3. Metodología propuesta

En este capítulo se plantea la solución adoptada. Primero se define el planteamiento del problema y luego se describe la metodología propuesta en sus aspectos tanto formales como de organización para su procesamiento paralelo y distribuido, así como el tipo de resultados que se pretende obtener.

3.1. Hipótesis de trabajo

Se va a suponer que las variables que se conocen, las entradas y las salidas, e_j y s_j respectivamente, definen el estado del sistema, bien por sus valores actuales, en el caso estático, bien por la historia de valores que han tomado, en el caso dinámico. Para esto las variables elegidas deben ser suficientemente representativas. Así pues, para una determinada salida s_i, se tiene:

$$s_i(t) = f\left(s_j(t-k), s_j(t-m), ..., s_l(t-n), ..., e_p(t-q), ..., e_r(t-s)\right) \qquad (3.1)$$

siendo la función *f* desconocida.

En cualquier caso, nuestro modelo va a recibir como información la historia de las entradas y salidas habidas en una serie de secuencias y se quiere que sea capaz de reproducir las salidas en el instante *t* según (3.1).

A todos los efectos se admitirá que las muestras son representativas del comportamiento del sistema, es decir, que incluyen todas las formas de funcionamiento.

3.2. Planteamiento del problema

Retomando los objetivos mencionados en la introducción, se quiere interpretar y simular un cierto sistema.

Sea *S* el sistema en cuestión; a efectos de nuestro estudio, se puede definir:

$$S: \{e_i\} \rightarrow \{s_j\}$$

donde e_i son las señales de entrada y s_j las de salida.

En general el sistema presentará unas respuestas en función de sus estados, que a su vez dependerán de los estímulos, en la forma:

$$s_j(t) = f\left(x_i(t-1)\right) + \varepsilon$$
$$x_k(t) = g\left(e_l(t-1)\right)$$

donde tanto los estímulos o entradas como las variables de salida pueden estar incluidas en el conjunto de variables de estado, x_n.

Si el conjunto de variables de entrada y salida es suficientemente descriptivo su historia puede sustituir a las variables de estado.

Este planteamiento se puede proponer también partiendo del estudio en tiempo continuo. Así, si el sistema verifica aproximadamente una cierta ecuación diferencial

y se sustituye esta expresión por una ecuación en diferencias, fácilmente se llega a conclusiones análogas.

Queda planteado entonces que los argumentos de la función que se va a intentar representar son valores previos de las salidas y las entradas al sistema. La expresión (3.1) define la estructura de la primera capa (la entrada) de la red. Será misión del algoritmo de ajuste descubrir cuáles son los términos que deben intervenir, es decir, qué variables y con qué desfases.

Este enfoque es análogo al usado en funciones de transferencia en el entorno de series temporales [63].

En resumen se puede deducir que el algoritmo tiene que ser capaz de averiguar:

- Los retrasos *k*, *m* que contienen la información necesaria.

- Las variables $\{s_j, s_k, ..., e_m, ..., e_l\}$ que intervienen, ya que es posible que algunas de ellas no sean necesarias.

- La estructura de la función *f*.

En estas tres líneas se tiene que mover el ajuste y desarrollo de la red.

3.3. Interpretabilidad

Además de tener un cierto modelo preparado para reproducir el sistema, la utilidad sería mayor si la función ajustada se pudiera interpretar con facilidad por lo menos cualitativamente, por ejemplo, nuestro modelo debería obtener, a partir de las muestras, si hay dependencias de distintas variables según el estado del sistema; y dar su forma aproximada. En general para que la interpretación sea sencilla deberá cumplirse que:

- El número de parámetros que caracteriza el modelo no sea elevado.

- La estructura del modelo sea similar a la de los utilizados habitualmente. Los modelos más usuales son los comentados en el capítulo anterior.

El primer objetivo es incluso necesario para un buen comportamiento del modelo, por lo que no es precisa una insistencia adicional en él. El segundo es el que introduce un condicionamiento adicional, ya que obliga a que la estructura de la red sea traducible a la estructura de los modelos habituales.

3.4. Capacidad de extrapolación

Recordando el paradigma de la creación inductiva del conocimiento, línea en que se mueve el algoritmo propuesto, se está extrayendo información de un conjunto experimental para aplicarlo después con toda generalidad. Para mantener el buen comportamiento del modelo fuera de las muestras de ajuste es necesario atenerse al principio del máximo coste con el mínimo beneficio, que en este contexto se interpreta como mínimo modelo para máxima precisión. Esto es equivalente a evitar un modelo que tenga una enorme variabilidad. El modelo más sencillo que contenga toda la información necesaria es el óptimo.

Es importante subrayar que en la práctica no se puede ni se debe pretender un error nulo. La curva que relaciona complejidad con precisión tiene la forma indicada en la fig. 3-1, lo que indica la conveniencia de elegir un valor de precisión que sea suficiente para nuestros objetivos sin necesidad de recurrir a una complejidad que forzosamente perjudicaría el comportamiento del modelo fuera de la muestra de ajuste.

Fig. 3-1. Forma genérica de la curva precisión/complejidad.

3.5. Solución adoptada

Es el momento de recordar los objetivos propuestos en la introducción, a saber, la creación de un tipo de red aplicable a secuencias temporales de variables continuas en forma lo más independiente posible del usuario. Como se expresó en el capítulo anterior, ya existe una metodología que consigue esto [1], pero presenta dos inconvenientes que no aparecerán en la metodología que se propone con la presente tesis. Estos inconvenientes son:

- No incluye control de sobreajuste.
- No es interpretable.

Así pues, en lo que sigue se describe una metodología que pretende mantener la independencia de usuario y solucionar estos dos aspectos.

3.5.1. Tipo y topología de la red

Según se menciona en el apartado 3.2 la estructura de la capa de entrada quedará definida de forma que recoja valores actuales de entradas y precedentes de las variables de entrada y salida que intervengan. Estos valores se denominarán x_i. Además, por cuestiones operativas, en esta capa se añade un procesador de respuesta constante unidad. Para referirse a la respuesta de un procesador de la capa de entrada, incluyendo el constante, se utilizará la notación z_i.

Las variables que intervienen y los procesadores que las representan quedan definidas tras el proceso de ajuste.

La capa de salida está obviamente definida, en función y número, por las variables de salida, con un procesador para cada una. Sus respuestas se identificarán como y_i.

En cuanto al resto de la red (las capas ocultas), en este modelo se hace una división en dos capas:

Una capa tendrá funciones de respuesta local, es decir, que de todo el espacio de sus entradas, sólo tomará valor en una cierta zona. Por L_i se identificarán estos procesadores o sus respuestas, según el contexto.

La otra capa será lineal. Así, cada procesador tendrá como respuesta una combinación lineal de sus entradas. La notación empleada para estos procesadores será l_i.

Con esto quedan definidos todos los tipos de procesadores que entran a formar parte de la red. A continuación se tratan las conexiones.

Cada procesador de salida estará conectado a un cierto número de procesadores de la capa lineal y a la misma cantidad de procesadores de la capa local. Cada procesador de las capas ocultas estará conectado a un procesador de la capa de salida, y sólo a uno. Véase un ejemplo de estas conexiones para el caso de 2 salidas en la fig. 3-1.

Las conexiones serán de segundo orden y peso unidad (no ajustable), es decir, cada conexión interviene en la respuesta mediante un término que es el producto de un procesador local y otro lineal. Según esto la expresión resultante para el procesador de salida i-ésimo es:

$$y_i = \sum l_{ij} L_{ij} \tag{3.2}$$

donde L_{ij} es el procesador de respuesta local j-ésimo de los conectados a la salida i y análogamente l_{ij} en la capa lineal. Como se ve los procesadores locales y lineales quedan emparejados. El rango de variación de j, esto es, cuántas parejas de procesadores hay asociadas a cada salida se define durante el ajuste constructivo de la red, que se expone más adelante.

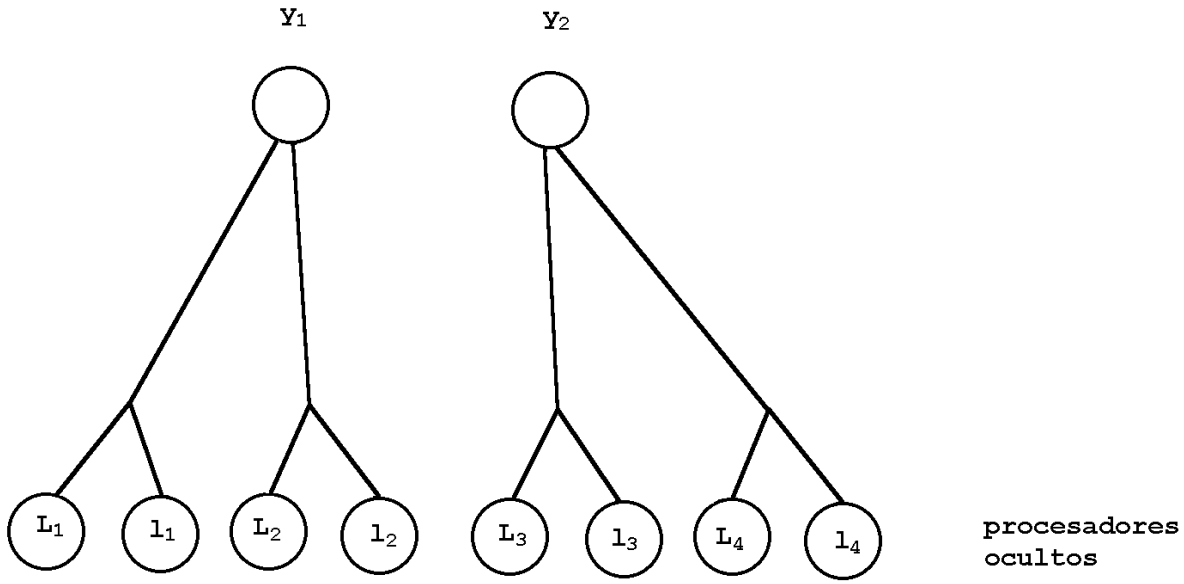

Fig. 3-1. Ejemplo de grafo de conexiones entre los procesadores ocultos (locales y lineales) y los de salida.

En cuanto a las conexiones que llegan a la capa lineal, cada uno de estos procesadores estará conectado en principio a todos los procesadores de entrada, pudiéndose, durante el ajuste, eliminar alguna de estas conexiones. Estas conexiones serán convencionales de pesos ajustables; así para el procesador j-ésimo de los asociados a la salida *i*, se tendrá su respuesta expresada como:

$$l_{ij} = \sum p_{ijk} z_k \tag{3.3}$$

siendo p_{ijk} el peso correspondiente.

