del error

La Figura 5 presenta la evolución de los errores medios cuadráticos absoluto y relativo durante el aprendizaje de la red. Para calcular el error utilizamos las últimas 625 (N=625) iteraciones.

Los errores RMS y rRMS se definen como:

$$RMS \text{ error} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i^s - y_i^n)^2} \quad (18)$$

$$rRMS\ \text{error} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i^s - y_i^n)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} y_i^{s\,2}}} \qquad (19)$$

$\vec{y}^s \rightarrow$ Salida del Sistema

$\vec{y}^n \rightarrow$ Salida del modelo Neuronal

En la figura 6 podemos ver la evolución del tamaño de la capa oculta durante el aprendizaje.

Las figuras 5 y 6 muestran como la red reacciona a los cambios en la dinámica del sistema. El error se incrementa después del cambio y entonces la red modifica su estructura (más nodos en la capa oculta) para adaptarse a los nuevos requerimientos y aprende la nueva dinámica rápidamente.

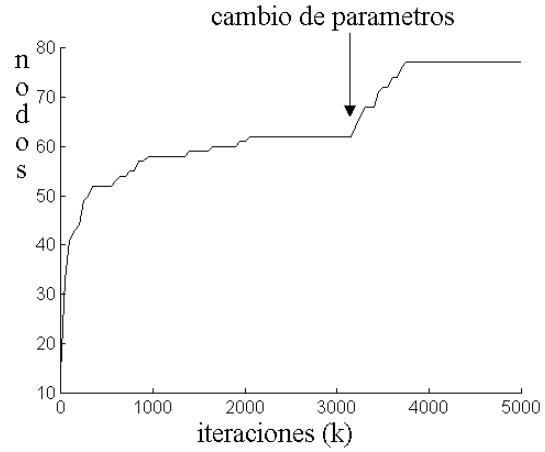


Figura 6. Evolución del tamaño de la capa oculta

Para analizar la calidad y precisión de la red entrenada comparamos la respuesta del sistema con la del modelo neuronal. Para ello se utilizó el esquema paralelo y se congeló durante la prueba la capacidad de aprendizaje. Se utilizaron dos señales de entrada distintas. La primera señal de entrada es un escalón de magnitud 18.2914 (el inverso de la ganancia del sistema).

Podemos ver ambas respuestas (sistema y Red Neuronal) para el sistema original en la figura 7.

Después de modificar el sistema y dejar que la red aprendiese la nueva dinámica se volvió a hacer la prueba (esquema paralelo y sin aprendizaje) para el sistema modificado (figura 8).

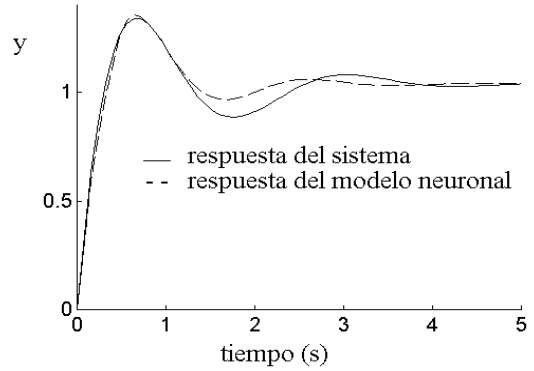


Figura 7. Respuesta al escalón sistema original

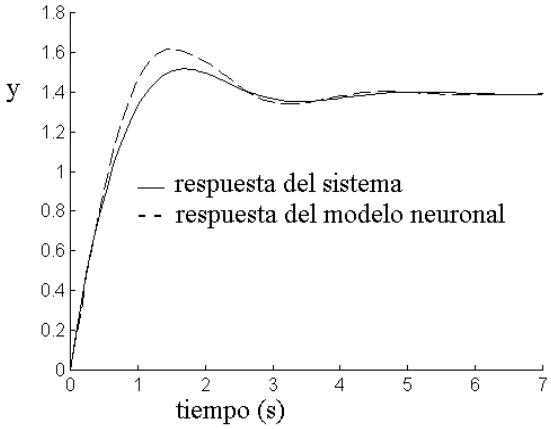


Figura 8. Respuesta al escalón sistema modificado

La segunda señal es una sinusoide de igual magnitud que la señal anterior y de frecuencia 0.1 Hz. La respuesta se presenta en la figura 9 para el sistema original y en la figura 10 para el sistema modificado.

