Redes Neuronales para identificación y predicción de series de tiempo

Adolfo González Yunes[1], Miguel A. Ávila Álvarez[1], Eduardo Gómez Ramírez[1], Oriol Mulet[2], Ferran Mazzanti[2], Xavier Vilasis-Cardona[2]
[1]Laboratorio de Investigación y Desarrollo de Tecnología Avanzada, Universidad La Salle Benjamín Franklin 47, Col. Hipódromo Condesa, México, D.F., 06170
e-mail: <agonz@ci.ulsa.mx> <maavila@helios.lci.ulsa.mx> <egomez@ci.ulsa.mx>.
[2]Departament d'Electronica - Enginyeria La Salle - Universitat Ramon Llull
Pg. Bonanova 8 08022 Barcelona - España.
e-mail: <mazzanti@salleURL.edu> <xvilasis@salleURL.edu>.

RESUMEN

La predicción de series de tiempo es un área que ha llamado gran atención debido a la gran cantidad de aplicaciones en control, economía medicina, entre otras. En este trabajo se presenta algunos de los algoritmos de redes neuronales artificiales que han mostrado mejores resultados en este campo. Se presenta la aplicación en la predicción de la serie de manchas solares como los datos estándar para que pueda ser comparado con otros algoritmos reportados.
Palabras clave: Redes Neuronales, predicción de series de tiempo, redes neuronales polinomiales, perceptrones, multicapa, redes de Elman.

ABSTRACT

The Area of Forecasting Time Series has grown in the last years due to the great number of applications in control, economy, medicine, etc. In this paper we present some algorithms of Artificial Neural Networks that has shown good results to predict time series. We use like standard example the sunspots forecasting to compare with other algorithms.
Keywords: Neural networks, time series prediction, polynomial neural networks, multilayer perceptrons, Elman networks.

1 INTRODUCCIÓN

La identificación de sistemas es una de las piezas fundamentales de la teoría de control. En efecto, el diseño de cualquier controlador empieza por disponer de un modelo matemático de la planta, o al menos del conocimiento del comportamiento dinámico de las variables involucradas. La calidad del controlador va a depender de la precisión del modelo, principalmente cuando se busca la convergencia de un controlador adaptable. El modelo se suele obtener a partir de las ecuaciones físicas del proceso, aunque a menudo sucede que éstas no describen el sistema con la precisión deseada en todo su rango de funcionamiento. También puede suceder que alguno de los parámetros del modelo sea desconocido, o incluso no existen ecuaciones conocidas para el sistema que se debe tratar. La identificación de un sistema consiste entonces en ajustar su respuesta a un modelo funcional a partir de datos experimentales.

Para sistemas lineales existe una amplia teoría desarrollada (1, 2), pero empiezan a aparecer dificultades al entrar en el dominio de los sistemas no lineales. En este terreno las redes neuronales artificiales (3, 4) resultan una herramienta muy útil para tales propósitos (5, 6),

básicamente por dos motivos: primero, porque se puede demostrar que las redes neuronales artificiales son aproximadores universales (7) de funciones tal como pueda serlo una serie de Fourier, por ejemplo. Segundo, porque la identificación de un sistema se pueda interpretar como un problema de aprendizaje supervisado: se dispone de un conjunto de muestras que la red debe reproducir tan fielmente como sea posible.

El objeto de este artículo es ilustrar el uso de las redes neuronales artificiales para un caso particular de identificación de sistemas como es la predicción de series de tiempo. En particular se utilizará una de las series que más a menudo se usa como banco de pruebas: la serie del Número de Wolff que indica la actividad solar y que se conoce comúnmente como la serie de manchas solares. Sobre esta serie, por un lado, se ilustrará el uso de algunos de los tipos de redes más comunes usados en identificación como son los perceptrones multicapa (8) y las redes de Elman modificadas (12), mientras que por otro, se prueba la eficiencia de las redes polinomiales de arquitectura adaptable (26).

Para explicar varias de las metodologías existentes en el área para la identificación de series de tiempo se estructuró el artículo de la siguiente forma: La sección 2 es una breve introducción a identificación de sistemas y la predicción de series de tiempo. En la sección 3 se describen los tres algoritmos de redes Neuronales utilizados: perceptrón multicapa, redes de Elman modificadas y, red neuronal artificial polinomial (RNAP). En la sección 4 se describe lo que son las manchas solares y la serie de tiempo generada. En la siguiente sección se muestran los resultados con cada una de estas redes y por último las conclusiones de este artículo.

2. IDENTIFICACIÓN DE SISTEMAS Y PREDICCIÓN DE SERIES DE TIEMPO

Tal como se ha mencionado en la introducción, identificar un sistema consiste en ajustar la respuesta de éste a un modelo matemático funcional a partir de datos experimentales. Dependiendo del enfoque es posible realizar esta tarea de dos formas:

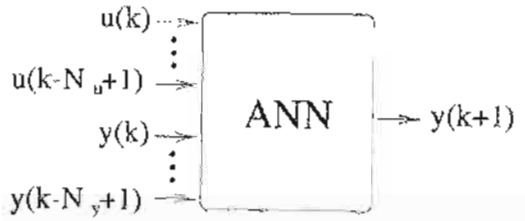

Figura 1: Esquema de la identificación entrada/salida con una red neuronal.

Un enfoque consiste en estudiar la planta como un sistema entrada/salida, es decir, medimos la respuesta *y(t)* a una entrada determinada *u(t)* e intentamos encontrar la dependencia funcional entre ellas. Para un sistema causal en tiempo discreto, se tiene,

$$y(k+1) = h(u(k), u(k-1), u(k-n_1); y(k), y(k-1), y(k-n_2)),$$

(Ec. 1)

donde, por simplicidad se han eliminado las dependencias de variables externas que no se pueden controlar. Así formulado, el problema consiste en aproximar la función $h(\cdot)$. En términos lineales, esto es equivalente a encontrar la función de transferencia.

El segundo enfoque pretende, por otro lado, hallar la dinámica interna de la planta a través de sus variables de estado *x(t)*. La salida del sistema *y(t)* es una función de estas variables. Para un sistema causal discreto, se tiene, simplificando las dependencias posibles para la salida,

$$x(k+1) = f(x(k), x(k-1), \ldots; u(k), u(k-1), \ldots),$$
$$y(k) = g(x(k), u(k)),$$

(Ec. 2)

donde se ha realizado la misma simplificación que en la Ec. 1.

En este marco, la tarea de las redes neuronales es aproximar las funciones $f(\cdot)$, $g(\cdot)$ o $h(\cdot)$, dependiendo del tipo de red empleada.

Si se utilizan redes de alimentación hacia adelante (*feed forward*), como son los perceptrones

multicapa (PM) o las redes neuronales artificiales polinomiales (RNAP), se optará por hacer una identificación del tipo entrada/salida. Así, se buscará ajustar una función de la forma,

$$y(k+1) = \hat{h}(u(k), u(k+1), \ldots, u(k-N_u+1); y(k), y(k-1), \ldots, y(k-N_y+1))$$

(Ec. 3)

Se construirá una red con N_u+N_y neuronas de entrada en las cuales se dispondrá de los valores $u(k),\ldots,u(k-N_u+1),y(k),\ldots,y(k-N_y+1)$ y una neurona de salida en la cual se espera obtener $y(k+1)$ (ver Figura 1). Se organizarán entonces los datos obtenidos experimentalmente sobre la respuesta del sistema para construir el conjunto de entrenamiento:

$$\{u(k),\ldots,u(k-N_u+1), y(k),\ldots,y(k-N_y+1); y(k+1)\}$$

Con estos valores se va a entrenar la red con los algoritmos apropiados.

El interés en usar redes recurrentes reside en la posibilidad de emular la dinámica del sistema con su propia dinámica interna. Se busca que la red encuentre una representación en variables de estado del sistema a partir de los estados de las neuronas ocultas. Idealmente, sólo se necesita introducir en la red el valor de las entradas actuales $u(k)$ para poder recuperar la salida futura $y(k)$ en las neuronas de salida de la red. En la práctica resulta a menudo más eficiente informar a la red del valor de entradas y salidas anteriores. Del mismo modo que para las redes de alimentación hacia adelante con los datos experimentales se construye el conjunto de entrenamiento y se entrenará la red.

En general, el entrenamiento de redes recurrentes suele ser más lento y difícil que el de las redes con alimentación hacia adelante, aunque ello se compensa por el mejor ajuste que potencialmente pueden realizar del sistema.

Un caso particular de identificación de sistemas es la predicción de series de tiempo. Una serie de tiempo no es más que un conjunto de datos medidos a intervalos regulares. Dentro de esta amplia definición se pueden considerar los fenómenos más dispares, desde las cotizaciones bursátiles diarias hasta el consumo horario eléctrico de una ciudad, el importe que paga diariamente un cajero automático o el número de personas por hora que cruzan una determinada calle. Vistos estos ejemplos, el interés de realizar predicciones sobre estas series es evidente. Desde el punto de vista de la identificación de sistemas, se puede interpretar una serie de tiempo como la salida de un sistema con el que no se cuenta con su entrada (y por ello se desconocen sus $u(k)$). Se dispone únicamente de los valores anteriores de la serie para predecir los valores siguientes. En algunos casos, se intenta buscar variables externas que puedan influir de forma relevante que están correlacionadas con la salida: por ejemplo, el consumo de gas puede depender de la temperatura, ya que en función de ella se encenderán o no los calefactores.