Queda por último definir las conexiones de los procesadores locales con los de entrada, así como la forma exacta de su función de respuesta o activación. Estos procesadores estarán conectados a todos los de entrada **menos** al de valor constante, por lo que se empleará la notación x_i. Cada conexión tendrá dos parámetros ajustables. La expresión de la activación del procesador j-ésimo asociado a la salida *i* será:

$$L_{ij} = \begin{cases} 1, \text{si} \sum_k \left(\frac{x_k - m_{ijk}}{\sigma_{ijk}} \right)^2 = \min_p \left[\sum_n \left(\frac{x_n - m_{ipn}}{\sigma_{ipn}} \right)^2 \right] \\ 0, \text{en caso contrario} \end{cases} \tag{3.4}$$

donde m_{ijk} y σ_{ijk} son dos parámetros asociados a la conexión con el procesador cuya respuesta es x_k, a los que se denominará respectivamente centros y anchos.

Esta expresión presenta la peculiaridad de que, de los procesadores locales asociados a la salida *i*, en cada instante sólo hay uno de respuesta no nula, salvo la circunstancia de que el mínimo no fuera único, lo que resulta muy poco probable en la práctica. Esto permitirá que los ajustes de los parámetros sean independientes.

Un esquema completo para una entrada y una salida es el mostrado en la fig. 3-2.

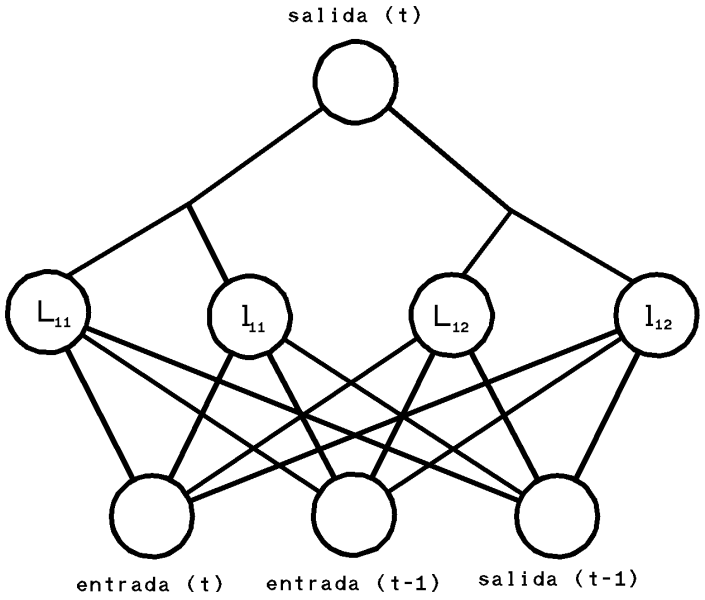

Fig. 3-2. Esquema del tipo de red propuesto.

Gracias a la falta de solape en la respuesta en la capa local, esta se convierte en una capa de clasificación, que permite a la red elegir en cada momento, en función de las entradas x_k una serie de procesadores lineales (uno para cada salida).

Las expresiones (3.3) y (3.4) junto con la (3.2) proporcionan una expresión completa de la red que define las salidas en función de las entradas, resultando la expresión:

$$y_i = \sum p_{ijk} z_k \partial_j^l \text{ donde } \sum_k \left(\frac{x_k - m_{ilk}}{\sigma_{ilk}} \right)^2 = \min_n \left[\sum_k \left(\frac{x_k - m_{ink}}{\sigma_{ink}} \right)^2 \right]$$

Obsérvese que el esquema de clasificación en la capa *local* conduce a un ajuste a trozos, esto es, discontinuo. Este posible inconveniente se estudia y analiza en el apartado 3.6.1.

3.5.2. Ajuste

En la línea del método híbrido presentado en [73] se propone una optimización de los parámetros de la red en dos fases; la primera consiste en un ajuste no supervisado de los centros (m_{ijk}) y anchos (σ_{ijk}) de la capa local de clasificación, y la segunda es una optimización del error cuadrático de los pesos de los procesadores lineales. Se deja abierta la vía de un posible ajuste supervisado posterior de la clasificación, como se comenta en el apartado 3.6.2.

El ajuste de los centros se hace de forma que los correspondientes a cada procesador L_{ij} sólo se ven afectados cuando la respuesta de este no es nula (según la expresión 3-4), de forma que el vector formado por todos los parámetros m_{ijk} se coloque en el punto medio del subconjunto de puntos de la muestra para los que L_{ij} resulta valer 1. Para ello se plantea un proceso como el propuesto en [73]:

$$\Delta_l \mathbf{M_{ij}} = \eta(\mathbf{x}_l - \mathbf{M_{ij}})$$

que define el incremento que se da con el punto l-ésimo de la muestra al vector $\mathbf{M_{ij}}$ cuyas componentes son los parámetros m_{ijk} de las conexiones que entran al procesador local L_{ij}, siendo $\mathbf{x_l}$ el vector cuyas componentes son los valores de los procesadores de variables de entrada y η un parámetro que puede variarse para optimizar el proceso.

Este método, planteado de forma que los centros se actualizan tras cada pasada por todos los puntos de la muestra converge rápidamente a la media. Efectivamente, el movimiento total será:

$$\Delta\mathbf{M_{ij}} = \eta\sum_l (\mathbf{x}_1 - \mathbf{M_{ij}}) = \eta\left(\sum_l \mathbf{x}_1 - \mathrm{N}\mathbf{M_{ij}}\right)$$

siendo N el número de puntos que están más próximos a $\mathbf{M_{ij}}$ que a otro centro de otro procesador; tomando $\eta=1/N$, se tiene

$$\Delta\mathbf{M_{ij}} = \overline{\mathbf{x}} - \mathbf{M_{ij}}$$

siendo $\overline{\mathbf{x}}$ el vector media de los $\mathbf{x_l}$.

Como N es desconocido a priori podemos tomar $\eta=1/N_{tot}\leq 1/N$, siendo N_{tot} el número total de puntos en la muestra, resultando:

$$\Delta \mathbf{M_{ij}} \leq \bar{\mathbf{x}} - \mathbf{M_{ij}}$$

lo que puede ocasionar retardo en la convergencia. La nueva distancia a la media será:

$$\bar{\mathbf{x}} - (\mathbf{M_{ij}} + \frac{N}{N_{tot}}(\bar{\mathbf{x}} - \mathbf{M_{ij}})) = \bar{\mathbf{x}} - (\mathbf{M_{ij}}(1 - \frac{N}{N_{tot}}) + \frac{N}{N_{tot}}\mathbf{M_{ij}}) = \bar{\mathbf{x}} - (\mathbf{M_{ij}}(1-a) + a\bar{\mathbf{x}}) = \bar{\mathbf{x}}(1-a) - \mathbf{M_{ij}}(1-a) = (\bar{\mathbf{x}} - \mathbf{M_{ij}})(1-a)$$

siendo $a=\frac{N}{N_{tot}}$

Si $\overline{\mathbf{x}}$ y a se mantienen constantes el número de iteraciones n_{it} necesario para que la distancia sea menor que ε será

$$n_{it} = \frac{\log\left(\frac{\varepsilon}{|\bar{\mathbf{x}} - \mathbf{M_{ij}^0}|}\right)}{\log(1-a)}$$

donde $\mathbf{M_{ij}^0}$ es el valor inicial de $\mathbf{M_{ij}}$.

Sin embargo es probable que $\overline{\mathbf{x}}$ y a se modifiquen al cambiar la posición de $\mathbf{M_{ij}}$, por lo que el tiempo de estabilización será mayor.

El ancho se toma simplemente como la desviación típica en cada variable de los elementos que se clasifican en un cierto procesador.

Los procesadores lineales se ajustarán mediante minimización del error cuadrático de las salidas, es decir del término:

$$E = \sum_{t,\,i} \left[y_i(t) - y_i'(t) \right]^2 \qquad (3.5)$$

donde y_i son las salidas estimadas por la red y y'_i son los valores observados.

El objetivo ideal es llegar a un mínimo global de E.

E dependerá de los pesos de la red, por lo que estos son los parámetros a ajustar. El algoritmo de optimización deberá buscar el conjunto de pesos con el que el error es mínimo, para la muestra que se utiliza.

El gradiente de la función *E* no es sencillo de obtener exactamente, ya que la realimentación hace que los pesos correspondientes requieran un almacenamiento de los valores de las derivadas precedentes [17], lo que complicaría la estructura de la red. Una aproximación es no tener en cuenta los términos de dependencia por realimentación, como se hace en [56], pero esto puede no resultar válido. En el método que se propone se ha adoptado esta forma de actuar y, para complementarla se añade un ajuste estocástico que realiza una búsqueda aleatoria de amplitud descendente en media en el espacio de pesos. No se utilizan puntos consecutivos para estimar el gradiente, porque, al ir reduciéndose la distancia, aparecerían problemas numéricos.

Llamando $\boldsymbol{P}$ al conjunto de todos los pesos, la búsqueda estocástica, dada una posición inicial $\boldsymbol{P_1}$, consiste en generar otro conjunto $\boldsymbol{P_2}$ aleatoriamente, pero dentro de una cierta banda alrededor de $\boldsymbol{P_1}$. Si el error con $\boldsymbol{P_2}$ es menor, se toma esta y se repite el proceso; si no es así se genera un nuevo $\boldsymbol{P_2}$. La distancia de búsqueda se va reduciendo cada $2n_W$ pasadas sucesivas (donde n_W es el nº de parámetros en $\boldsymbol{P}$), según la expresión:

$$banda' = banda \bullet al$$

siendo *al* una variable aleatoria distribuida uniformemente en el intervalo [0.4, 1.4). Así la banda decrece en media, pero ocasionalmente puede aumentar, lo que le da capacidad para buscar mínimos globales. Con esta aproximación se incrementa el tiempo de cálculo necesario a costa de una mayor simplicidad en la estructura del algoritmo y una menor cantidad de memoria.

El algoritmo tiene más probabilidades de acercarse al óptimo global cuanto más suave sea la pendiente de la curva varianza-error, definida para un valor *er* como

$$ve(er) = \mathrm{var}(\mathbf{p}_i) \,/\, E\left[M\left(\mathbf{p}_i\right)\right] < er$$

es decir, da la varianza de todos los puntos del espacio de parámetros que proporcionan un error menor que uno dado con el modelo que se esté usando (este es constante). En esta curva los mínimos locales aparecen como saltos bruscos.

Si el tiempo de cálculo resultase crítico sería necesario ir a un planteamiento como el descrito en [17], que viene a requerir prácticamente otra red en paralelo para calcular la evolución de los pesos.

3.5.3. Esquema constructivo

Hasta el momento se ha definido una red estática, es decir, como si su estructura estuviese totalmente determinada. Así pues, queda por explicar el modelo de desarrollo y crecimiento de la red, es decir, la adaptación de su estructura al problema concreto.

Se pretende conseguir que el propio algoritmo sea capaz de ir refinando la topología, sin interacción con el usuario.

Siguiendo a [50], que aborda esta cuestión en el contexto de percepción, pensando en procesamiento visual, con validez de carácter general, para alcanzar la precisión requerida debe haber el suficiente número de conexiones (grados de libertad), por lo que o se parte de una arquitectura muy rica que luego se reducirá (enfoque sustractivo), o se parte de una pobre que se hará crecer (enfoque aditivo). Ambas posturas tienen partidarios y detractores.