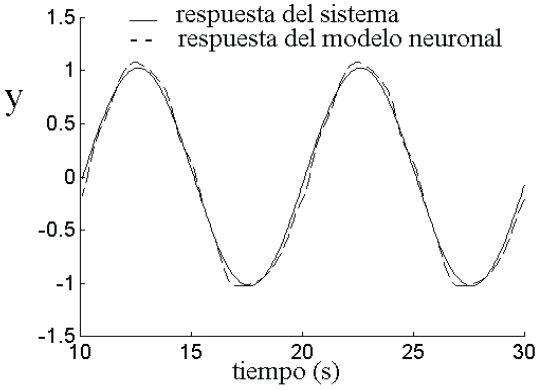


Figura 9. Respuesta sinusoidal sistema original

Es importante remarcar que esta señal no fue utilizada anteriormente, mientras la red aprendía, con lo que se demuestra que el algoritmo tiene una buena capacidad de generalización y que la señal utilizada en el aprendizaje cubre bien la zona apropiada del espacio de entrada.

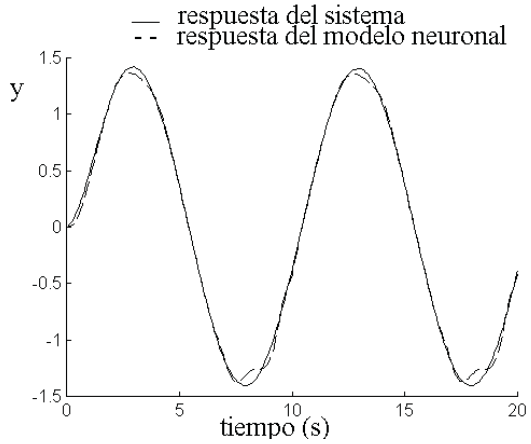


Figura 10. Respuesta sinusoidal sistema modificado

4.2 Sistema No-Lineal con una entrada y una salida

El segundo sistema corresponde a una serie temporal no-lineal que se corresponde con el modelo:

$$y(k) = (0.8 - 0.5 \cdot \mathrm{e}^{-y^2(k-1)})y(k-1) - (0.3 + 0.9 \cdot \mathrm{e}^{-y^2(k-1)})y(k-2) + 0.1\sin(\pi \cdot y(k-1)) + e(k) \quad (20)$$

Donde el ruido *e(k)* es una secuencia gausiana de media cero y varianza 0.01. Este sistema fue utilizado en (Chen y Billings, 1994) para ilustrar el algoritmo de mínimos cuadrados ortogonal (OLS, Orthogonal Least Squares). En este caso nuestra red neuronal tiene dos entradas [*y(k–1), y(k-2)*], una salida, y la correspondiente capa oculta. Empezamos con el sistema en el punto de equilibrio (*y(0)=y(-1)=0*) y dejamos que evolucione durante 200 iteraciones. Esto constituye un ciclo de entrenamiento. Repetimos esta parte hasta que el rRMS para todo un ciclo de entrenamiento (N=200) esté por debajo del límite (≈0.002). Fue necesario realizar este paso 30 veces (i.e. 6000 iteraciones).

La Figura 11 muestra la evolución del error RMS y rRMS durante el entrenamiento, y la figura 12 presenta el crecimiento de la capa oculta en esta fase.