Existe abundante literatura sobre predicción de series de tiempo empleando métodos lineales, (9, 10) aunque las redes neuronales, por su capacidad de aprendizaje se están erigiendo en útiles aliados para estas tareas. Tomando una ventana de datos, se alimenta a las redes, sean las de alimentación hacia adelante o recurrentes, y se obtiene la salida estimada:

$$\hat{y}(k+l) = h(y(k), y(k-1), \ldots)$$

(Ec. 4)

Como caso particular de identificación, se puede interpretar el valor a predecir directamente como una función de los valores anteriores, tal como indica la Ec. 4, o como la medida del estado de un sistema:

$$x(k+1) = f(x(k), x(k-1), \ldots),$$
$$y(k) = g(x(k))$$

(Ec.5)

Operando de la misma manera que ha descrito para la identificación de sistemas, se pueden usar redes de alimentación hacia adelante o recurrentes, tal como se ilustrará en las próximas secciones.

3. EJEMPLOS DE REDES USADOS EN LA IDENTIFICACIÓN DE SISTEMAS

A continuación se describirán dos de los tipos

de redes neuronales artificiales más utilizadas en la identificación de sistemas. Se eligió, para ilustrar el desarrollo de la sección 2, un tipo de red de alimentación hacia adelante como es el perceptrón multicapa, y un tipo de red recurrente, como es la red de Elman modificada.

3.1 Perceptrón multicapa

Este es sin duda el tipo de red más famoso y de hecho, el responsable de la popularidad de las redes neuronales a partir de mediados de los años ochenta. Esto se debe a la simplicidad de su algoritmo de aprendizaje supervisado, el conocido propagación hacia atrás (backpropagation) (11).

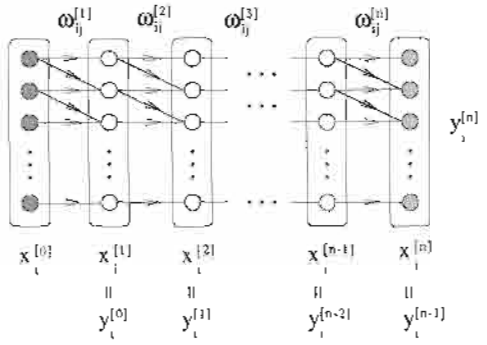

Figura 2: Perceptrón multicapa.

Como su nombre indica, el perceptrón multicapa tiene sus neuronas organizadas por capas, de acuerdo a la estructura que se muestra en la Figura 2. Su número es variable dependiendo de la funcionalidad de la red, pero en general se distinguen los tipos siguientes:

- una capa de entrada, donde se alimenta a los datos a la red (capa 0 del dibujo),
- capas ocultas, cuyo número es variable,
- una capa de salida, donde se lee el resultado del cálculo de la red (capa *n* de la figura).

De acuerdo con el esquema de la figura, en general se designa $x_i^{[k]}$ al valor que toma la neurona *i*-ésima de la capa *k*-ésima, mientras que $y_i^{[k]}$ es el valor que transmite a las neuronas de la capa siguiente. Como la señal se propaga exclusivamente hacia adelante, la salida $y_i^{[k]}$ coincide con la entrada $x_i^{[k+1]}$. La relación entre la señal que llega a una neurona y su valor de salida se establece a través de la denominada *función de activación f(·)*, que generalmente es no-lineal,

$$y_i^{[k]} = x_i^{[k+1]} \equiv fl\left(\sum_j \omega_{ij}^{[k]} x_j^{[k]}\right)$$

(Ec. 6)

donde $\omega_{ij}^{[k]}$ es el peso que la neurona *i* de la capa *k* con el resto de neuronas de la capa anterior.

Tal como se ha mencionado, en este tipo de redes, se suele emplear como algoritmo de entrenamiento, es decir de adaptación de pesos, el algoritmo de propagación hacia atrás. Este consiste en minimizar el error cuadrático medio del conjunto de muestras comparando la salida esperada con la obtenida por la red:

$$E = \frac{1}{2N}\sum_{\mu}\left(y_i^{[n],\mu} - \tilde{y}_i^{\mu}\right)^2$$

(Ec. 7)

siendo *N* el número de muestras, etiquetadas por el índice μ, $\tilde{y}_i^{\mu}$ la salida esperada y $y_i^{[n],\mu}$ la salida de la red. La minimización se lleva a cabo con técnicas de gradiente inverso aplicadas a los pesos de cada capa:

$$\Delta w_{ij}^{[k]} = -\eta \frac{\partial E}{\partial w_{ij}^{[k]}}$$

(Ec.8)

donde η es el parámetro de aprendizaje. Sobre esta base, suelen aplicarse métodos de aceleración, como por ejemplo añadir un término de momento, que recuerda la dirección escogida en la anterior actualización de los pesos:

$$\Delta w_{ij}^{[k]}(t) = -\eta \frac{\partial E}{\partial w_{ij}^{[k]}}(t) + \alpha \Delta w_{ij}^{[k]}(t-1)$$

(Ec. 9)

Con esto se consigue, además de acelerar la convergencia del algoritmo de aprendizaje, evitar algunos mínimos locales poco profundos de la función de error.

Se toma en cuenta que la Ecuación 6 relaciona el valor de las neuronas de la capa *k* con los de la capa *k-1*, y que éstas a su vez se relacionan con las de la capa *k-2* y así sucesivamente hasta llegar a las neuronas de la capa de entrada. Entonces es posible expresar la función de error en términos de los pesos de la capa respecto a la que se quiere derivar.
Entonces:

$$E = \frac{1}{2N}\sum_{\mu,i}\left[\tilde{y}_i^{\mu} - f\left(\sum_{j_1}\omega_{i,j_1}^{[n]} f\left(\sum_{j_2}\omega_{j_1,j_2}^{[n-1]} f(\beta)\right)\right)\right]^2$$

$$\beta = \cdots f\left(\sum_{j_k}\omega_{j_{k-1},j_k}^{[k]} x_{j_k}^{[k]}\right)$$

(Ec.10)

A partir de esta expresión se puede derivar la regla de actualización de los pesos de la capa k empleando las ecuaciones 8 y la regla de la cadena. Para los pesos que conectan con la capa de salida, se obtiene:

$$\frac{\partial E}{\partial \omega_{ij}^{[n]}} = \sum_{\mu} \delta_i^{[n],\mu} x_j^{[n-1],\mu}$$

(Ec. 11)

donde $\delta_i^{[n],\mu}$ es una cantidad característica de las ecuaciones del algoritmo de propagación hacia atrás

$$\delta_i^{[n],\mu} = (y_i^{\mu} - \tilde{y}_i^{\mu}) f(h_i^{[n],\mu})$$

(Ec.12)

y $h_i^{[n],\mu}$ es el campo local

$$h_i^{[n],\mu} = \sum_k \omega_{ik}^{[n]} x_k^{[n],\mu}$$

(Ec.13)

Procediendo de forma análoga, la regla de actualización de los pesos de la penúltima capa resulta ser:

$$\frac{\partial E}{\partial \omega_{ij}^{[n-1]}} = \sum_{\mu} \delta_i^{[n-1],\mu} x_j^{[n-1],\mu}$$

(Ec. 14)

donde ahora

$$\delta_i^{[n-1],\mu} = \sum_j \delta_i^{[n],\mu} \omega_{ij}^{[n]} f(h_j^{[n-1],\mu})$$

(Ec. 15)

y

$$h_j^{[n-1],\mu} = \sum_k \omega_{ik}^{[n-1]} x_k^{[n],\mu}$$

(Ec. 16)

Como puede observarse a partir de estos resultados, existe cierta recurrencia que permite obtener la expresión general de las derivadas de la función de costo respecto a los pesos de cada capa. El resultado general es que la actualización de los pesos de la capa *k*-ésima se establece de acuerdo a:

$$\frac{\partial E}{\omega_{ij}^{[k]}} = \sum_{\mu} \delta_i^{[k],\mu} x_j^{[k],\mu}$$

(Ec. 17)

donde el valor de $\delta_i^{[k],\mu}$ se halla a través de la relación

$$\delta_i^{[k],\mu} = \sum_j \delta_j^{[k+1],\mu} \omega_{ij}^{[k]} f(h_j^{[k-1],\mu})$$

(Ec. 18)

en términos de los campos locales

$$h_i^{[k]} = \sum_j \omega_{ik}^{[k]} x_j^{[k],\mu}$$

(Ec. 19)

En conclusión y tal como puede observarse, el cálculo de $\delta_i^{[k],\mu}$ requiere del uso de $\delta_i^{[k+1],\mu}$, mientras que ésta a su vez se calcula partiendo de $\delta_i^{[k+2],\mu}$ y así hasta llegar a $\delta_i^{[n],\mu}$, que se obtiene a partir de la salida predicha por la red,

el valor real de tal salida y el campo local en la última capa. Así pues, en este algoritmo la actualización de los pesos se realiza en sentido inverso al de propagación de la señal, es decir que primero se evalúa la variación de los pesos de la última capa, luego la variación de los pesos de la penúltima capa, y así hasta llegar a la variación de los pesos de la capa de entrada: es por ello que el algoritmo se denomina propagación hacia atrás.