El enfoque que se adopta aquí es el aditivo, es decir, se parte de estructuras sencillas que se irán complicando progresivamente. Este tipo de planteamiento es frecuente en ingeniería. Por otra parte, si ambos métodos han de llegar a la misma solución, se habrá trabajado con redes más simples, con lo que el ajuste progresará más deprisa, siguiendo la línea esquematizada en la fig. 3-3.

Parece, además, más sencillo acertar tomando como punto de partida un subconjunto de la arquitectura ideal que con un superconjunto. Además, en un modelo sobredimensionado el ajuste de parámetros con una muestra no conduce a estructuras con mayor eficiencia para otras muestras [20]. Esto se debe a que el número de observaciones necesario para dar con los valores correctos de los parámetros, es muy elevado [32].

En este trabajo se adopta como sistema de crecimiento de la red el aumento del número de procesadores, buscando una doble aproximación del orden del sistema y de la estructura o forma de la función de transferencia.

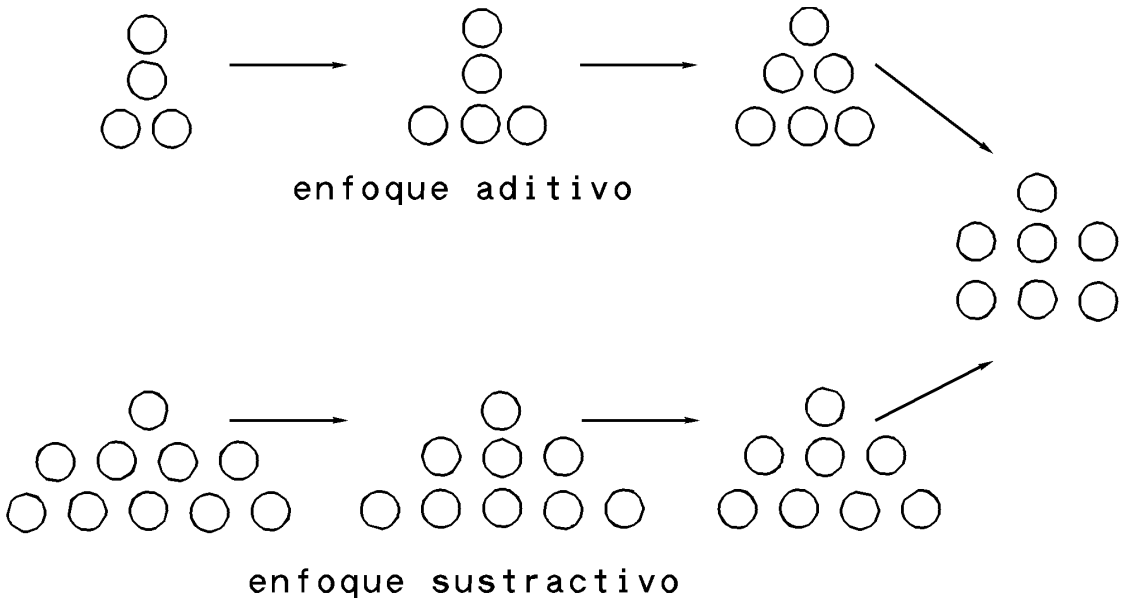

Fig. 3-3. Dos formas de llegar a una estructura. Por arriba la vía aditiva; por debajo la sustractiva.

3.5.3.1. Estimación del orden

En este apartado se trata la línea de crecimiento de la red que tiene por objeto aproximar el orden del sistema, esto es, hasta que desfase en las variables hay que considerar.

Objetivo

Sin considerar el ruido, la expresión desconocida que proporcionará el valor de una salida tendrá la forma

$$y = f(z_{ij}) \tag{3.6}$$

donde el subíndice *i* indica la variable y *j* indica el retraso, sin que en principio todas las variables tengan que estar presentes hasta los mismos retrasos, ni en cada una estar presentes retrasos consecutivos.

Variando los *j* se llegará a diferentes aproximaciones. Se trata de obtener el conjunto de $\{z_{ij}\}$ que hace mínimo el error residual, que se toma aquí como el expresado en la ecuación (3.5). Sea $\{z_{ik}\}$ este conjunto.

Si se tiene

$$\{z_{il} / l = 1,2,\ldots,c\} \text{ donde } c = cte \geq max\,(k) \Rightarrow \{z_{ik}\} \subset \{z_{ic}\}$$

con lo que el óptimo que se puede obtener es el propio $\{z_{ik}\}$.

Procedimiento iterativo

Para dilucidar cuáles han de ser esos valores previos (cuantos retrasos hay que incluir) no es posible partir del conjunto completo de ellos y eliminar los innecesarios, sino que se plantea un procedimiento paso a paso.

Sea en un cierto paso el modelo en el que intervienen las salidas $s_k(t_j)$ y las entradas $e_i(t_l)$,

$$R(s_k(t_j), e_i(t_l))$$

A partir de este se crea R' con un orden inmediato superior para todas las variables. Este R' se tomará como base en vez del R si

$$E_G(M_1) < E_G(M)$$

donde E_G es una estimación del error de generalización que se obtiene como el error de predicción en una muestra que no coincide con la de ajuste, como se detalla en el apartado 3.5.5.1.

Este planteamiento es análogo al ya mencionado para validar el orden de un modelo donde se estima el orden correcto como aquel a partir del cual el error cuadrático residual no decrece significativamente, con la diferencia de que aquí, en vez de plantear si el decrecimiento del error residual es significativo, se plantea el error de generalización, donde el orden que hace mínimo el error en generalización , como puede verse en el ejemplo de la fig. 3-4, es el que se usa para estimar el orden desconocido.

Hay que reseñar de todas formas, que como todos los algoritmos paso a paso, no se puede garantizar que vaya a alcanzar la solución correcta; este procedimiento es una opción que funcionará aceptablemente siempre que los retrasos que intervengan sean próximos unos a otros. Pues si la utilidad de los retrasos va disminuyendo, estos se acabarán rechazando, y el algoritmo no verá si más adelante vuelve a haber retrasos cuya contribución sea significativa.

3.5.3.2. Aproximación de la función

Planteada la problemática de los argumentos que intervienen, la siguiente cuestión a abordar es la estructura de la función *f* que aparece en la expresión (3.6).

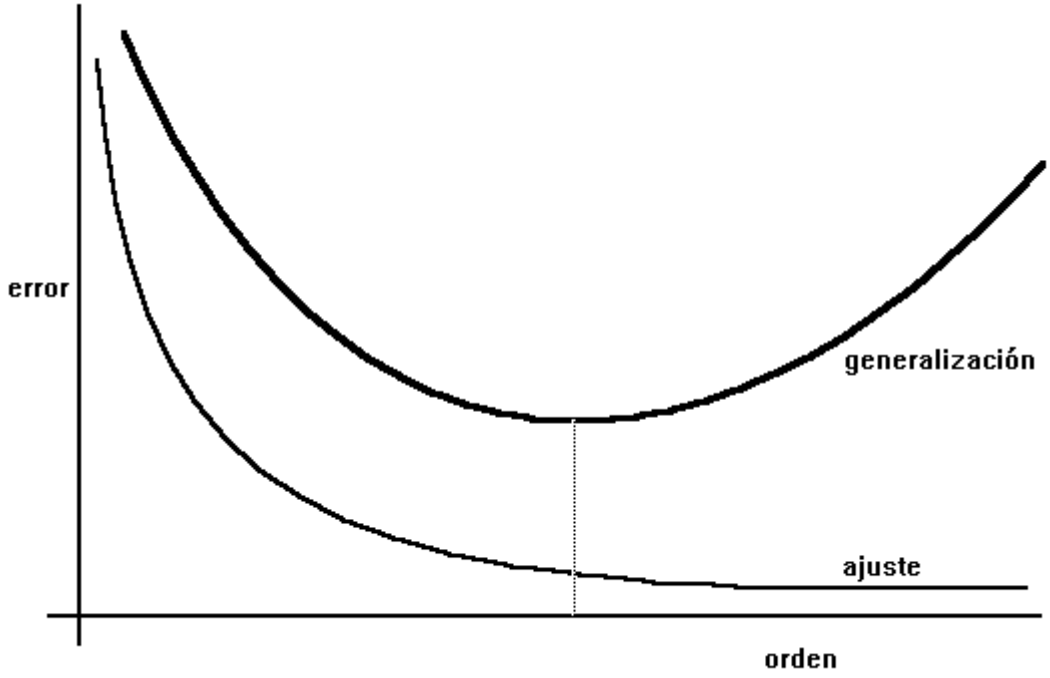

Fig. 3-4. Forma de la variación del error residual de ajuste y del de generalización al aumentar el orden del modelo.

Esquema de la aproximación

Tomando la expresión (3.6) como

$$y = f\left[y(t-i), x_j(t-k)\right]$$

o más abreviadamente

$$y = f(z_l)$$

donde z_l representan las variables genéricas, a las que por el momento no vamos a diferenciar, y desarrollando en serie de Taylor en el entorno de un conjunto de ellas que representado como el vector $\mathbf{z_0}$

$$f(\mathbf{z}) = f(\mathbf{z_0}) + (\mathbf{z}-\mathbf{z_0})\nabla f + O(|\mathbf{z}-\mathbf{z_0}|^2) = P + O(|\mathbf{z}-\mathbf{z_0}|^2)$$

donde $\mathbf{z}$ es un vector genérico cuyas componentes son las variables z_l y P es un hiperplano.

Si se adopta P como modelo en el entorno de $\mathbf{z_0}$, el error será:

$$E = f(\mathbf{z}) - P = O(|\mathbf{z}-\mathbf{z_0}|^2) \qquad (3.7)$$

Si se quiere acotar este error por debajo de un valor e,

$$E < e \Rightarrow O(|\mathbf{z} - \mathbf{z_0}|^2) < e$$

los puntos que verifiquen esta expresión estarán definidos por una expresión de la forma:

$$|\mathbf{z} - \mathbf{z_0}| < \partial(\mathbf{z_0}, e) \tag{3.8}$$

Luego la aproximación mediante P será válida en un entorno de $\mathbf{z_0}$ acotado por la expresión anterior.

Sea $\{D_i\}$ una partición del dominio C de f:

$$C = \bigcup D_k \,/\, D_i \cap D_j = \varnothing \text{ si } i \neq j$$

Para cada uno de estos subdominios D_i de la partición se fija $\mathbf{z_i}$ y se define m_i mediante

$$m_i = \max_j |\mathbf{z_i} - \mathbf{z_j}| \;/\; \mathbf{z_i}, \mathbf{z_j} \in D_i$$

Si se verifica

$$\forall m_i < \partial(\mathbf{z_i}, e) \tag{3.9}$$

según la ecuación (3.8) la aproximación mediante los correspondientes hiperplanos será válida en todo el dominio con un error por debajo del requerido (e). Para conseguir unos m_i reducidos es conveniente además elegir unos $\mathbf{z_i}$ tales que las distancias que intervienen no sean elevadas.

Procedimiento iterativo

Tomando como origen la partición trivial, es decir, la constituida por el propio dominio, se propone, como sistema para reducir el error, ir aumentando el número de subdominios, lo que hará descender los valores m_i, lo que conducirá a una aproximación mayor hasta la precisión requerida.