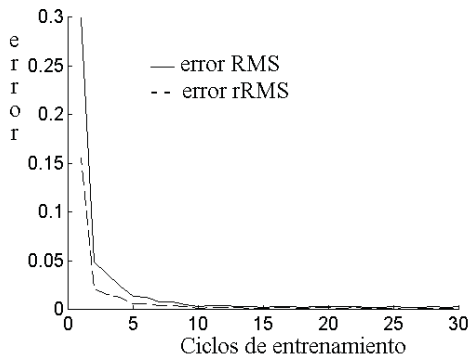


Figura 11. Evolución de los errores

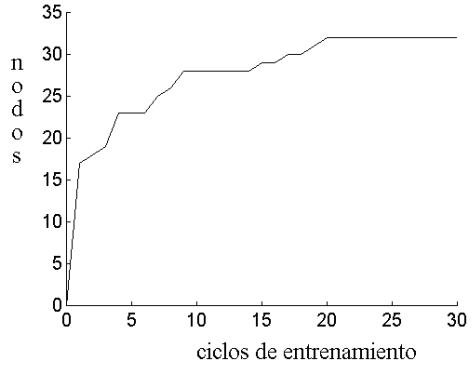


Figura 12. Evolución del tamaño de la capa oculta

En la figura 13 podemos ver la distribución de señal de ruido durante un ciclo de entrenamiento, y las posiciones finales de los nodos RBF en la red entrenada. El centro de cada círculo es la posición del nodo en el espacio de entrada y su radio es proporcional al valor de σ_j de ese nodo.

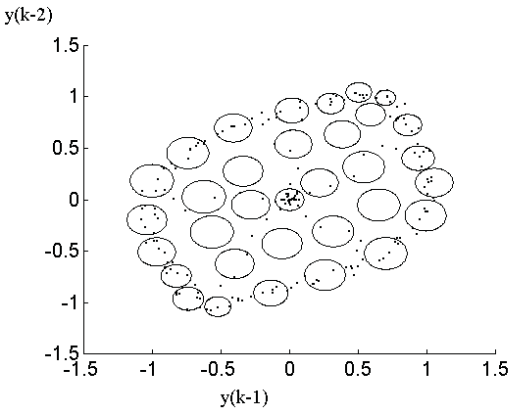


Figura 13. Posición y tamaño de los nodos RBF

La Figura 14 muestra el ciclo límite generado por el sistema cuando empezamos con condiciones iniciales *y(0)=y(-1)=0.1*, y sin ruido *e(t)*. La figura 15 presenta el ciclo límite generado por la red neuronal entrenada en las mismas circunstancias y por supuesto utilizando el esquema paralelo.

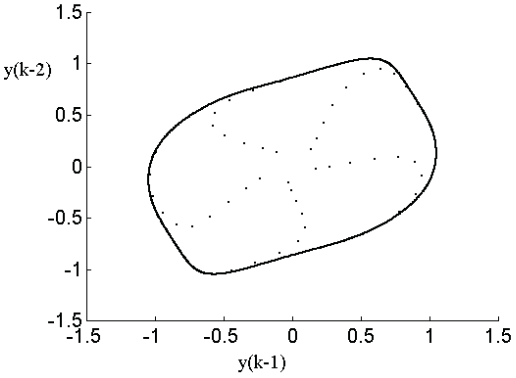


Figura 14. Ciclo límite generado por el sistema

Como hemos explicado anteriormente, nuestro algoritmo ajusta el valor de σ para cada nodo de acuerdo a la dificultad de la zona del espacio de entrada donde es ubicado.

En este caso podemos ver como existen más nodos con pequeño σ_j en las zonas superior derecha e inferior izquierda, donde los cambios son más bruscos (más difícil).

La Figura 16 muestra la respuesta temporal del sistema y de la red neuronal en esas circunstancias. Como se puede ver el modelo neuronal aproxima bien la dinámica del sistema.

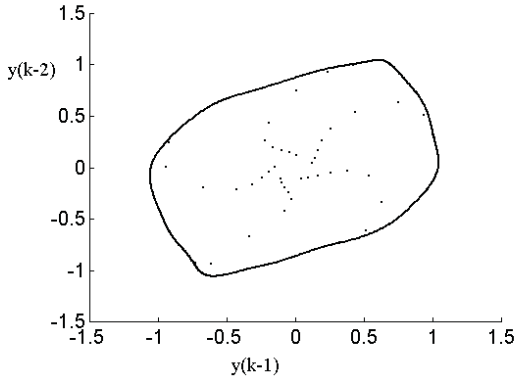
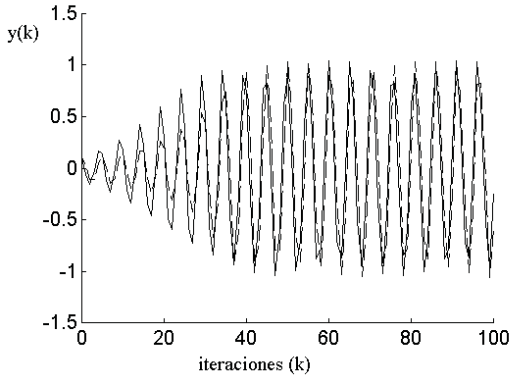