3.2 Redes de Elman modificadas

La red de Elman modificada es un tipo de red neuronal con una topología muy particular y que puede entenderse como un modelo híbrido de red de alimentación hacia adelante y red recurrente (12, 6). Consta de una capa de entrada, una capa intermedia, una capa de salida y una capa de contexto tal como se muestra en la Figura 3. Tanto entre la capa de entrada y la oculta como entre la capa oculta y la de salida, la señal se propaga hacia adelante como en una red de alimentación hacia adelante, mientras que la conexión entre la capa oculta y la capa de contexto es bidireccional haciendo que la red sea recurrente. Por otro lado, no existe conexión directa entre la capa de contexto y las otras capas.

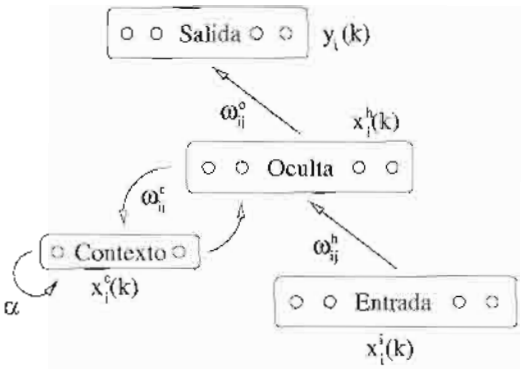

Figura 3: Red de Elman modificada

Las redes de Elman modificadas son útiles en teoría de control, debido a que conjugan satisfactoriamente los beneficios de las redes de alimentación hacia adelante y los de las redes recurrentes. Lo que se consigue con ello es disponer de un sistema de menores dimensiones pero con una capacidad computacional similar a la que se obtiene con otras redes (un perceptrón multicapa por ejemplo) con un número mayor de neuronas. Sin embargo, para poder especificar la dinámica de funcionamiento de esta red es necesario introducir previamente la nomenclatura que especifique cada parámetro de la misma. Para ello se designa $y_i(k)$ al valor de la neurona *i-ésima* en la capa de salida, $x_j^a(k)$ el valor que toma la neurona *j*-ésima de la capa *a* (este índice varía entre *i*, *o*, *h* o *c* para indicar *input*, *hidden* o *context*), y ω_{ij}^a el valor que toma el peso que modera la interacción entre las neuronas *i* y *j* de la capa *a* donde *a* toma los mismos valores de antes. Si adicionalmente se denomina $f(\cdot)$ a la función de activación de cada neurona, el conjunto de ecuaciones que determinan los valores que toman éstas en el equilibrio es el siguiente:

$$y_i(k) = f\left(\sum_j \omega_{ij}^o x_j^h(k)\right)$$

$$x_i^h(k) = f\left(\sum_j \omega_{ij}^h x_j(k) + \sum_j \omega_{ij}^c x_j^c(k)\right)$$

$$x_i^c(k+1) = \alpha x_i^c(k) + x_i^h(k)$$

(Ec. 20)

Así pues, tal como puede observarse el valor de las neuronas de salida depende del valor de las neuronas de la capa oculta, mientras que éstas últimas dependen de forma compleja de los valores de las neuronas de entrada y de las de contexto, siendo éstas a su vez dependientes del valor de las propias neuronas de la capa oculta. Esto último pone de manifiesto el carácter recurrente de la red. El problema por tanto es *autoconsistente* y el valor que toma cada neurona en el equilibrio es precisamente aquel que hace que el conjunto satisfaga la Ecuación 12. Nótese que en realidad estas ecuaciones no son todo lo general posible ya que la función de activación que determina el valor de $y_i(k)$ y de $x_i^h(k)$ es la misma, y que la relación que determina el valor de $x_i^c(k)$ es lineal.

El hecho de que el número de neuronas necesarias para igualar las capacidades computacionales de otras redes sea menor en la red de Elman queda compensado por una manifiesta mayor complejidad en las ecuaciones que determinan la reglas de aprendizaje. Dichas reglas pueden hallarse procediendo de forma análoga

a la descrita en el apartado anterior pero atendiendo a las peculiaridades especificas que conlleva la topología de esta red. Así pues, un poco de álgebra más o menos tediosa proporciona, a las siguientes reglas de actualización, los pesos:

Conexiones con la capa de salida

$$\Delta\omega_{ij}^{o} = \eta \frac{1}{2} \sum_{k} \delta_i^{o}(k) x_j^{h}(k)$$

(Ec. 21)

donde:

$$\delta_j^{o}(k) = (y_i^{*}(k) - y_i(k)) f(\sum_{l} \omega_{il}^{o} x_l^{h}(k))$$

(Ec. 22)

Conexiones con la capa oculta

$$\Delta\omega_{ij}^{h} = \eta \sum_{k} \delta_i^{h} x_j(k)$$

(Ec. 23)

donde:

$$\delta_i^{h}(k) = \sum_{l} \delta_l^{o}(k) \omega_l^{o} f(\sum_{n} \omega_{in}^{h} x_n(k) + \sum_{n} \omega_{in}^{c} x_n^{c}(k)) x_j(k)$$

(Ec. 24)

Conexiones con la capa de contexto

$$\Delta\omega_{ij}^{c} = \eta \sum_{k} \left[\delta_i^{h}(k) x_j^{c}(k) + \sum_{l,m} \delta_l^{h}(k) \omega_{lm}^{c} \frac{\partial x_m^{c}(k)}{\partial \omega_{ij}^{c}} \right]$$

(Ec. 25)

donde:

$$\frac{\partial x_m^{c}(k)}{\partial \omega_{ij}^{c}} = \alpha \frac{\partial x_m^{c}(k)}{\partial \omega_{ij}^{c}} + \ldots..$$

$$\ldots. + f\left(\sum_{n} \omega_{mn}^{h} x_n(k-1) + \sum_{n} \omega_{mn}^{c} x_n^{c}(k-1) \right)$$

$$\times \left[\sum_{n} \omega_{mn}^{c} \frac{\partial x_n^{c}(k-1)}{\partial \omega_{ij}^{c}} + \delta_{im} x_j^{c}(k-1) \right]$$

(Ec. 26)

Tal como puede observarse, el sistema de ecuaciones obtenido es complejo y por ello en la práctica deben usarse con cuidado. En términos generales el proceso a seguir para entrenar una red de Elman modificada es el siguiente: partiendo de una elección inicial adecuada de los pesos (que pueden ser valores aleatorios o que por el contrario pueden estar guiados por criterios externos acerca del problema que se quiere resolver) y de un valor inicial aleatorio o nulo de las neuronas de la capa de contexto, se calcula el valor de las $\Delta\omega_{ij}^{o}$ mediante la Ec. 21 y Ec. 22. De la misma forma se evalúan las variaciones de los pesos de la capa oculta y de la capa de contexto. Una vez acabado el ciclo, se actualiza el valor de todos los pesos y se vuelve a iniciar el proceso, que acaba cuando las variaciones de éstos es menor que un cierto error tolerado o bien cuando se ha realizado el conjunto de actualizaciones un número determinado de veces.

3.3 Red neuronal artificial polinomial

La historia de las RNA comienza con el trabajo de McCulloch y Pitts (13), planteando algunas ideas para modelar el sistema neuronal. Estos modelos biológicos han cambiado con los nuevos avances reportados en las neurociencias y la tecnología. Actualmente, se sabe que las conexiones sinápticas no solamente pueden ser modeladas mediante una suma de la ponderación de las entradas (14). Los resultados muestran que algunas neuronas pueden modular, potenciar y ponderar la respuesta de otras neuronas (15). Esto significa que el modelo por neurona pudiera considerarse como una relación de multiplicación o potenciación. En la literatura se pueden encontrar algunos modelos que aprovechan estas ideas como: Las Redes Neuronales Polinomiales (RNP), (Polynomial Neural Networks, PNN) (16), Redes Neuronales de Orden Mayor (Higher Order Neural Networks, HONN) (17) y modelos con interconexiones no lineales. Este tipo de representación no es exclusiva de las RNA y se pueden consultar modelos similares matemáticamente en otras áreas, por ejemplo: el modelo NARMAX (18)(19)(20)(21), Método de grupo para el manejo de datos (Group Method of Data Handling (GMDH)(22)(23) y Aproximaciones Polinomiales (24)(25).