Desde el punto de vista de la red, este planteamiento equivale a aumentar el número de procesadores en la capa de clasificación.

No obstante, si un nuevo procesador se inicializa en forma aleatoria su contribución puede no ser óptima, lo que llevaría a un número muy alto de procesadores. Si su inicialización es aleatoria, la densidad, al final, tenderá a ser

uniforme en el espacio de las entradas, y, como la densidad necesaria viene condicionada por la zona de peor ajuste, se tendría una red alejada del óptimo, en cuanto a complejidad se refiere, ya que en otras zonas puede bastar una densidad mucho menor. El exceso de densidad reflejaría el hecho de que en el agrupamiento no se tuviera en cuenta la información relativa a su utilización posterior; es este un agrupamiento genérico, que siempre será menos eficaz que uno adaptado al propósito concreto.

Esto podría obviarse con un adecuado tratamiento a posteriori que combinara los procesadores, pero durante el ajuste obligaría a mantener un número innecesariamente elevado de ellos en operación, lo que conllevaría problemas de velocidad y de sobreajuste, por haber procesadores que se ajustarían con pocos puntos.

Por todo esto, se propone una solución alternativa. Dada la estructura de la red, según se ve en la ec. (3.2), por cada salida hay un procesador lineal asociado a uno de la capa de clasificación, como puede verse en la fig. 3-1. Así pues, planteándolo a la inversa, cada procesador de clasificación lleva asociados un procesador lineal. Durante el ajuste de la capa lineal de la red, será necesario trabajar con el error; al ser la capa de clasificación de respuesta binaria, un error viene asociado a un sólo procesador lineal, y por tanto a un procesador de la capa de clasificación. Entonces es posible almacenar en cada procesador de clasificación el error asociado a él. Así la partición del dominio llevará asociado unos errores e_i:

$$C = \bigcup D_i;\ e_i = \sum_{t_j / L_i(t_j)=1} \left[y(t_j) - l_i(t_j) \right]^2$$

Cuando, teniendo n procesadores, se añada uno nuevo, en vez de inicializar el centro aleatoriamente, como es la práctica usual, se inicializa próximo al que tenga el error asociado más alto, con lo que se asegura que aumenta la discretización del dominio en la zona más necesaria.

En cuanto al criterio de error máximo para subdivisión, teniendo en cuenta que el error medido incluye la varianza propia de los datos, y que este término no debe reducirse, el enfoque a seguir es tomar el error medio, no el total, ya que se busca que quede sólo como error el ruido propio de la muestra. Admitiendo que el ruido sea de amplitud constante, para minimizar el error medio total hay que minimizar el del subdominio que lo tenga máximo, mientras que el total no implica necesariamente máximo, ya que

$$N_1 E_y + N_1 E_1 > N_2 E_y + N_2 E_2 \Rightarrow \frac{N_1}{N_2} > \frac{E_y + E_2}{E_y + E_1}$$

siendo E_y el error debido al ruido y E_1 y E_2 los debidos al ajuste.

Llamando a los centros $\mathbf{M_i}$ para cada D_i, para el nuevo procesador se tiene:

$$\mathbf{M_{n+1}} = \mathbf{M_k} + \partial \qquad (3.10)$$

siendo D_k el dominio con mayor error asociado en esta etapa.

Para conseguir el efecto antedicho interesa:

$$D_k^n \approx D_k^{n+1} \cup D_{n+1}^{n+1}$$

donde el subíndice se refiere al subdominio y el superíndice a la etapa en la construcción del modelo.

Si hacemos:

$$\mathbf{a_i} = \mathrm{var}(\mathbf{z_j}) \ / \ \mathbf{z_j} \in D_i \qquad (3.11)$$

es decir, asignamos a las componentes del vector $\mathbf{a_i}$ la varianza de las correspondientes componentes de los puntos $\mathbf{x_j}$ del dominio D_i

Podemos tomar en la ec (3.10)

$$\partial = \frac{\sqrt{\mathbf{a_k}}}{\mathbf{2}}$$

donde $\sqrt{\mathbf{a_k}}$ representa a un vector de componentes la raíz cuadrada de los de $\mathbf{a_k}$.

Así pues la clasificación es dependiente de tarea en su crecimiento.

Esta mecánica es parecida a la partición recurrente, y resulta similar al método propuesto en [25] para estimar la varianza de los datos.

Convergencia

El hecho de que los argumentos pertenezcan a secuencias temporales hace posible la existencia de relaciones entre ellas, lo que puede conducir a una representación que no será única, sino dependiente del algoritmo seguido para

buscarla, como se ha comentado en el apartado de estimación del orden, no obstante, la aproximación está garantizada, ya que siempre se podrá conseguir que se verifique la expresión (3.9). Los requisitos necesarios para llegar a ella son que el número de puntos en que la función no sea continua sea finito. La reducción de distancias siempre será posible, puesto que en último término, cada punto de la muestra puede configurar un dominio independiente.

En general la unicidad de la expresión no existe si para alguna de las variables que intervienen hay una relación de tipo autorregresivo como:

$$y_t = \sum a_i y_{t-i}$$

3.5.3.3. Combinación de las líneas de crecimiento

Han quedado expuestos, pues, los mecanismos de crecimiento de la red, que implican algoritmos paso a paso. Estos se combinan comparando el error de generalización de las posibles ampliaciones y eligiendo aquella que produzca un aumento mayor de precisión.

Las posibles ampliaciones son de dos tipos:

- aumento de la capa de clasificación, lo que implica el correspondiente aumento en la capa lineal

- incremento de los retrasos en las variables de entrada a la red

En este último grupo cabría hacer una diferenciación, según cuál o cuales sean las variables afectadas por el incremento.

Dado que no parece conveniente, por eficacia del algoritmo, tener un número elevado de redes en comparación, máxime cuando estas pueden tener ya una cierta complejidad se ha tomado una vía intermedia, que divide a las variables en los grupos de entradas y salidas, que ha proporcionado el usuario, por lo que quedan tres posibles pasos a competir, tomándose en cada momento el mejor.

Para evitar que se produzca interferencia entre las tres líneas de acción, se reajusta la división del dominio en cada red, con lo cual, no puede salir perjudicada la estimación del orden, ya que su efecto es el de tener un modelo más preciso.

3.5.4. Características de este tipo de red

Este tipo de red es análogo al utilizado por [73], [82], [90] y [54] entre otros; presenta la ventaja de necesitar menos capas para la aproximación de funciones.

En [73] se propone una red, de arquitectura fija, que se diferencia de la expuesta aquí en que los términos lineales se limitan a constantes. Dos son los tipos de ajuste mencionados en [73]: aprendizaje totalmente supervisado, en el que se minimiza el error cuadrático variando centros, anchos y las constantes, y un aprendizaje híbrido en que el ajuste de centros y anchos es no supervisado, y las constantes son las que minimizan el error cuadrático. Los centros se obtienen minimizando:

$$M_a(\mathbf{x})(\mathbf{x}_a - \mathbf{x})^2$$

donde M_a es una función de pertenencia; también se propone el algoritmo

$$d\mathbf{x}_a = 0.03(\mathbf{x}_i - \mathbf{x}_a)$$

Los anchos se ajustan en forma heurística para conseguir un cierto solape (hasta el segundo vecino más cercano)

Las ventajas aducidas en [73] son:

• aprendizaje lineal, luego rápido.

• la localización puede hacer que se necesiten menos requisitos de computación y dar por eso más rapidez.

Según [72] el ajuste de este tipo de redes es mucho más rápido que el de las de varias capas de respuesta sigmoidal. Según [90] la precisión con redes similares a la propuesta es mayor que la obtenida con otros tipos cuando se trata de predecir la evolución de un sistema caótico.

En [53] se presenta un algoritmo de la misma clase, consistente en un conjunto de redes que hacen las veces de lo que en el modelo propuesto sería cada procesador lineal, y otra red que elige la más adecuada de las anteriores en función de las entradas, lo que se corresponde con la capa de clasificación. Se trata de un enfoque en la misma línea, aunque planteando el sistema de elección en forma probabilística.

En [82] se presenta la conexión de este tipo de redes (sustituida la combinación lineal por una constante) con la teoría de regularización, donde una red que responde a una expresión del tipo:

es una aproximación al mínimo del funcional:

el cual representa un ajuste de mínimos cuadrados con penalización de oscilaciones.

Las G_i son funciones radiales, es decir:

$$G_i(x,t) = G_i\left(\|x-t\|\right)$$

con x el término independiente y t el "centro" de G_i, y

W representaría la influencia de distintos tipos de entradas. En el caso simple de W diagonal, lo que hay es una ponderación de cada coordenada, que es el enfoque propuesto en esta tesis.

En [82] se manifiesta que este tipo de redes tienen mejores propiedades aproximantes que un perceptrón multicapa.

En [54] se presenta redes con combinación lineal tomando la forma (caso unidimensional):

El término lineal tiene la forma de una aproximación de Taylor de primer orden en el entorno del punto centro de cada ρ_j, que son funciones decrecientes con la distancia a su centro. Un ejemplo de estas funciones son las gaussianas:

Posibles limitaciones de las funciones de base radial se discuten en [47]. Una de ellas es que la propia localización, las puede hacer sensibles a variables irrelevantes. También resulta necesario un cierto número de ellas para cubrir un espacio. Estos inconvenientes se solucionan en el método propuesto con la

sustitución de las funciones de base radial por una clasificación, que se puede hacer en tantas zonas como se desee, amplias o no.

Además un modelo tan sencillo como el lineal conlleva obviamente, por su propia estructura matemática [99], que los entornos próximos se transforman en entornos próximos.

Cara a una posible implementación en soporte físico, hay que reseñar que estas redes se pueden resolver con menos precisión numérica que las convencionales, por su menor complejidad, ya que no existen capas ocultas propiamente dichas. En una red de 2 capas ocultas la información es procesada 3 veces, dependiendo los resultados de cada una de los de la anterior. En el esquema propuesto hay dos procesos: uno de clasificación y otro de combinación lineal, pero los cálculos de cada uno no dependen numéricamente del otro.

Dentro de las representaciones con realimentación usadas en PPD, esta es similar a la presentada en [56] con la salvedad de que puede haber varios retrasos afectando a la realimentación. En [34] y [76] se proponen esquemas de realimentación de la primera capa oculta. Como se puede ver, para la topología utilizada no va a haber una distinción tan marcada entre capa oculta y capa de salida, pero, en cualquier caso, la realimentación propuesta consiste en tomar valores previos de la salida de la red, no el de unidades realimentadas.

La distinción surge de que en estos modelos existe una capa oculta, cuya aportación ha sido sustituida en el aquí propuesto por una serie de retrasos de las variables utilizadas.

Véase en la tabla 3-1 la comparación entre estos modelos.

3.5.5. Control de extrapolación

3.5.5.1. Estimación de la capacidad de generalización

Es necesario en este momento definir como se va a hacer la comparación de la capacidad predictiva de distintos modelos.

Se propone trabajar con sólo una parte de la muestra para el ajuste y realizar las comparaciones sobre el error predictivo del total (validación cruzada).