Figura 15. Ciclo límite generado por el modelo neuronal



— salida del sistema --salida de la red

Figura 16. Respuestas del sistema y de la Red

4.3 Sistema No-Lineal con varias entradas y salidas MIMO

El tercer sistema es no-lineal de tipo MIMO definido por

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} \dfrac{y_1(k)}{1+y_2^2(k)} \\ \dfrac{y_1(k)\cdot y_2(k)}{1+y_2^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \quad (21)$$

Este sistema fue utilizado en (Narendra y Parthasarathy, 1989). El modelo neuronal tiene en este caso cuatro entradas, dos salidas, y la capa oculta. La red fue entrenada on-line siguiendo el esquema serie-paralelo. Las señales de entrada u_1 y u_2 fueron dos secuencias diferentes de pulsos de magnitud y frecuencia aleatorias. La Figura 17 muestra la evolución de los errores RMS y rRMS durante la fase de aprendizaje, mientras que la figura 18 presenta la evolución en tamaño de la capa oculta.

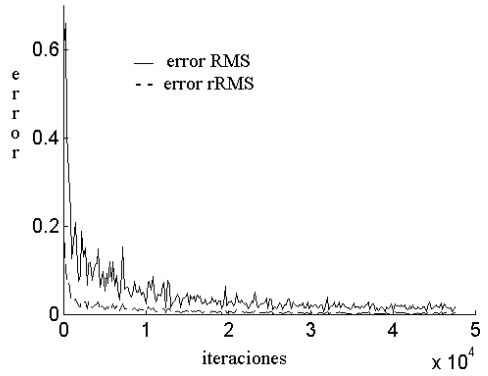


Figura 17. Evolución de los errores RMS y rRMS para el tercer sistema de prueba

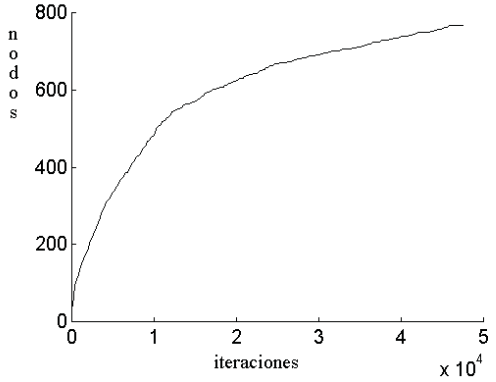
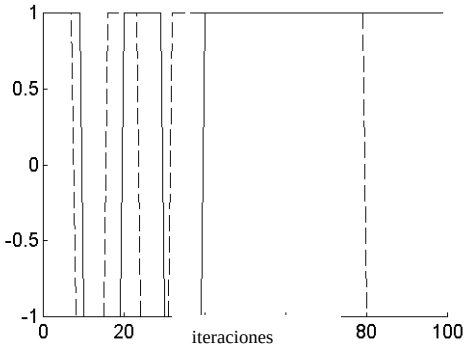


Figura 18. Evolución del tamaño de la capa oculta en la red RBF para el tercer sistema



— u_1 --u_2

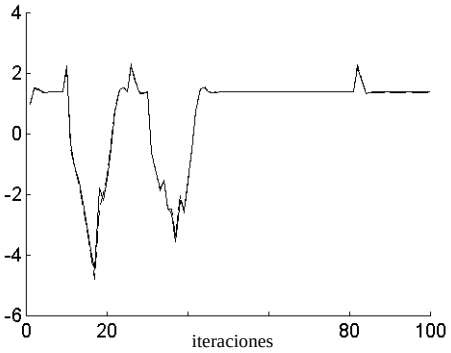
Figura 19. Señales de entrada para entrenamiento

Para observar la precisión del modelo neuronal comparamos la respuesta del sistema y del modelo en dos casos. Primero utilizamos como señales de entrada al sistema las dos secuencias de escalones presentados en la figura 19. La figura 20 muestra la primera salida y_1 del sistema y del modelo neuronal mientras que la figura 21 presenta la segunda salida y_2.

Como se puede observar en las figuras la respuesta del modelo es casi perfecta. Hay que tener en cuenta que las señales de entrada son muy similares a las utilizadas en la fase de entrenamiento.