El modelo de RNAP propuesto puede ser descrito como: (27)

$$\hat{y}_k = [\phi(x_{1,k}, x_{2,k} \quad \ldots x_{n_1,k}, \quad x_{1,k-1}, \quad x_{2,k-1}, \quad \ldots \quad x_{n_1,k-n_1}, \quad \ldots \quad y_{k-1}, \quad y_{k-2}, \quad \ldots y_{k-n_2})]_{\phi_{min}}^{\phi_{max}}$$

(Ec. 27)

donde $\hat{y}_k \in \Re$ es el estimado de una función, es decir la salida de la red, $\phi(x,y) \in \Re$ es una función no lineal, $x_i \in X$ son las entradas, $i=1,\ldots,n_i$; n_i=número de entradas, $y_{k-j} \in Y$ son los valores anteriores de la salida, $j=1,..,n_2$, n_1 número de retardos de la entrada, n_2 número de retardos de la salida, X, Y son conjuntos compactos de $\Re$.

La función no lineal $\phi(z)$ está dada por:

$$[\phi(z)]_{\phi_{min}}^{\phi_{max}} = \begin{cases} \phi_{max} & \phi(z) \geq \phi_{max} \\ \phi(z) & \phi_{min} < \phi(z) < \phi_{max} \\ \phi_{min} & \phi(z) \leq \phi_{min} \end{cases}$$

(Ec. 28)

donde ϕ_{max} and ϕ_{min} son los límites máximo y mínimo respectivamente.

Para simplificar el manejo de la notación en las ecuaciones se va a hacer un cambio de variable de tal forma que:

$$z = \{x_{1,k}, x_{2,k}, .., x_{n_1,k}, \ldots\ldots\ldots, y_{k-1}, y_{k-2}, \ldots\ldots, y_{k-n_2}\} = \{z_1, z_2, z_3, \ldots\ldots z_{n_v}\}$$

(Ec. 29)

donde: n_v es el número total de elementos en la descripción z, es decir, el número total de entradas, valores anteriores y valores anteriores de la salida:

$$n_v = n_i + n_1 n_i + n_2$$

(Ec. 30)

Entonces la función $\phi(z) \in \Phi_p$ pertenece a una familia de polinomios Φ_p que pueden ser representados:

$$\Phi_p(z_1, z_2, \ldots, z_{n_v}) = \left\{ \begin{array}{l} \phi(z) : \phi(z) = a_0(z_1, z_2, \ldots, z_{n_v}) + a_1(z_1, z_2, \ldots, z_{n_v}) \\ + a_2(z_1, z_2, \ldots, z_{n_v}) \ldots\ldots + a_p(z_1, z_2, \ldots, z_{n_v}) \end{array} \right\}$$

(Ec. 31)

El subíndice p es la potencia máxima de la expresión polinomial y los términos $a_i(z_1, z_2, \ldots, z_{n_v})$ son polinomios homogéneos de grado total i, para $i=0,\ldots,p$. Cada polinomio homogéneo puede ser definido como:

$$a_0(z_1, z_2, \ldots, z_{n_v}) = w_0$$

$$a_1(z_1, z_2, \ldots, z_{n_v}) = w_{1,1} z_1 + w_{1,2} z_2 + \ldots + w_{1,n_v} z_{n_v}$$

$$a_2(z_1, z_2, \ldots, z_{n_v}) = w_{2,1} z_1^2 + w_{2,2} z_1 z_2 + w_{2,3} z_1 z_3 + \ldots + z_1 z_{n_1} + \ldots z_2^2 + \ldots z_2 z_3 \ldots + w_{2,N_2} z_{n_v}^2$$

$$a_3(z_1, z_2, \ldots, z_{n_v}) = w_{3,1} z_1^3 + w_{3,2} z_1^2 z_2 + w_{3,3} z_1^2 z_3 + w_{3,4} z_1 z_2^2 + w_{3,5} z_1 z_2 z_3 + w_{3,6} z_1 z_3^2 + \ldots + \ldots z_2^3 + \ldots z_2^2 z_3 + \ldots z_2 z_3^2 + \ldots + w_{3,N_3} z_{n_v}^3$$

$$a_p(z_1, z_2, \ldots, z_{n_v}) = w_{p,1} z_1^p + w_{p,2} z_1^{p-1} z_2 + \ldots + \ldots + w_{p,N_p} z_{n_v}^p$$

(Ec. 32)

donde w_{ij} corresponde al peso asociado a cada neurona. El término w_0 corresponde al input bias de la red. Este término tiene el mismo significado que el coeficiente cero de la transformada de Fourier, es decir, obtiene el promedio de la señal que se quiere estimar. El polinomio $a_1(z)$ corresponde únicamente a la ponderación lineal de las entradas. De $a_2(z)$ a $a_p(z)$ se representan los términos de modulación entre las entradas correspondientes y las relaciones de potenciación.

Como se puede observar los términos utilizados a partir de z_i^2 permiten al algoritmo resolver el problema de paridad bidimensional que tenía el perceptrón. Esto es análogo al efecto de tener varias capas en una red neuronal tradicional.

El valor N_i corresponde al número de términos de cada polinomio homogéneo:

$$N_0 = 1, N_1 = n_v, N_2 = \sum_{i=1}^{n_v} i, N_3 = \sum_{s_1=0}^{n_v-1} \sum_{i=1}^{n_v-s_1} i,$$

(Ec. 33)

$$N_p = \underbrace{\sum_{s_{p-2}=0}^{n_{p-1}-1} \cdots \sum_{s_2=0}^{n_3-s_3} \sum_{s_1=0}^{n_2-s_2} \sum_{i=1}^{n_1-s_1} i}_{p-1}$$

La dimensión N_Φ de cada familia Φ_p puede ser obtenida de la siguiente forma.

$$N_\Phi = \sum_{i=0}^{P} N_i$$

(Ec. 34)

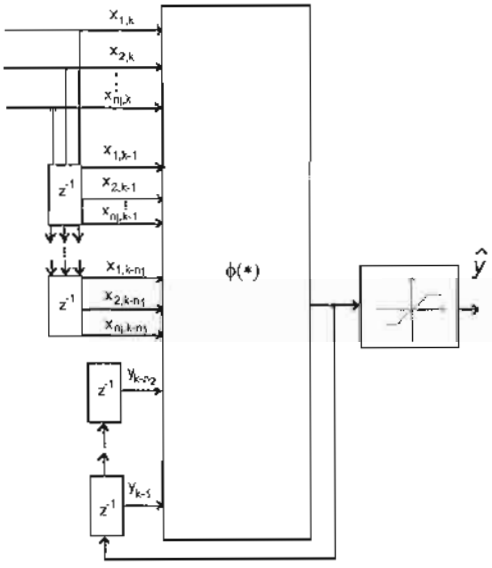

Figura 4: Esquema de RNAP

3.3.1 Aprendizaje de RNAP

Para introducir el aprendizaje en RNAP es necesario, primero, introducir algunos conceptos que serán utilizados en este artículo.

Definición 4.1

El error de aproximación de RNAP se define como:

$$err_n\left(y^n, \phi(z)\right) = \frac{1}{n}\sum_{k=1}^{n}\left(y_k - \phi\left(z_k\right)\right)^2$$

$$= \frac{1}{n}\sum_{k=1}^{n}\left(y_k - \hat{y}_k\right)^2, y^n = \left(y_1, y_2, .., y_n\right)$$

(Ec. 35)

donde y_n es la salida deseada, $\phi(z_k) \in \Phi_p$ y n es el número de puntos.

Definición 4.2

El error óptimo está definido como:

$$opterr_n\left(y^n, \phi(z)\right) = \min_{\phi \in \Phi_p} err_n\left(y^n, \phi(z)\right)$$

$$= err_n\left(y^n, \phi_n^*(z)\right)$$

(Ec. 36)

donde

$$\phi_n^*(z) \in \Phi_p$$

es la estimación óptima de y^n.

Definición 4.3

La RNAP $\phi(z) \in \Phi_n$ aprende uniformemente la salida deseada con precisión ε si

$$err_n\left(y^n, \phi(z)\right) - err\left(y^n, \phi_n^*(z)\right) > \varepsilon = 0 \quad \varepsilon > 0$$

Después de describir estos conceptos ahora el problema del aprendizaje de RNAP consiste en encontrar la estructura de $\phi \in \Phi_n(z)$ que verifica esta desigualdad.

En la siguiente sección se aplica el uso de algoritmo genético para obtener el valor del arreglo W_b^* . Se presenta un algoritmo que obtiene la arquitectura óptima de la red mediante el uso de AG (26). Para lograr esto defínase un vector de componentes $M(z)$ utilizando la simplificación de (29):

$$M(z) = \{z_1, z_2, z_3, \ldots, z_{nv}, z_1^2, z_1 z_2, \ldots$$

$$\ldots, z_{nv}^2, z_1^3, z_1^2 z_2, \ldots, z_{nv}^p\}$$

(Ec. 37)

Entonces la función no lineal $\phi \in \Phi_p(z)$ descrita en la Ec. 31 puede ser representada como:

$$\phi = \langle W', M(z) \rangle \; donde \; W' = W. * W_b^t, \forall w_i^b \in \{0,1\}$$

(Ec. 38)

$$donde \; W = \{w_1, w_2, ..., w_{N_\Phi}\}$$

son los pesos de RNAP, W_b es un vector binario, y N_Φ se obtiene utilizando la Ec. 34.