La idea es coger el mínimo en error de generalización, tal y como se presenta en la fig. 3-4. Así el algoritmo avanzará cogiendo la ampliación que reduce el error de validación cruzada, hasta llegar al punto en que este aumenta, momento en el cual se da por bueno el modelo. También se da por bueno si se ha alcanzado la precisión requerida por el usuario.

Para que la validación cruzada sea fiable, la muestra de ajuste debe ser representativa del sistema, para lo cual conviene separar una parte que contenga el máximo de información, es decir, la llamada persistencia en la excitación, esto es, un barrido completo en amplitudes y frecuencias de las señales de entrada.

Una vez que se ha llegado a la red que se estima mejor, o suficientemente buena, se hará el ajuste final con la muestra completa, para incluir la máxima información antes del uso definitivo.

3.5.5.2. Eliminación de información irrelevante

Si la información que contenga alguna de estas variables no fuera relevante, sería conveniente que el propio algoritmo fuese capaz de detectar y eliminar las dependencias innecesarias.

Para ello se propone un algoritmo sustractivo del tipo de penalización de complejidad, como pueden ser los métodos de eliminación [92] o decaimiento de pesos [49]. El objetivo es que los parámetros que no mejoren el ajuste se anulen.

Tabla 3-1. Diferencias entre el método propuesto y otros usados en el tratamiento de secuencias.

Red y referencia	**Realimentación**	**Función de cada procesador**	**Criterio de generación de red**	**Ajuste**
Filtro [5]	Entrada	Monótona	NO	Gradiente
Híbrida [2]	Entrada	Local	NO	No supervisado + gradiente
CNLS [1]	Entrada	Local+lineal	NO	Gradiente
Contexto [1]	Interna	Monótona	NO	Gradiente aproximado
Cascada [2]	Individual no ajustable	Monótona	NO	Gradiente
RTRL [5]	General	Monótona	NO	Gradiente
Realimentada [1]	Salida	Monótona	NO	Gradiente aproximado
Desarrollada [2]	General	Monótona	NO	Gradiente
Correlación en cascada [1]	Individual	Monótona	Correlación con error residual	Gradiente
BPTT [2]	General	Monótona	NO	Gradiente
Enfocada [2]	Entrada, individual	Monótona	NO	Gradiente
Propuesta	Entrada, salida	Local+lineal	Error de generalización	Estocástico Gradiente aproximado

A esto hay que añadir la propia tendencia del ajuste por mínimos cuadrados a eliminar los pesos irrelevantes.

La forma en que se aplicarán estos términos se discute en el apartado 3.6.3.

3.5.6. Uso del modelo

3.5.6.1. Simulación

Tras el ajuste, la red se archiva para uso posterior. Este consistirá en hacer pasar unas secuencias por las entradas y obtener las salidas.

Puesto que el modelo conlleva la utilización de valores retrasados, sólo es aplicable cuando se conocen los anteriores al punto de inicio en número suficiente. Se supone que el punto inicial es una situación estacionaria, y por lo tanto los valores anteriores son iguales.

3.5.6.2. Interpretación

Uno de los objetivos del presente trabajo era conseguir una cierta interpretabilidad del modelo.

La estructura final consistirá en una serie de dominios y una combinación lineal en cada uno, a partir de una serie de valores retrasados de las variables, lo cual cumple con facilidad el objetivo propuesto.

Los retrasos que afecten a las variables indican el tipo general de sistema de que se trate, que se concretará por los coeficientes de la combinación lineal resultante.

Así pues, el modelo consiste en una colección de representaciones lineales tipo dominio-z cada una en el entorno de un cierto punto de funcionamiento. La estructura de los subdominios define estas zonas en que es aplicable cada modelo lineal.

3.6. Variantes

En el modelo planteado quedan aún ciertos aspectos por comentar. En este apartado se aborda su planteamiento presentando distintas posibilidades, cuyo comportamiento e incidencia se estudia mediante ejemplos sencillos.

3.6.1. Continuidad

Un inconveniente que se puede plantear a este tipo de métodos es la falta de precisión por la poca representatividad en las discontinuidades. Este es un problema que se da con la partición recurrente [74], como se destaca en [41]. Para aislar posibles efectos del orden, este se suprime en los ejemplos de este apartado.

Como ya se ha mencionado, la falta de continuidad en la respuesta permite independizar el ajuste en cada parte disjunta. La respuesta continua, en cambio, será más precisa en el caso de que la función incógnita también lo sea.

A continuación se plantean distintas opciones para atacar este problema y se comparan mediante un ejemplo, para elegir la vía más apropiada.

3.6.1.1. Opciones

Si el problema es el aspecto de los resultados para uso posterior, su solución puede abordarse en una fase de post-proceso, mientras que si la dificultad es la precisión, esto debe tratarse dentro del propio algoritmo.

Una solución natural a este problema, usada en [72], [54] y [82], es utilizar funciones de base radial en vez de una clasificación binaria. De las varias posibilidades que se ofrecen en [39], a continuación se consideran dos:

Función bicuadrática: $f(x) = \sqrt{d^2 + \rho^2}$

Función gaussiana: $f(x) = e^{-d^2}$

donde d es la distancia al centro y ρ es un parámetro.

Para una amplia discusión de las ventajas e inconvenientes de distintas funciones de interpolación que se pueden utilizar en este contexto, véase [39].

Una solución distinta, es la ya comentada del post-proceso. En este caso se partiría de un modelo generado en forma discontinua y se representaría de forma continua. Para evitar los efectos de paso de una línea a otra se puede interpolar entre líneas próximas. Esto recuerda a la regresión de los k puntos más próximos, pero dejando libre el valor de k. El algoritmo se basa en:

$$y = \frac{\sum a_k(\mathbf{x}) R_k(\mathbf{x})}{\sum a_k(\mathbf{x})}$$

siendo R_k cada uno de los ajustes de cada subdominio y

$$a_k(\mathbf{x}) = 1 \text{ si } dist(\mathbf{x}, \mathbf{M_k}) < Q$$
$$a_k(\mathbf{x}) = 0 \text{ si } dist(\mathbf{x}, \mathbf{M_k}) > Q$$

siendo $\mathbf{M_k}$ el centro del subdominio D_k

Se toman como distancia $dist(\mathbf{x},\mathbf{M_k})$ y como límite Q los siguientes valores:

$$dist(\mathbf{x},\mathbf{M_k}) = \sum \frac{(x_i - m_{ki})^2}{\sigma_{ki}^2}$$

$$Q = 4n_i$$

siendo n_i el número de variables, esto es, la dimensión del espacio de entrada.

3.6.1.2. Resultados de la prueba

Aplicadas las dos opciones primeras a una función continua no lineal, cual es el seno, muestreada en un periodo completo y con un ruido uniforme en el intervalo (-0.05, 0.05), se obtuvieron los resultados que refleja la tabla 3-5, que incluye a efectos de comparación la clasificación binaria propuesta inicialmente.

Tabla 3-5. Ajustes conseguidos sobre la función sen(x) con un 5% de ruido uniforme

	Error residual	Nº de funciones
Bicuadrática	0.028	6
Gaussiana	0.029	9
Clasificación binaria	0.029	9

Todos los modelos son capaces de estimar el ruido de la muestra, pues el error residual obtenido es aproximadamente el valor de la desviación típica de la distribución uniforme utilizada. Se observa que el número de procesadores necesario puede ciertamente disminuirse utilizando funciones continuas en lugar de las de clasificación binaria, y que, como ya se señala en [39] los mayores beneficios se consiguen con funciones básicas no locales, como es la función bicuadrática. Al ser uno de los requisitos aquí planteados la

interpretación posterior del ajuste, no parece esta una opción especialmente recomendable, ya que conduce a una mezcla de funciones.

Por otra parte, en la fig. 3-6 pueden verse los resultados utilizando para este ejemplo el algoritmo de post-proceso comentado anteriormente.

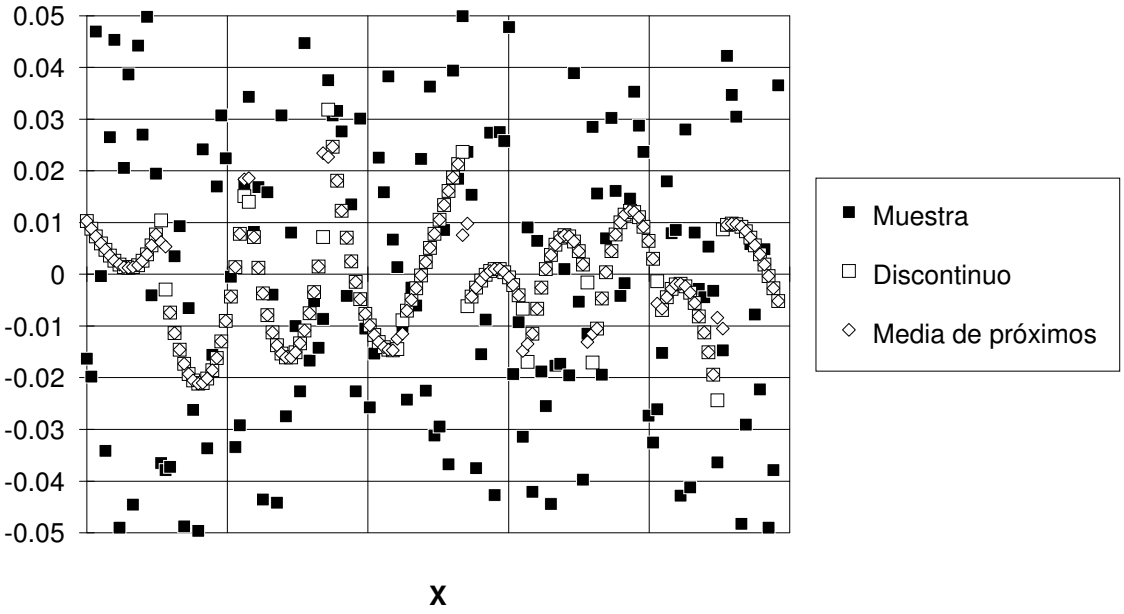

Fig. 3-6. Error en el ajuste de sen(x). La señal original (utilizada para el ajuste) incorpora un 5% de ruido. La media de próximos es el resultado obtenido con el algoritmo propuesto.

Cabe decir que el progreso obtenido no es espectacular, pero esto se debe a que el error al que puede llegar el método en su forma original discontinua es suficientemente pequeño de por sí.

Por lo anterior, no parece justificado incluir funciones continuas durante el ajuste. Sin embargo, para una mejor representación de los resultados puede ser útil el post-proceso propuesto.

3.6.2. Adaptación de los dominios durante el ajuste

Tal y como se ha descrito la capa de clasificación se hace crecer de forma que vaya siendo más óptima para la tarea de predicción. No obstante en principio, con el algoritmo básico, esta adaptación queda restringida a la posición inicial del centro nuevo. Sería interesante realizar una adaptación posterior de los dominios, para optimizar más su distribución. Así se obtendría

una adecuada falta de sensibilidad a variables irrelevantes [47] y una mayor precisión.