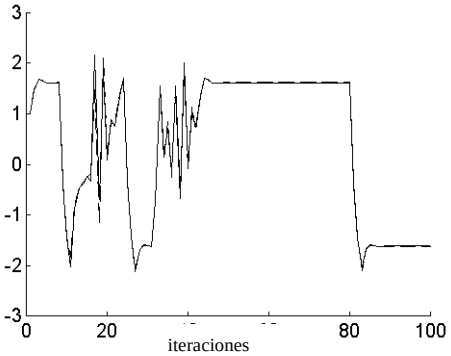
En la segunda prueba utilizamos las señales de entrada que se muestran en la figura 22 y que no se han utilizado para el aprendizaje. Las salidas *y1* y *y2* se presentan en la figura 23 y 24 respectivamente.

En este segundo caso, el comportamiento del modelo no es tan bueno como en el caso anterior pero queda claro que la red ha capturado la dinámica fundamental del sistema.



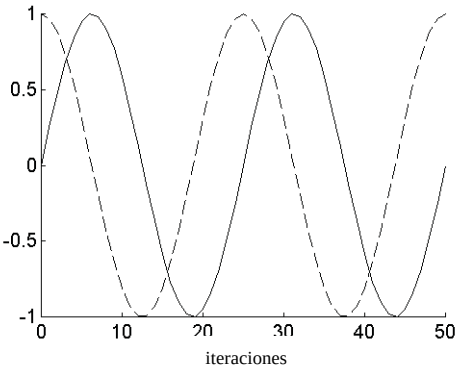
— salida del sistema --salida de la red

Figura 20. Primera salida (y_1) del tercer sistema de prueba



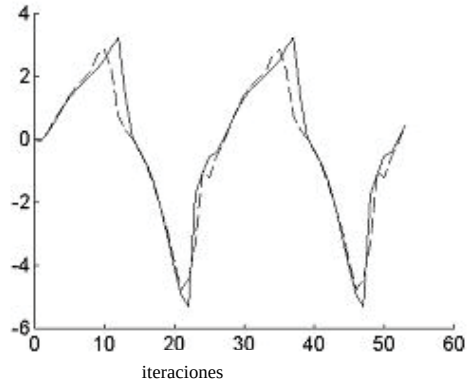
— salida del sistema --nn salida de la red

Figura 21. Segunda salida (y_2) del tercer sistema de prueba



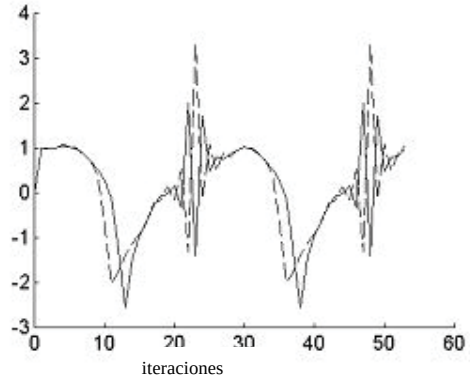
— u_1 --u_2

Figura 22. Señales de entrada para la segunda prueba en el tercer sistema



— respuesta del sistema --respuesta de la red

Figura 23. Primera salida (y_1)



— respuesta del sistema -- respuesta de la red

Figura 24. Segunda salida (y_2)

5. CONCLUSIONES YTRABAJO FUTURO

Estos primeros resultados obtenidos con nuestra red dinámica RBF son muy prometedores. El análisis de las gráficas de error demuestra un buen comportamiento general, con alta capacidad de aprendizaje de dinámicas complejas y adaptabilidad a cambios en el sistema identificado.

La aplicación de la arquitectura propuesta como identificador en un lazo de control avanzado (esquema serie-paralelo) permite trabajar con sistemas que varíen en el tiempo. Una gran ventaja sobre otros tipos de arquitecturas.

Hay que tener en cuenta que el artículo describe los primeros pasos en la validación de la nueva arquitectura. Los resultados obtenidos son buenos, pero queda mucho trabajo por hacer.

Nuestro próximo paso en el desarrollo del esquema propuesto es un estudio de sensibilidad a los parámetros de entrenamiento. En las pruebas realizadas hasta ahora se han utilizado valores de α, a_{min}, E_t, y γ en los rangos citados anteriormente. Es necesario saber de qué manera la selección de esos valores afecta a la capacidad de aprendizaje, velocidad de entrenamiento y adaptación, estabilidad en la topología de la red, e inmunidad al ruido.