Definición 4.4

El producto . se define como:*

$$W. * W_b^T = \begin{cases} w_{ij} & if \; w_j^b = 1 \\ 0 & if \; w_j^b = 0 \end{cases}$$

(Ec. 39)

Por ejemplo, para W y W_b como:

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \quad W_b = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

$$W. * W_b^T = \begin{bmatrix} w_{11} & 0 & w_{13} \\ w_{21} & 0 & w_{23} \\ w_{31} & 0 & w_{33} \end{bmatrix}$$

y para el caso vectorial:

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} . * \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T = \begin{bmatrix} w_1 & 0 & w_3 & 0 \end{bmatrix}$$

La Ec. 38 representa que ϕ solamente tiene términos específicos de Φ_α que pueden ser seleccionados por W_b de tal forma que la estructura

$$\phi^*(z) = \langle (W')^*, M(z) \rangle = \langle W. * (W_b^T)^*, M(z) \rangle$$
$$= \langle W, M(z). * (W_b^T)^* \rangle$$

(Ec. 40)

$$\Rightarrow err_n(y^n, \phi(z)) = err_n(y^n, \langle (W'), M(z) \rangle)$$

(Ec. 41)

$$\Rightarrow \underset{W'}{opt}\, err_n(y^n, \phi(z)) = \underset{W'}{opt}\, err_n(y^n, \langle (W')^*, M(z) \rangle)$$

(Ec. 42)

Utilizando la Ec. 37 y Ec. 42, el problema de aprendizaje para una estructura específica puede ser representado por un problema de optimización con los siguientes dos pasos:

$$\min_{W_b} \min_{W \in \Re^N} err_n(y^n, \phi(z))\Big|_{W_b}$$

(Ec. 43)

donde $err_n(y^n, \phi(z))\Big|_{W_b}$

es el error definido en la Ec. 35 para un valor determinado de W_b.

Los valores del parámetro W pueden ser obtenidos utilizando el método de mínimos cuadrados (27):

$$W|_{W_b} = \underset{W \in \Re^N}{\arg\min}\, err_n(y^n, \phi(z))\Big|_{W_b}$$

$$W|_{W_b} = \Gamma^N \sum_{k=1}^{n} y_k(M_b(z_k)) \qquad M_b(z_k) = M. * W_b^T$$

$$\Gamma^N = \left(\sum_{k=1}^{n} M_b(z_k) M_b(z_k)^T \right)^{-1}$$

(Ec. 44)

o en su forma recurrente

$$(W|_{W_b})_k = (W|_{W_b})_{k-1} + \Gamma_k M_b(z_k)^T \left[y_k - (W|_{W_b})_{k-1} M_b(z_k)^T \right]$$

$$\Gamma_k = \Gamma_{k-1} - \frac{\Gamma_{k-1} M_b(z_k)^T M_b(z_k) \Gamma_{k-1}}{1 + M_b(z_k) \Gamma_{k-1} M_b(z_k)^T}$$

(Ec. 45)

Para este caso $N=N_\Phi$ y el espacio de búsqueda tiene dimensión 2^N En algunos casos considerando un valor fijo de p, n_1, n_2 es muy probable que no se requieran de todos los elementos de $\phi(z)$ y, como se describió en la sección anterior, es posible seleccionar qué elementos se requieren para obtener una arquitectura o estructura óptima.

De esta forma, el problema de aprendizaje puede ser traducido a obtener la óptima estructura de RNAP utilizando AG.

3.3.2 Algoritmos Genéticos

Algoritmo Genético es un modelo de conocimiento que se encuentra en función de algunos de los mecanismos de evolución que se observan en la naturaleza. Este modelo de computación genética puede ser implementado con el uso de arreglos de bits o caracteres que representan los cromosomas. Aun cuando existe mucha investigación acerca de cadenas de longitud variable y sobre otras estructuras, la mayoría del trabajo con Algoritmos Genéticos está enfocado hacia cadenas de longitud fija; si no es manejado de esta manera, el Algoritmo de Evolución que se está considerando será Programación Genética, la cual se enfoca hacia cadenas de longitud variable (28, 29).

Cuando se implementa el Algoritmo Genético, usualmente sigue el ciclo (30, 31):

- Generación de una población inicial de forma aleatoria.
- Evaluación del más apto o alguna función objetivo de todos los individuos que pertenecen a la población.
- Creación de una nueva población debido a la ejecución de operaciones como recombinación (crossover) y mutación sobre los individuos de donde el más apto o el valor mejorado fue medido.
- Eliminación de la población original e iteración utilizando la nueva población hasta que el criterio de terminación sea cumplido o se haya llegado a cierto número de generaciones sin haber completado el objetivo planeado.

Figura 5: Etapas genéricas del Algoritmo Genético

El distinto tipo de operaciones que se hacen sobre las cadenas o cromosomas que son manipulados dentro del AG son llamadas operadores, en este trabajo se utiliza el operador de recombinación sexual, la mutación (32) y otros procesos especiales que son llamados agregar padres (26). Las siguientes secciones describen estos procesos en detalle.

3.3.2.1 Recombinación

El operador de recombinación se caracteriza por la combinación de material genético de los individuos que son seleccionados en función a su buen desempeño (función objetivo) que tuvieron en comparación con el resto de los individuos o "padres" que conforman la población. Existen algunas variantes de este operador (33), que puede ser determinado por el número de puntos de recombinación que serán considerados para la generación de una población, el número de padres involucrado para la generación del linaje, etc. Es obvio que si el número de puntos de recombinación se incrementa, el número de individuos que conforman el linaje, será también mayor.

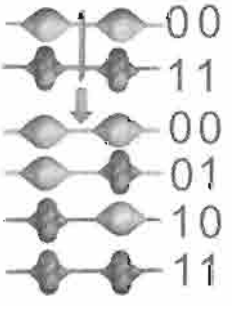

Figura 6: Operador de recombinación en un punto.

Para explicar la recombinación multipunto empleada, considérese que $C(F_g, n_i) \in IB^{n_i \times n_b}$

es el conjunto de padres de una población dada donde n_p es el número de padres de la población en la generación g y n_s es el número de bits del arreglo (cromosoma), $g=1,..,n_g$; y n_g es el número total de generaciones.

$C\left(F_g,n_t\right)\in IB^{n_p \times n_b}$ es el operador de recombinación y puede ser definido como la combinación entre el conjunto de padres considerando el número de intervalos n_t de cada individuo y el número de hijos n_s por lo que:

$$n_s = n_p^{\,n_t}$$

(Ec. 46)

Para mostrar como es que puede ser aplicado el operador de recombinación, considere que F_g se forma con $n_p=2$ y $n_t=3$. Esto significa que se divide el arreglo en tres secciones y se denomina a cada sección como a_i y b_i respectivamente para $i=1,..,n_t$. Si:

$$F_g=\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

$$\Rightarrow C\left(F_g,3\right)=\begin{bmatrix} a_1 & a_2 & a_3 \\ a_1 & a_2 & b_3 \\ a_1 & b_2 & a_3 \\ a_1 & b_2 & b_3 \\ b_1 & a_2 & a_3 \\ b_1 & a_2 & b_3 \\ b_1 & b_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

Es importante notar que con este operador los padres F_g de la población g se incluyen en el resultado de la recombinación.

3.3.2.2 Mutación

En el operador de mutación, sólo se niegan algunos bits que son seleccionados en forma aleatoria por medio de un factor de probabilidad P_m; en otras palabras, tan sólo se varían los componentes de algunos genes, es decir, se modifican los alelos. Este operador es extremadamente importante, ya que permite asegurar que se mantenga la diversidad dentro de la población, la cual es básica para la evolución (34,35).

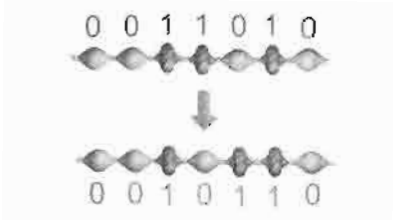

Figura 7: Operador de mutación

Este operador $M: IB^{n_s \times n_b} \rightarrow IB^{n_s \times n_b}$ cambia con probabilidad P_m la población generada por el operador recombinación en la siguiente forma:

$$M\left(C_{ij}\right)=\begin{cases} C_{ij}\oplus 1 & r \le P_m \\ C_{ij} & r > P_m \end{cases}$$

(Ec. 47)

donde $r\in U(0,1)$ es una variable aleatoria e, $i=1,..,n_s$; $j=1,..,n_b$; y $\oplus$ es el operador *x-or*.

El operador de mutación asegura que la probabilidad de encontrar cualquier punto en el espacio de búsqueda nunca sea cero y evita que se alcance un mínimo local.

3.3.2.3 Agregar Padres

En esta parte sólo se agrega F_g al resultado del proceso de mutación, entonces la población A_g en la generación g se puede obtener como:

$$A_g=\begin{bmatrix} M\left(C\left(F_g,n_t\right)\right) \\ F_g \end{bmatrix}$$

(Ec. 48)

Cabe hacer notar que A_g tiene los mejores individuos de A_{g-1} ya que se agrega F_g en este procedimiento. Este paso y el anterior aseguran la convergencia del algoritmo.