3.6.2.1. Opciones

Los dominios se identifican por sus centros y anchos. Su optimización podría plantearse si se comprueba, tras un ajuste inicial, que para determinado punto un procesador consigue mejor ajuste que otro, incluyendo este punto en el dominio asociado a aquel. Esto conduciría a un reajuste de los dominios en función de la precisión, en vez de la simple proximidad geométrica. Surgiría así un problema que consiste en que al ajustar sólo por criterios de mayor precisión se crean zonas inconexas, que no se corresponden con la filosofía del método (fig. 3-1). Así pues, para que el ajuste de dominios sea acorde con la estructura de estos, la ampliación debe hacerse con puntos contiguos.

Fig. 3-7. La aproximación 1 es más precisa que la 2 en las zonas A y B. Sin embargo en la zona A la interpretación cualitativa más correcta es la obtenida con 2, ya que 1 indica que la curva es decreciente, cuando es lo contrario.

Puesto que los parámetros que identifican los dominios son de dos clases: centros y anchos, son planteables dos estrategias:

- Mantener los centros y reajustar los anchos

- Reajustar los centros y calcular nuevos anchos por proximidad media geométrica, ya que esto mantiene el sentido físico de que la aproximación lineal se corresponde con un ajuste localizado.

Si la clasificación es binaria, como aquí se ha presentado, para ir variando los valores de los parámetros, no puede plantearse un método de gradiente. En este caso hay que recurrir a un planteamiento aproximado. Una alternativa es apoyarse en el mismo método utilizado para generar nuevos elementos de clasificación. Según él, se ampliaría el ancho del dominio si el error es pequeño, porque indicaría que se está en una zona que es bastante aproximable. El algoritmo sería del tipo:

$$\Delta\sigma = \frac{k}{1 + err^2}$$

En la línea de "concentrar" los elementos de proceso con la idea de ir moviendo sus centros hacia las zonas de mayor error se plantea un algoritmo del tipo:

$$\Delta \mathbf{M_i} = k(\overline{\mathbf{M}} - \mathbf{M_i})$$

con $\overline{\mathbf{M}}$ siendo la media ponderada de los centros actuales con sus errores.

En cualquier caso se mantiene el ajuste independiente, para evitar el efecto de interferencia comentado en [37]. Esto se debe a que si se está ajustando el dominio y la linearización a la vez es posible que los coeficientes a que se está ajustando la segunda sean válidos para la configuración actual del primero, pero no para la configuración definitiva.

Hasta ahora, en el algoritmo, al ajustar una nueva red, resultado de añadir un procesador a la anterior, se reinicializan los anchos a valor unidad; esto da flexibilidad al conjunto, ya que no impone un punto de partida que no tiene porque ser óptimo para la nueva arquitectura.

Si esto no se hace así existirán para cada procesador unos anchos iniciales y hay que definir un punto de partida para el nuevo. Retomando la línea del apartado 3.5.3.2, sería

$$\boldsymbol{\sigma}_{\mathbf{n+1}} = 0.5\boldsymbol{\sigma}_k$$

lo que puede tener una representación como la que se indica en la fig. 3-8.

El paso de una red a otra, que supone el avance del modelo viene condicionado por el centro y ancho de partida de cada subdominio. Se han comentado anteriormente dos variantes en cuanto al ancho: conservar los anteriores o reinicializarlos. En cuanto a los centros, dado el algoritmo propuesto para la creación de nuevos centros, no tendría sentido reinicializarlos.

Fig. 3-8. Creación del subdominio n+1 a partir del j

Puede no resultar interesante, por motivos de interferencia, ajustar conjuntamente el agrupamiento y las funciones lineales en cada subdominio. Por ello se estudian además resultados de esta forma de actuar al final, con la cantidad de subdominios que se ha dado por definitiva.

3.6.2.2. Resultados de las pruebas

A efectos de comparación se analiza un caso en que intervenga la subdivisión, es decir, un caso no lineal, y para evitar que los algoritmos, a causa del ruido y de tomar distintos pasos, difieran en el orden, lo que haría complicada la comparación, se anula en el modelo la capacidad de generar retrasos.

Sea un sistema de una entrada y una salida:

$$y(t)=sen[x(t)] + ruido$$

El ruido es el mismo que el del ejemplo utilizado en el apartado 3.6.1.2.

La comparación se basa en los resultados de precisión alcanzados y en el número de procesadores de clasificación que han sido necesarios. Se deja continuar el ajuste hasta que lo limite el propio control de extrapolación. El error residual y el número de subdominios alcanzado se presenta en la tabla 3-9.

Tabla 3-9. Comparación entre algoritmos que incluyen supervisión en distintos grados en el agrupamiento. CONSTANTE incluye agrupamiento independiente del ajuste posterior. En ANCHO la posición de los centros es independiente y los anchos son dependientes. En CENTRO ambos parámetros varían con el ajuste.

	error residual	zonas
CONSTANTE	0.029	7
ANCHO	0.029	8
CENTRO	0.049	3

Como puede verse el ajuste variando sólo el ancho no introduce un cambio significativo, mientras que el movimiento de los centros interfiere con el ajuste de los términos lineales, lo que conduce a un resultado no óptimo, si bien está claro que para usar sólo 3 elementos se alcanza una precisión mayor que la que se hubiera obtenido con un agrupamiento independiente del ajuste lineal.

La tabla 3-9 incluye unos resultados en los que, al proceder a ajustar una nueva red, resultado de añadir un procesador a la anterior, se reinicializan los anchos En la tabla 3-10 se presentan los resultados de simulaciones en las que

se ha variado el criterio de reinicialización de los anchos, utilizando agrupamiento no supervisado.

Como puede observarse el conservar los anchos puede tener repercusiones en los resultados; favorables cuando la asimetría de la función no está reflejada en los datos y desfavorable cuando, por no ser necesario, interfiere con el ajuste lineal.

Utilizando el criterio de no reinicializar anchos en cada fase la tabla 3-9 se convierte en la 3-11.

Puede verse como los resultados son peores, ya que no se permite a cada nueva red redistribuir adecuadamente los anchos. El efecto de interferencia entre el ajuste lineal y la supervisión del agrupamiento se mantiene.

El ejemplo adecuado para subrayar la diferencia entre un agrupamiento supervisado y otro no supervisado debe ser tal que la división óptima sea distinta de la distribución de los puntos de entrada. Por ejemplo, si la distribución de estos es uniforme, la división óptima no debe serlo. Un caso sencillo de este tipo es el que se propone a continuación. Se toma una distribución de entradas, x, uniforme entre 0 y 1 y se intenta ajustar la función:

$$y = \begin{cases} 5x \text{ si } 0 \le x \le 0.2 \\ 1.25(1-x) \text{ si } 1 \ge x > 0.2 \end{cases}$$

Tabla 3-10. Error residual y nº de subdominios alcanzado en distintos ejemplos variando los anchos iniciales de cada etapa.

	A	B	C	D
inicializando ancho	0.028 10	0.013 4	0.028 2	0.12 2
conservando ancho	0.029 9	0.0039 3	0.036 2	0.12 4

A: sen(x) + 5% de ruido

B: quebrada de dos tramos no centrada en el dominio

C: quebrada de dos tramos centrada en el dominio+ 5% de ruido

D: quebrada de dos tramos centrada en el dominio+ 20% de ruido

Tabla 3-11. Variación de los resultados del ajuste del seno con el grado de supervisión en el agrupamiento. Los nombres son los utilizados en la tabla 3-9

	error	zonas
CONSTANTE	0.059	4
ANCHO	0.064	5
CENTRO	0.069	4

Esta función es continua e idealmente el ajuste estaría sólo limitado por la precisión numérica, ya que los dos tramos son rectos. Para potenciar este efecto no se añade ruido.

Un agrupamiento que ignore los valores de *y* y se base sólo en los de *x* tenderá a distribuir los subdominios uniformemente, y por tanto será muy raro que aparezca una división justamente en 0.2, que es la posición óptima, limitándose el error al subdominio que incluya este punto, ya que se estará tratando de ajustar dos tramos rectos con uno. La precisión sólo puede aumentar en este caso por la sucesiva partición de este subdominio.

En la tabla 3-12 se presentan los resultados de este ejemplo, incluyendo la supervisión del agrupamiento solamente en el ajuste final.

Tabla 3-12. Ajustes conseguidos sobre dos quebradas.

	Error residual	Ecuaciones ajustadas
No supervisado	0.013	4.98x 1.18-1.07x 1.25(1-x) 1.28(1-x)
Anchos	0.00083	4.99x 1.25-1.24x 1.23(1-x) 1.25-1.24x
Centros y anchos	0.00097	4.99x 1.22(1-x) 1.25(1-x) 1.25-1.26x

Como se puede advertir, los resultados muestran que los algoritmos han tenido éxito en alcanzar una división más óptima cara al ajuste.

De todo lo anterior podemos extraer las conclusiones que se exponen a continuación.

La interferencia entre los ajustes no quiere decir que el agrupamiento no deba incorporar supervisión, ya que en la ec. (3.7) el término $O(|\mathbf{z}-\mathbf{z_0}|^2)$ que define el error depende de las segundas derivadas en el subdominio. Así:

$$E = \sum e_i = \sum |y - y_i|^2 = \sum O(|\mathbf{z} - \mathbf{z}_0|)^2 = \sum \left(\mathbf{z} - \mathbf{z}_0\right)^T H(\mathbf{z} - \mathbf{z}_0)$$

siendo H la matriz de derivadas segundas de la función desconocida en un punto intermedio entre $\mathbf{z}$ y $\mathbf{z_0}$. Luego la magnitud del término de error viene controlada por los dos factores $|\mathbf{z}-\mathbf{z_0}|$ y H, lo que quiere decir que eligiendo subdominios en los que los términos de H sean favorables, se pueden permitir términos $|\mathbf{z}-\mathbf{z_0}|$ mayores.

No obstante esta combinación no es necesario realizarla durante el ajuste, es decir, para modelos subóptimos por lo que se deja para el final con las muestras completas, como en el caso mostrado en la tabla 3-1.

En este caso, el ligeramente peor comportamiento de la variante que optimiza la distribución de centros y anchos puede interpretarse como un efecto de interferencia entre ambas distribuciones. No obstante es conveniente, sobre todo de cara a casos multidimensionales mantener la capacidad de ajustar la posición de los centros, ya que hacerlo sólo con los anchos puede no ser suficiente.

3.6.3. Eliminación de conexiones

Como ya se ha comentado puede resultar necesario un control del valor de los pesos para evitar que se consideren variables, o retrasos en una determinada variable, que sean irrelevantes para cierto procesador; este control actuaría como mecanismo de supresión de las dependencias introducidas pero no necesarias, dentro del proceso paso a paso de selección de orden.

3.6.3.1. Opciones

A efectos de tener una comparación experimental se plantean a continuación varios métodos en un caso de prueba. Cada uno añade un término

a la función de error que se está optimizando. Los métodos usuales y los términos correspondientes son:

Decaimiento [49]:

$$\lambda \sum pesos^2$$

Eliminación [92]:

$$\lambda \sum \frac{peso^2}{1 + peso^2}$$

Como puede verse en [92] estos términos se pueden contemplar como una determinada distribución a priori de los pesos. En concreto el término de decaimiento supone una distribución normal de pesos centrada en 0 (con varianza *1/2*λ en la expresión propuesta) y el de eliminación, una combinación de una distribución uniforme y otra del tipo normal, centradas ambas en 0. Cambiando pesos por error podemos tomar un planteamiento análogo al de [75], y definir otros términos.