A partir de ese análisis se decidirá sobre la implementación de "mecanismos de seguridad" que garanticen la estabilidad de la red y limiten su tamaño.

Otro aspecto que necesita más trabajo es la utilización de la arquitectura con sistemas realmente desconocidos. En los ejemplos presentados el número de entradas de la red neuronal se ha seleccionado a partir del conocimiento del sistema a identificar, aunque no siempre se ha utilizado exactamente (primer ejemplo, sistema SISO). La cuestión a abordar es cómo seleccionar el número de retardos de las señales de entrada y salida a utilizar cuando no se conoce el sistema a identificar.

REFERENCIAS

Chen S., S.A. Billings y P.M. Grant (1990). Nonlinear system identification using neural networks. *International Journal of Control*, vol. **51**(6), 1191-1214.

Chen, S. y S.A. Billings (1994). Neural Networks for Nonlinear Dynamic System Modelling and Identification. En: *Advances in Intelligent Control* (C.J. Harris (ed.)) 85-112. Taylor & Francis, London.

Chi S.R., R. Shoureshi y M. Tenorio (1990). Neural networks for system identification. *IEEE Control Systems Magazine* **10**, 31-34.

Goodwin G.C. y K.S. Sin (1984) *Adaptive filtering prediction and control.* Prentice-Hall, Englewood Cliffs, NJ.

Kuschewski G.J, S. Hui y S.H. Zak (1993). Application of feedforward neural networks to dynamical system identification and control. *IEEE Trans. Control Systems Technology* **1**(1), 37-49.

Le Cun, Y. (1985) Une procedure d'aprentissage pour reseau a sequil assymetrique. *Proceedings of Cognitiva* **85**, 599-604.

Li, Y., N. Sundararajan, y P. Saratchandran (2000) Analysis of Minimal Radial Basis Function Network Algorithm for Real-Time Identification of Nonlinear Dynamic Systems. *IEE Proc. on Control Theory and Applications* **147**(4), 476-484.

Narendra, K.S. y K. Parthasarathy (1989). *Backpropagation in dynamical systems containing neural networks*. Technical Report 8905, Centre for Systems Science, Department of Electrical Engineering, Yale University, New Haven, CT.

Narendra K.S. y K. Parthasaraty (1990). Identification and Control of Dynamical Systems using Neural Networks. *IEEE Transactions on Neural Networks* **1**(1), 4-27.

Obradovic, D. (1996). On-Line Training of Recurrent Neural Networks with Continuous Topology Adaptation. *IEEE Trans. on Neural Networks* **7**(1), 222-228.

Panchapakesan, C., y M. Palaniswami (2002). Effects of Moving the Centres in an RBF Network. *IEEE Transactions on Neural Networks* **13**(6), 1299-1307.

Parker, D. B. Learning logic (1985). En: *Technical Report TR-47*, Massachusetts Institute of Technology, Cambridge, MA.

Peng, H. et al (2004) RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process. *Control Engineering Practice* **12** (2), 191-203.

Poggio T. y F. Girosi. (1989). A Theory of Networks for Approximation and Learning. En: *A.I. Memo* No. 1140. Artificial Intelligence Laboratory, M.I.T.

Rumelhart, D.E., G.E. Hinton, y R.J. Williams (1986). Learning internal representations by error propagation. En: ., *Parallel Distributed Processing: Explorations in the microstructure of cognition. D.E. Rumelhart and J.L. McClelland,*

eds Vol 1, Foundations. Cambridge, MA: Bradford Books/ MIT Press

Sanner R.M. y J.E. Slotine (1991) Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks. En: *Proceedings 30th Conference on Decision and Control*. Brighton, England.

Sjöberg, J., Q. Zhang, L. Ljung et al (1995). Nonlinear Black-box Modelling in System Identification: a Unified Overview. *Automatica* **31**(12), 1691-1724.

Werbos P.J. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. En: *Ph. D. dissertation.*, Harvard University, Cambridge, MA.

Widrow B. y S.D. Stearns (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Zemouri, R., D. Racoceanu y N. Zerhouni (2003). Recurrent radial basis function network for time-series prediction. *Engineering Applications of Artificial Intelligence* **16**, (5-6), 453-463.