3.3.2.4 Proceso de selección

El Proceso de selección S_g calcula la función objetivo O_g que representa una función especifica que se quiere minimizar y se seleccionan los

mejores individuos n_p de A_g tal que:

$$S_g\left(A_g, n_p\right) = \min^{n_p} O_g\left(A_g\right)$$

(Ec. 49)

Entonces:

$$F_{g+1} = S_g\left(A_g, n_p\right)$$

(Ec. 50)

Note que los mejores individuos de la generación pueden ser obtenidos por el siguiente operador:

$$S_g\left(A_g, 1\right)$$

(Ec. 51)

En resumen el Algoritmo Genético puede ser descrito por medio de los siguientes pasos:

1. Para la condición inicial $g=0$ seleccionar en forma aleatoria A_0, tal que $A_0 \in IB^{n_s x n_b}$
2. Calcular $F_1 = S_0(A_0)$
3. Obtener A_g
4. Calcular S_g
5. Regresar al paso 3 hasta que el número máximo de generaciones sea alcanzado o uno de los individuos de S_g obtenga el mínimo valor deseado de O_g.

El empleo de Algoritmo Genético es especialmente útil cuando se tiene gran cantidad de posibilidades en el espacio de búsqueda y no se posee información acerca de cómo obtener la solución óptima para un problema específico. Para este caso, la aplicación de la teoría es automática si se considera a W_b^* como el arreglo buscado. El problema de aprendizaje puede ser trasladado para obtener la estructura óptima de PANN usando AG.

Los pasos de AG modificados pueden observarse en la siguiente tabla:

Tabla 1. Comparación entre AG tradicional y en la RNAP

ALGORITMO GENÉTICO	ALGORITMO GENÉTICO EN RNAP
1. Para la condición inicial $g=0$ se calcula A_0 de forma aleatoria con las siguientes dimensiones: $A_0 : n_s x n_b$	1. Para la condición inicial $g=0$ se calcula A_0 de forma aleatoria con las siguientes dimensiones: $A_0 : n_s x n_b$ donde cada renglón de la matriz corresponde a una propuesta para W_b
2. Se obtiene la función objetivo y se selecciona al mejor individuo para la población inicial $F_1 = S_0(A_0, n_p)$	2.1. Para calcular F_1 primero se debe calcular $S_0(A_0, n_p)$ donde la función objetivo O_g puede ser calculada utilizando el error definido por Ec. 41 de la siguiente forma: $O_0^i = err_n^i\left(y^n, \left\langle \left(W^i\right)\Big\|_{W_b}^{i}, M(z)\right\rangle\right), i = 1,..,n_s$ 2.2. Se calcula $S_g\left(A_g, n_p\right) = \min^{n_p} O_g\left(A_g\right)$
3. Se obtiene la nueva población A_g en la generación g con el operador recombinación y mutación.	3. Se obtiene la nueva población A_g en la generación g con el operador recombinación y mutación.
4. Calcular la función objetivo y seleccionar a los mejores individuos de la generación con S_g	4. Calcular S_g con la siguiente función objetivo: $O_g^i = err_n^i\left(y^n, \left\langle \left(W^i\right)\Big\|_{W_b}^{i}, M(z)\right\rangle\right), i = 1,...n_s + n_p$
5. Regresar al paso 3 hasta que se alcance el máximo número de generaciones o uno de los individuos de S_g obtengan el mínimo valor deseado de O_g	5. Regresar al paso 3 hasta que se alcance el máximo número de generaciones o uno de los individuos de S_g obtengan el mínimo valor deseado de O_g

El error óptimo en la generación g puede ser obtenido por:

$$\left(\underset{W' \in \Re^{N}}{opt\ err_{n}} \right)_{g} = \min_{W_b \in A_c} err_{n}\left(y^{n}, \left\langle (W')\big|_{W_b}, M(z) \right\rangle \right)$$

(Ec. 52)

Teorema 4.1

Definiendo

$$W_b^{g} \overset{\Delta}{=} S_g\left(A_g, 1\right)$$

(Ec. 53)

donde W_b^g es el mejor individuo en la generación *g*, es decir, la estructura óptima de la RNAP en ese momento, se puede reescribir el error de aproximación como:

$$err\left(y^{n}, \phi_g(z)\right) = \frac{1}{n}\sum_{k=1}^{n}\left(y_k - \left\langle W, M(z) .* \left(W_b^{g}\right)^{T}\right\rangle\right)^{2}$$

(Ec. 54)

donde y^n es la salida deseada, $\phi_g(z) \in \Phi_p$ es la representación óptima en la generación *g*, *y*

$$W\big|_{W_b} = \arg\min_{W \in \Re^{N}} err_{n}\left(y^{n}, \phi(z)\right)\Big|_{W_b = W_b^{g}}$$

Si $P_m>0$ entonces *RNAP* aprende uniformemente la salida deseada de tal forma que:

$$\lim_{g \to \infty} P\left\{ err\left(y, \phi_g(z) - opterr\left(y, \phi(z)\right)\right) \geq \varepsilon \right\} = 0, \quad \varepsilon > 0$$

(Ec. 55)

$\phi_g \in \Phi_p$ puede ser descrita como:

$$\phi_g = \left\langle W, M(z) .* S_g\left(A_g, 1\right)^{T} \right\rangle$$

(Ec. 56)

Debido al progreso de agregar padres y el de mutación

$$err\left(y, \phi_{g+1}(z)\right) \leq err\left(y, \phi_g(z)\right)$$

(Ec. 57)

$$\Rightarrow \lim_{g \to \infty} err\left(y, \phi_g(z)\right) = opterr\left(y, \phi(z)\right)$$

(Ec. 58)

Comentario 1

Para casos prácticos el Teorema 4.1 puede escribirse utilizando la definición 4.1 como:

Si $P_m>0$ entonces la RNAP aprende uniformemente la salida deseada con precisión e en un número finito de generaciones n_g si:

$$\lim_{g \to n_g} P\left\{ err\left(y, \phi_g(z) - opterr\left(y, \phi(z)\right)\right) > \varepsilon \right\} = 0 \quad \varepsilon > 0$$

(Ec. 59)

4 MANCHAS SOLARES

Las manchas solares, por razones de contraste con el resto de la superficie solar, aparentan ser regiones oscuras; sin embargo, son completamente luminosas. Su temperatura es un par de miles de grados más pequeña que la del resto de la superficie solar. (36)

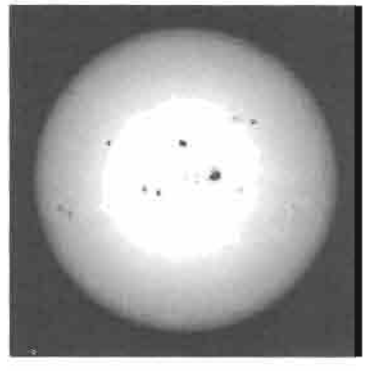

Figura 8: Manchas solares

El astrónomo norteamericano Hale descubrió que las manchas solares contienen campos magnéticos muy fuertes. Una mancha solar tipica tiene campo magnético con una fuerza de 2500 gauss. En comparación, el campo magnético de la Tierra tiene una fuerza de 1 gauss. Las particulas cargadas eléctricamente tienen tendencia a seguir la dirección del campo magnético, describiendo espirales alrededor de las líneas de fuerza generadas por las manchas solares como se muestra en la estructura de las manchas solares propuesta por Hale.

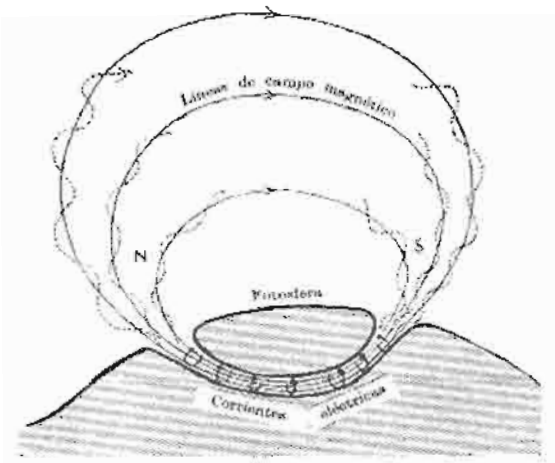

Figura 9: Estructura de las manchas solares

Un estudio sugirió que el número de manchas varía con un período de aproximadamente once años y medio (1). Desde entonces, el número de manchas solares ha sido registrado cuidadosamente por los astrónomos, generando así una serie de tiempo con casi trescientos datos.

Algunos fenómenos terrestres provocados por la variación del número de las manchas solares son:

- Crecimiento acelerado de los árboles durante los años máximos de manchas.
- Crecimiento lento del trigo durante los años pobres en manchas.
- Dependemos de las capas superiores de la atmósfera para las comunicaciones, las manchas solares varían la ionización de estas capas, por lo que la reflexión angular resulta distorsionada, causando que la comunicación radiada en el mundo falle.
- Las manchas hacen que algunas partículas energéticas alcancen la tierra, causando las auroras boreales.