Valor absoluto: $\lambda \sum |peso|$

Intermedio: $\lambda \sum peso \ \tanh(peso)$

Desde un punto de vista bayesiano el penúltimo término equivale a suponer que la distribución a priori del valor absoluto de los pesos es una exponencial.

El último término es intermedio entre el de valor absoluto y el de decaimiento, ya que tiende a ser cuadrático para pesos pequeños y lineal para grandes. No se ha planteado en función de una distribución a priori determinada, si bien será intermedia entre la normal y la exponencial, del tipo de la de Huber.

Los términos sustractivos que van a afectar a los pesos por cada método se muestran en la fig. 3-13 en función del peso.

3.6.3.2. Resultados

En lo que se refiere a variables irrelevantes, los distintos métodos se probaron con una curva compuesta de dos rampas rectas, de forma que dos

coeficientes correspondientes a variables (procesadores en la capa de entrada) distintas fuesen nulos. Los resultados se presentan en las tablas 3-14 y 3-15.

En lo que se refiere al orden, para un sistema del tipo:

$$y_n = 0'5(y_{n-1} + x_{n-2})$$

en la tabla 3-2 se muestran los resultados utilizando las técnicas de eliminación durante todo el desarrollo del proceso

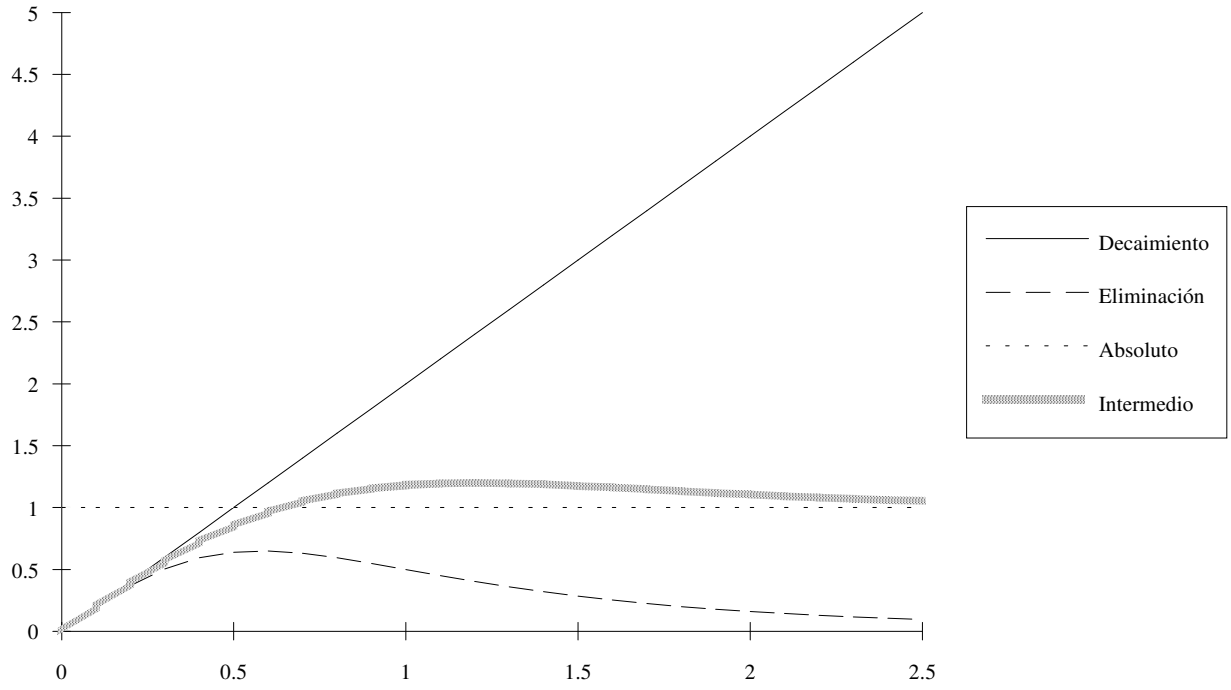

Fig. 3-13. Términos que entran restando al peso según los respectivos métodos. En abscisas el peso. El parámetro λ se supone 1.

Más difícil es estimar el orden correcto cuando el sistema tenga un retraso importante. Para el caso de

$$y_n = 0.5(y_{n-1} + x_{n-6})$$

los resultados de los distintos modelos sólo muestran que el error residual se ajusta al ruido de la muestra, pero ningún resultado se parece a la ecuación de partida. Esto es debido a que se consigue una estimación subóptima en el proceso incluyendo términos de autodependencia.

Tabla 3-14. Resultados de los distintos métodos de eliminación en un ejemplo con 5% de amplitud de ruido.

Término	λ	error residual	coeficiente 1	coeficiente 2
Decaimiento	0.01	0.034	0.17	0.010
Absoluto	0.01	0.029	0.027	0.0048
Eliminación	0.01	0.029	0.033	0.0046
Intermedio	0.01	0.031	0.13	<0.001
Ninguno	NO	0.028	0.0046	0.0015
Decaimiento	0.0001	0.028	0.0025	<0.001
Absoluto	0.0001	0.028	0.010	0.0012
Eliminación	0.0001	0.028	0.020	<0.001

Tabla 3-15. Resultados de los distintos métodos de eliminación en un ejemplo con 20% de amplitud de ruido.

Término	λ	error residual	coeficiente 1	coeficiente 2
Decaimiento	0.01	0.12	0.055	>0.2
Absoluto	0.01	0.12	0.072	0.048
Eliminación	0.01	0.12	0.034	0.090
Intermedio	0.01	0.12	0.053	>0.2
Ninguno	NO	0.12	0.019	0.017
Decaimiento	0.0001	0.12	0.021	0.015

Como conclusión de las pruebas anteriores cabe extraer las ideas que se indican a continuación.

No parece imprescindible en lo que a variables irrelevantes se refiere incluir los términos de eliminación en el ajuste. En parte esto puede ser debido a que los modelos no son redundantes, dado el enfoque aditivo planteado en la construcción.

Tabla 3-2. Resultados obtenidos con distintos términos de regularización mantenidos durante todo el proceso

Método de regularización	Error residual	Modelo(s) obtenido(s)
Ninguno	0.0051	r $0.59x_{n-2}+0.045x_n-0.12x_{n-1}+0.48y_{n-1}$ r $0.19x_{n-2}-0.19x_n+0.44x_{n-1}+0.55y_{n-1}$
Decaimiento	0.0013	r $0.54x_{n-2}+0.024x_n-0.060x_{n-1}+0.49y_{n-1}$
Eliminación	0.0013	r $0.55x_{n-2}+0.026x_n-0.063x_{n-1}+0.49y_{n-1}$
Valor absoluto	0.0016	r $0.55x_{n-2}+0.026x_n-0.066x_{n-1}+0.49y_{n-1}$

Los sistemas de regularización no eliminan totalmente los pesos, si bien reducen el valor de los innecesarios. Una estrategia del tipo eliminación parece adecuada, pero sigue siendo necesaria la interpretación del usuario.

En lo que se refiere a retrasos importantes o tiempos muertos, lo más apropiado es proporcionar el retraso inicial en los datos que utilice el programa, a partir de estudio de correlaciones, por ejemplo, de la misma manera que se hace habitualmente al identificar la función de transferencia en los métodos convencionales, retrasando la variable independiente hasta que la primera correlación sea significativa.

REFERENCIAS

[16] Ash, T. "Dynamic Node Creation in Back-Propagation Networks", Technical Report 8901, Institute for Cognitive Science, University of California, San Diego (1989)

[17] Back, A. D., Tsoi, A.C., "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling", Neural Computation 3, 375-385 (1991)

[18] Barron, A. R. y Xiao, X., Discussion en "Multivariate Adaptive Regression Splines" (J. H. Friedman), The Annals of Statistics, Vol. 19, No. 1, 67-82 (1991)

[19] Barry, D. "Nonparametric Bayesian regression". Ann. Statist. 14, 934-953 (1986).

[20] Barto, Andrew G. "Connectionist Learning for Control: An Overview". COINS Technical Report 89-89 (1989).

[21] Baum, E.B. y Haussler, F.D. "What Size Net Gives Valid Generalization?" Neural Computation 1, 151-160 (1989).

[22] Box, G.E.P., y Jenkins, G.M. . "Time Series Analysis, Forecasting and Control", Holden Day, San Francisco (1970).

[23] Box, G.E.P., y Jenkins, G.M. "Time Series Analysis: Forecasting and Control". Rev. ed., Holden Day, San Francisco (1976).

[24] Box, G.E.P. y Tiao, G.C. . "Intervention analysis with application to economic and environmental problems". J. Amer. Statist. Assoc. 70:70-79 (1975)

[25] Breiman, L. y Meisel, W. S. "General estimates of the intrinsic variability of data in nonlinear regression models". J. Amer. Statist. Assoc. 71, 301-307 (1976)

[1] Burnash, R. J. C.; Ferral, R. L.; McGuire, R. A. "A generalized streamflow simulation system. Conceptual modeling for digital computers", (1979)

[26] Carrascosa, L. I.; Busturia, J. M.; Giménez, J. G. "Estudio comparativo de métodos globales en el análisis modal experimental". Anales de Ingeniería Mecánica . Vol. 2. pag. 148-158 (1984)

[27] Cleveland, W. S. "Robust locally weighted regression and smoothing scatter plots". J. Amer. Statist. Assoc. 74, 828-836 (1979)

[28] Connor, J. ; Atlas, L. y Martin, D. "Recurrent Neural Networks and Time Series Prediction".(1991)

[29] Craven, P. y Wahba, G. "Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation". Numer. Math. 31, 317-403.(1979)

[30] Cybenko, G. "Continuous Valued Neural Networks with Two Hidden Layers Are Sufficient". Technical Report, Department of Computer Science, Tufts University, Medford, MA.(1988)

[31] Doya, Kenji . "Bifurcations in the learning of recurrent neural networks". Proceedings of 1992 IEEE International Symposium on Circuits and Systems. (1992)

[32] Duda, R.O. y Hart, P. E.. "Pattern classification and scene analysis". Wiley, New York. (1973)

[33] Elgerd, O.I. "Control Systems Theory", McGraw-Hill Book Company, New York, (1967)

[34] Elman, J.L. ."Finding Structure in Time". Cognitive Science 14, 179-211. (1990)

[35] Fahlman, S. E. ."The Recurrent Cascade-Correlation Architecture", CMU-CS-91-100, Carnegie Mellon University, (1991)

[36] Fahlman, S.E. y Lebiere, C. ."The Cascade-Correlation Learning Architecture". Advances in Neural Information Processing Systems II (Denver 1989), ed. D.S. Touretzky, 524-532. Morgan Kaufmann, San Mateo. (1990)

[37] Fahlman, S. E. , "An Empirical Study of Learning Speed in Back-Propagation Networks", CMU-CS-88-162, (1988)

[38] Farmer, D. y Sidorovich, J. ."Predicting Chaotic Time Series". Physical Review Letters 59, 845-848. (1987)

[39] Franke, R. "Scattered data interpolation: Test of some methods". Mathematics of Computation, v. 38, nº 157, pp 181-200. (1982)

[40] Frean, M. "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks". Neural Computation 2, 198-209. (1990)

[41] Friedman, J. H. ."Multivariate adaptive regression splines". Ann. Statist. 19, 1-141. (1991)

[42] Gallant, A. T. ,"Nonlinear Statistical Models", John Wiley & Sons, (1987.)