5 RESULTADOS

5.1 Red Neuronal Artificial Polinomial.

La serie de tiempo de manchas solares fue introducida a RNAP para generar el modelo matemático.

Los parámetros con los cuales se llevaron a cabo las pruebas son:

(no se tienen entradas).

$n_1 = 0$	(para cada número de valores anteriores, se hace la prueba).
$n_2 = 3...15$	(potencia máxima del polinomio).
$p = 2$	

Las pruebas se dividen en dos grupos, el modelado y predicción de la serie de tiempo de manchas solares.

Los primeros 250 datos son usados para entrenar la red, mientras que los últimos 30 ayudan a evaluarla. En las gráficas presentadas, las cruces señalan los valores reales, mientras que los trazos indican la aproximación obtenida a la serie durante el aprendizaje y la predicción de la red.

La Tabla 2 presenta el número de valores anteriores empleados para cada una de las predicciones y su respectivo error, tanto de prueba como de entrenamiento.

Tabla 2. Errores de entrenamiento y prueba de 3 hasta 15 los valores anteriores

Valores Anteriores	Error de Entrenamiento	Error de Prueba
3	0.0045	0.0162
4	0.0044	0.0152
5	0.0042	0.0180
6	0.0039	0.0168
7	0.0036	0.0153
8	0.0034	0.0185
9	0.0031	0.0114
10	0.0030	0.0135
11	0.0029	0.0102
12	0.0026	0.0146
13	0.0025	0.0137
14	0.0024	0.0085
15	0.0022	0.0104

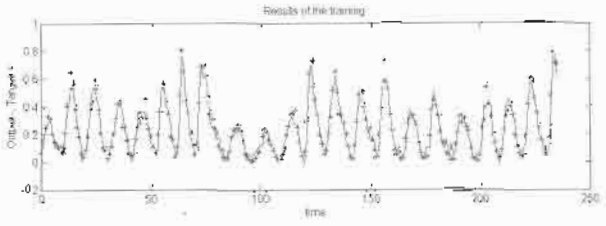

Figura 10: Entrenamiento con 14 valores anteriores. Error de entrenamiento: 0.0024

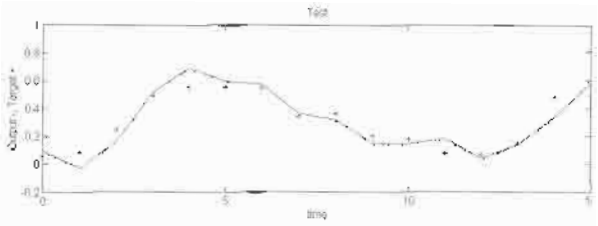

Figura 11: Prueba con 14 valores anteriores. Error de prueba: 0.0085

El algoritmo de PANN demostró tener gran capacidad para modelar series de tiempo.

Se pudo observar que tanto el error de entrenamiento como el de prueba son pequeños, por lo que se les puede considerar como aceptables En las pruebas realizadas, se observa que cuando se utilizaron 14 valores anteriores, se obtuvo el menor error. Es importante mencionar que la serie de tiempo fue introducida a la red neuronal polinomial artificial y ésta se encargó de modelar su comportamiento. Se ha observado que cuando se aplica un preprocesamiento a la entrada por medio de filtros multiresolución, el error disminuye considerablemente.

5.2 Perceptrón Multicapa.

A la hora de entrenar un perceptrón multicapa, surgen una serie de preguntas, como, por ejemplo, cuál debe ser la topología de la red: cuántas capas se ponen, cuántas neuronas debe tener cada capa, qué valores deben tomar los parámetros de entrenamiento, h y a? Lo cierto es que no existe respuesta a estas dudas más que las pruebas sistemáticas o la codificación de esta información no definida en algún tipo de algoritmo de optimización (como por ejemplo los algoritmos genéticos). Se optará en este caso por una exploración sistemática desde 3 hasta 14 neuronas de entrada y, en la capa oculta, desde 3 hasta 3 veces el número de neuronas de entrada. Se tomarán como parámetros de entrenamiento h = 0.01 y a = 0.1 que son valores usuales. Se prepararán los elementos de la serie para crear los pares (entrada / salida) que constituirán los conjuntos de entrenamiento y de prueba. De entrada se esclarecerá la serie de tal manera que tome valores entre 0.9 y 0.9, para aprovechar de manera óptima el rango de valores que toma la imagen de la tangente hiperbólica que será nuestra función de activación. Como conjunto de pruebas, se tomarán la predicción de la serie para los últimos 30 años. Se evaluarán los resultados a partir del cómputo del error cuadrático medio. Durante el entrenamiento, los patrones deben presentarse cambiando el orden a cada ciclo de entrenamiento (este ciclo se conoce también como epoch o iteración) para evitar que la capacidad de generalización de la red quede sesgada. Además, se debe tener en cuenta que la superficie de error que se intenta minimizar tiene una gran cantidad de mínimos locales donde las técnicas de gradiente inverso que se utilizan pueden quedar atrapadas. Por ello el valor final de los pesos al terminar el entrenamiento dependerá del valor inicial aleatorio que tomen éstos, así como del orden en que se presenten los diferentes elementos del conjunto de entrenamiento. Por este motivo, para cada topología se entrenarán 20 redes para quedarse con la mejor, esto es la que presente un menor error de prueba.

Los resultados se leen en la tabla adjunta :

Tabla 3. Resultados del entrenamiento del perceptrón multicapa para h=0.01 y a=0.1 sobre 200 iteraciones.

Neuronas de la Capa de Entrada	Neuronas de la Capa Oculta	Error de Entrenamiento	Error de Prueba
3	5	0.006571	0.018927
4	3	0.005783	0.015767
5	8	0.004423	0.016238
6	16	0.003889	0.014532
7	20	0.003841	0.013357
8	19	0.003929	0.012058
9	18	0.004222	0.011434
10	11	0.004430	0.011849
11	09	0.004162	0.009613
12	17	0.003428	0.010885
13	9	0.004937	0.011717
14	23	0.003766	0.012589

Según estos datos la mejor red corresponde a 11 neuronas de entrada y 9 en la capa oculta. En las figuras siguientes se puede observar los resultados obtenidos para esta red.

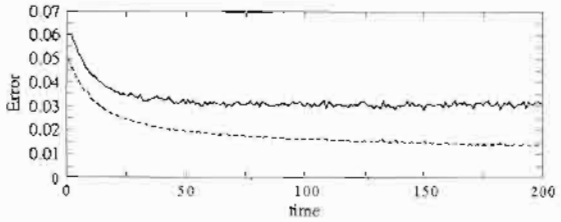

Figura12: Evolución del error de prueba (línea continua) y de entrenamiento (línea discontinua). Error de entrenamiento en la red perceptrón multicapa.

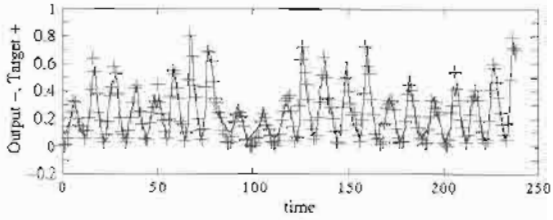

Figura 13: Comparación de los datos reales (cruces) con los de entrenamiento (línea continua). Comparación de los datos reales con los obtenidos por el perceptrón multicapa en el entrenamiento.

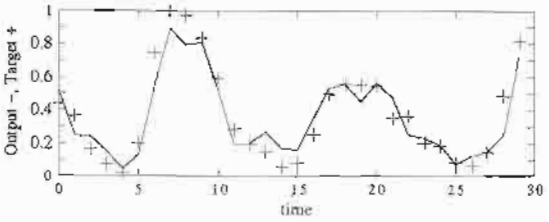

Figura 14: Comparación de los datos reales (cruces) con los de prueba (línea continua). Comparación de los datos reales con los predichos por el perceptrón multicapa

5.3 Redes de Elman modificadas

Para las redes de Elman modificadas y ante dudas parecidas a las que se plantean con el perceptrón multicapa, realizamos una exploración sistemática del número de neuronas de entrada y de la capa oculta. Tomamos valores entre 1 y 4 para el número de neuronas de entrada y entre 3 y 11 para el número de neuronas de la capa oculta. Estos números son claramente inferiores a los escogidos para el perceptrón multicapa para ilustrar cómo la red de Elman modificada es capaz de obtener resultados comparables con un número menor de neuronas. De nuevo se entrenarán 20 redes inicializadas con pesos aleatorios y se tomará la mejor, según el criterio definido en el apartado anterior. Como valores de los parámetros de entrenamiento se toman h=0 01 y g, el término de momento, renombrándolo para no confundir con el parámetro a de memoria de la capa de contexto, 0.09. Para a se usa el valor 0.5. Estos valores se escogen por ser usuales.

Los resultados se leen en la tabla adjunta.

Tabla 4. Tabla de resultados para la red de Elman modificada. Los valores de los parámetros de la red son η=0.01 y γ=0.09.

Neuronas de la Capa de Entrada	Neuronas de la Capa Oculta	Error de Entrenamiento	Error de Prueba
1	11	0.008201	0.009513
2	11	0.014714	0.010992
3	7	0.005972	0.008425
4	11	0.010037	0.010666

Seguidamente presentamos las gráficas de error y resultados para la mejor red.

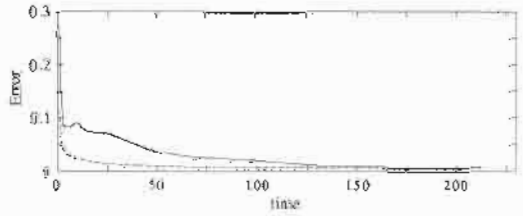

Figura15: Evolución del error de prueba (línea continua) y de entrenamiento (línea discontinua). Error de entrenamiento en la red de Elman.

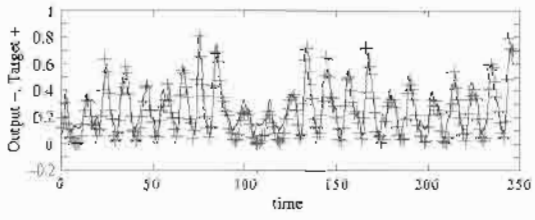

Figura16: Comparación de los datos reales (cruces) con los de entrenamiento (línea continua). Comparación de los datos reales con los obtenidos por la red de Elman en el entrenamiento.

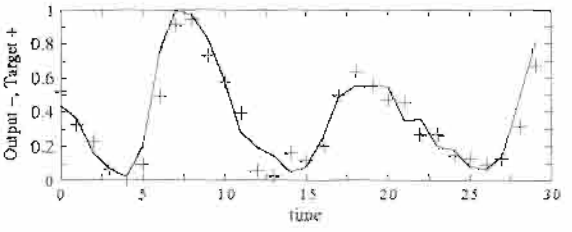

Figura17: Comparación de los datos reales (cruces) con los de prueba (línea continua). Comparación de los datos reales con los predichos por la red de Elman.

6 CONCLUSIONES

A juzgar por los resultados obtenidos con los tres tipos de redes neuronales analizadas en este artículo llegamos a las siguientes conclusiones referentes al rendimiento de cada red. En principio y a partir de los datos analizados, parece ser que el Perceptrón multicapa requiere de un mayor número total de neuronas (11 de entrada y 9 en la capa oculta) que la red neuronal polinomial (14 valores anteriores) y que la red de Elman (3 en la capa de entrada y 7 en las capas ocultas y de contexto) para obtener los mejores resultados en el error de prueba. Por contrapartida, el Perceptrón multicapa es la red más simple de las tres, con lo que el aumento en el número de neuronas queda compensado por el menor costo computacional requerido. Asimismo los errores más bajos obtenidos con cada tipo de red son comparables, con lo que aparentemente las tres redes resuelven de forma similar el problema planteado.

Estos resultados sin embargo no pueden generalizarse fácilmente a otros problemas como los que se obtienen cambiando la serie de manchas solares por otra gobernada por una dinámica diferente. De la misma forma, es difícil extrapolar las conclusiones referentes al número de neuronas óptimo en cada red a problemas en los que las redes deban ser mucho mayores, máxime teniendo en cuenta que los parámetros específicos de cada red (a, h, ...) se han fijado a valores razonables en lugar de realizar una amplia exploración en su espacio de valores.

7 REFERENCIAS BIBLIOGRÁFICAS

(1) G.F.Franklin, J.D.Powell, M.Workman, *Digital control of dynamic systems* - 3rd Ed., Addison-Wesley, 1998.

(2) L.Ljung, *System identification*, Prentice-Hall, 1987.

(3) J.A.Freeman, D.M.Skapura, *Redes Neuronales*, Addison-Wesley/Díaz de Santos, 1993.

(4) Elisabet Golobardes, Eduard Pous i Manuel Román. *Introducció a les xarxes neuronals, presentació de les Backpropagation*, INPUT 10 17-26, 1996.

(5) K.S.Narendra, K. Parthasaranthy, *Identification and control of dynamic systems using neural networks*, IEEE Trans. on Neural Networks, 1, 4-27, 1990.

(6) D.T.Pham, X.Liu, *Neural Networks for identification, prediction and control*, Springer-Verlag, 1995.

(7) Hornik K., Stinchcombe M. & White H.. Multilayer Feedforward Networks Are Universal Approximators, *Neural Networks*, 1989.

(8) Rosenblatt, Frank., The Perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, 65, 386-408, 1958.

(9) G.E.P.Box, G.M.Jenkins *Time series analysis*, Holden-Day, 1976.

(10) R.Lewandowski, *La prévision à court terme*, Dunod, 1985.

(11) D.E.Rumelhart, J.L.M.McClelland and the PDP research group, *Parallel Distributed Processing*, MIT Press, 1986.

(12) J.L.Elman *Finding structure in time*, Cognitive Science 14, 179-211, 1990.

(13) McCulloch, Warren S. & Pitts, Walter. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5:115-133, 1943.

(14) MacGregor, R. J., *Neural and Brain Modeling*. Academic Press, San Diego, Ca, 1987.

(15) Hornik K., Stinchcombe M. & White H.. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*. 1989.

(16) Park, D.C. et al., "Electric Load Forecasting Using an Artificial Neural Network", *IEEE Trans. Power Systems*, Vol. 6, núm. 2, pp. 442-449, 1991.

(17) Chang C , Lin J.; & Cheung J., Polynomial and Standard Higher Order Neural Network. IEEE *International Conference on Neural Networks*. p. 989-94 vol.2, San Francisco, CA, USA, 28 Marzo-1 Abril, 1993.

(18) Chen S. & Billings S., Representations of nonlinear systems: the NARMAX model, *Int. J. Control*, vol. 49, núm. 3, 1989.

(19) Alippi C. & Piuri V., Experimental Neural Networks for Prediction and Identification, *IEEE Transactions on Instrumentation and measurement*, vol. 45, núm. 2 Abril, 1996.

(20) Leontaritis Y. & Billings S., Input-output parametric models for nonlinear systems, *Int. J. Control*, vol. 41, núm. 2, 1985.

(21) Chen S. & Billings A., Recursive Prediction error parameter estimator for nonlinear models, *Int. J. Control*. vol. 49, núm. 2, 1989.

(22) Ivakhnenko A., Polynomial Theory of Complex systems, *IEEE Transactions on systems, Man and Cybernetics*, vol. SMC-1, núm. 4 Octubre, 1971

(23) Duffy J. & Franklin M., A Learning Identification Algorithm and Its Application to an Environmental System, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-5, núm. 2, marzo, 1975.

(24) Shin Y. & Gosh J., Approximation of Multivariate Functions Using Ridge Polynomial Networks, *IJCNN'92*, p. 380-5, Baltimore, MD,USA, 7-11 Junio, 1992

(25) Shin Y., Modified Bernestein Polynomials and Their Connectionist Interpretation, *ICNN*, p. 1433-8, vol. 3, Orlando, Florida, USA, 27 Junio-2 Julio, 1994.

(26) E. Gómez-Ramírez, A. Poznyak, A. González Yunes & M. Ávila-Álvarez. Adaptive Architecture of Polynomial Artificial Neural Network to Forecast Nonlinear Time Series. *CEC99 Special Session on Time Series Prediction*. Mayflower Hotel, Washington D.C., USA, Julio 6-9, 1999.

(27) Gómez Ramírez E. & Poznyak A. How to Select a Number of Nodes in Artificial Neural Networks. *Neural Networks Applied to Control and Image Processing. NNACIP'94*, CINVESTAV IPN, Noviembre 7-11, México, D.F., 1994.

(28) Altenberg L., The Evolution of Evolvability in Genetic Programming, 1994, MIT Press.

(29) Andre D., Automatically Defined Features: The Simultaneous Evolution of 2-Dimensional Feature Detectors and an Algorithm for Using Them, 1994, MIT Press, Kenneth E..

(30) Alander J. T., An Indexed Bibliography of Genetic Algorithms: Years 1957-1993, 1994, Art of CAD ltd.

(31) Bedner I., Genetic Algorithms and Genetic Programming at Standford 1997, 1997, Stanford Bookstore.

(32) Andre D. and Koza J., Advances in Genetic Programming 2, 1996, MIT Press.

(33) Andrews M. & Prager R., Advances in Genetic Programming, 1994, MIT Press.

(34) Altenberg L., Genome growth and the evolution of the genotype-phenotype map, 1995, Springer-Verlag, Berlín, Alemania.

(35) Banzhaf W., Nordin P, Kelier R. & Francone F., Genetic Programming - An Introduction On the Automatic Evolution of Computer Programs and its Applications, 1997, Morgan Kaufmann, dpunkt.verlag.

(36) George Gamow, «Una estrella llamada Sol» Ed. Espasa-Calpe, 1967.