[43] Geman, S. ; Bienenstock, E. y Doursat, R. "Neural Networks and the Bias/Variance Dilemma", Neural Computation, Vol. 4, No. 1, (1992)

[44] Grossberg, S. ."The Adaptive Brain", Amsterdam: Elsevier.(1987)

[45] Härdle, W.; Hall, P. y Marron, J.S. "How far are automatically chosen regression smoothing parameters from their optimum?" J. Am. Statist. Assoc. 83, 86-95.(1988)

[46] Hartman, E.J.; Keeler, J.D. y Kowalski, J.M. ."Layered Neural Networks with Gaussian Hidden Units As Universal Approximations". Neural Computation 2, 210-215. (1990)

[47] Hartman, E. y Keeler, J. D., "Predicting the Future: Advantages of Semilocal Units", Neural Computation 3, 566-578 (1991)

[48] Hertz, J.; Krogh, A. y Palmer, R. G. . "Introduction to the theory of neural computation". Addison-Wesley Publishing Company, Redwood City. (1991)

[49] Hinton, G.E. ."Learning Distributed Representations of Concepts". Proceedings of the Eighth Annual Conference of the Cognitive Science Society (Amherst 1986), 1-12. Hillsdale: Erlbaum.(1986)

[50] Honavar, V. y Uhr, L.. "Generation, local receptive fields and global convergence improve perceptual learning in connectionist networks". IJCAI-89 11th. Intl. Joint Conference on AI. Detroit, pp 180-185. (1989)

[51] Howell, J.; Barnes, C., etc. "Control of a negative-ion accelerator source using neural networks". Nuclear instruments & methods in physics research, A, v293, n1-2, p 517-522 (1990)

[52] Isermann, R. "Practical aspect of process identification", Automatica 16, 575-587. (1980)

[53] Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J. y Hinton, G. E. "Adaptive Mixtures of Local Experts", Neural computation 3, 79-87 (1991)

[54] Jones, R.D.; Lee, Y.C.; Barnes, C.W.; Flake, G.W.; Lee, K.; Lewis, P.S. y Qian, S. "Function Approximation and Time Series Prediction with Neural Networks", Los Alamos National Laboratory Report LA-UR-90-21 (1990)

[55] Jones, R. D. , Lee, Y.C.; Qian, S.; Barnes, C.W.; Bisset, K.R.; Bruce, G.M.; Flake, G.W.; Lee, K.; Lee, L.A.; Mead, W.C.; O'Rourke, M.K.; Poli, I.J. y Thode, L.E.. "Nonlinear Adaptive networks: a little theory, a few applications". Cognitive Modeling in System Control: ... Santa Fe, New Mexico, (1990)

[56] Jordan, M.I. "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine". Proceedings of the Eighth Annual Conference of the Cognitive Science Study (Amherst 1986), 531-546. Erlbaum, Hillsdale. (1986)

[2] Karplus, W.J. Curso "Mathematical Modeling and Digital Computer Simulation of Engineering and Scientific Systems". Zurich (1991)

[57] Kheir, Naim A. (edit) . "Systems Modeling and Computer Simulation". Marcel Dekker, Inc. New York.(1988)

[58] Kohonen, T. "Self-Organized Formation of Topologically Correct Feature Maps". Biological Cybernetics 43, 59-69. (1982)

[59] Kushner, H. "Asymptotic Global Nehavior for Stochastic Approximations and Diffusions With Slowly Decreasing Noise Effects: Global Minimization via Monte Carlo", SIAM Journal on Applied Mathematics, 47, 169-185. (1987)

[60] Kushner, H. y Clark, D. "Stochastic Approximation Methods for Constrained and Unconstrained Systems". Springer-Verlag, Berlin (1978)

[61] Lapedes, A. y Farber, R., "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling", Technical Report, LA-UR87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, (1987)

[62] Lapedes, A. y Farber, R. "How Neural Nets Work". Neural Information Processing Systems (Denver 1987), ed. D.Z. Anderson, 442-456. American Institute of Physics, New York. (1988)

[63] Liu, Lon-Mu "Box-Jenkins Time Series Analysis", BMDP Statistical Software Manual, Vol. 1, pag. 435-488, Univ. of California Press, (1990)

[64] Long, J; Gregoriou, V; Gemperline P. "Spectroscopic calibration and quantitation using artificial neural networks". Analytical chemistry, v62, n17, 1791-1797 (1990)

[65] Marchand, M.; Golea, M. y Ruján, P. "A Convergence Theorem for Sequential Learning in Two-Layer Perceptrons". Europhysics Letters 11, 487-492. (1990)

[66] McClelland, J. L. y Rumelhart, David E. "Explorations in Parallel Distributed Processing", The MIT Press, (1989)

[67] McClelland, J.L. and Elman, J.L. "Interactive Processes in Speech Perceptron: The TRACE Model". Parallel Distributed Processing, vol. 2, chap. 15.(1986)

[68] McCulloch, W.S. y Pitts, W. "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathemathical Biophysics 5, 115-133. Reprinted in Anderson and Rosenberg [1988] . (1943)

[69] Mead, W.C. , Jones, R.D.; Lee, Y.C.; Barnes, C.W.; Flake, G.W.; Lee, L.A. y O'Rourke, M.K.. "Prediction of chaotic time series using CNLS-net--Example: the Mackey-Glass equation". LA-UR-91-720 (1991).

[70] Mèzard, M. y Nadal, J.-P. "Learning in Feedforward Layered Networks: The Tiling Algorithm". Journal of Physics A 22, 2191-2204.(1989)

[71] Minsky, M.L. y Papert, S.A. "Perceptrons". MIT Press, Cambridge.(1969)

[72] Moody, J. y Darken, C. "Learning with Localized Receptive Fields". Proceedings of the 1988 Connectionist Models Summer School (Pittsburg 1988), eds. D. Touretzky, G. Hinton, and T. Sejnowski, 133-143. Morgan Kaufmann, San Mateo. (1988)

[73] Moody, J. y Darken, C. "Fast Learning in Networks of Locally-Tuned Processing Units". Neural Computation 1, 281-294. (1989)

[74] Morgan, J. N. y Sunquist, J. A. "Problems in the analysis of survey data, and a proposal". J. Amer. Statist. Assoc. 58 415-434 (1963)

[75] Movellan, J.R. "Error functions to improve noise resistance and generalization in backpropagation networks". (1989)

[76] Mozer, M.C. "A Focused Back-Propagation Algorithm for Temporal Pattern Recognition". Complex Systems 3, 349-381. (1989).

[77] Nguyen , D. y Widrow, B. "Neural Networks for Self-Learning Control Systems". Proc. IJCNN, Washington, June 1989, IEEE Control Systems Magazine 10 (1990)

[78] Owens, A.J. y Filkin, D.L. "Efficient Training of the Back Propagation Network by Solving a System of Stiff Ordinary Differential Equations". International Joint Conference on Neural Networks (Washington 1989), vol. II, 381-386. IEEE, New York. (1989).

[79] Parzen, E. "On estimation of a probability density function and mode". Ann. Math. Statist. 33, 1065-1076. (1962)

[80] Pearlmutter, B.A. . "Learning State Space Trajectories in Recurrent Neural Networks". Neural Computation 1, 263-269. (1989)

[81] Plaut, D.; Nowlan, S. y Hinton, G. . "Experiments on Learning by Back Propagation". Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA. (1986)

[82] Poggio, T. y Girosi, F. . "Regularization Algorithms That Are Equivalent to Multilayer Networks". Science 247, 978-982. (1990)

[83] Pugachev, V. S. y Sinitsyn, I. N. "Stochastic Differential Systems", John Wiley & Sons, (1987)

[84] Raibert, M.H.. "A model for sensorimotor control and learning", Biological Cybernetics, 29:29-36 (1978)

[3] Revilla Cortezón, J. A.; Liaño Herrera, A. y Sainz Borda, A. "Apuntes de Hidrología Superficial Aplicada", ETSICCP Universidad de Cantabria (1982)

[85] Rice, J. , "Bandwith Choice for Nonparametric Regression", The Annals of Statistics, 12, 1215-1230 (1984)

[86] Rissanen, Jorma . "Stochastic Complexity in Statiscal Inquiry". World Scientific, (1989)

[87] Schwartz, D.B.; Samalam, V.K.; Solla, S.A.; y Denker, J.S. . "Exhaustive Learning". Neural Computation 2, 371-382. (1990)

[4] Shaw, E. M. "Hidrology in practice". Chapman and Hall. (1990)

[88] Shimohara, K., Uchiyama, T. y Tokunaga, Y. . "Back-Propagation Networks for Event-Driven Temporal Sequence Processing". IEEE International Conference on Neural Networks (San Diego 1988), vol. I, 665-672. IEEE, New York. (1988)

[89] Sietsma, J. y Dow, R.J.F. . "Neural Net Pruning-Why and How". IEEE International Conference on Neural Networks (San Diego 1988), vol. I, 325-333, IEEE, New York (1988)

[90] Stokbro, K.; Umberger, D.K. y Hertz, J.A. . "Exploiting Neurons with Localized Receptive Fields to Learn Chaos". Preprint 90/28 S, Nordita, Copenhagen, Denmark.(1990)

[91] Takens, F. ."Detecting Strange Attractors In Turbulence". Dynamical Systems and Turbulence. Lecture Notes in Mathematics, vol. 898 (Warwick 1980), eds. D.A. Rand and L.-S. Young, 366-381. Springer-Verlag, Berlin.(1981)

[5] Waibel, A. . "Modular Construction of Time-Delay Neural Networks for Speech Recognition". Neural Computation 1, 39-46. (1989)

[92] Weigend, A.S.; Rumelhart, D.E. y Huberman, B.A. . "Generalization by Weight-Elimination with Application to Forecasting". Advances in Neural Information Processing 3. ed: R.P. Lippmann, J. Moody and D.S. Touretzky. Morgan Kaufmann, San Mateo, CA.(1991)

[93] Werbos, P. . "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences". Ph.D. Thesis, Harvard University. (1974)

[94] White, H. "Some Asymptotic results for learning in single hidden-layer feedforward network models". JASA , v84, n408, p1003-1013 (1989)

[95] White, H. , "Consequences and Detection of Misspecified Nonlinear Regression Models", Journal of the American Statistical Association, 76, 419-433 (1981)

[96] Widrow, B. y Stearns, S. D. . "Adaptive Signal Processing". Prentice-Hall, Inc., Englewood Cliffs, N. J., (1985)

[97] Widrow, B. , "Adaptive Inverse Control", Adaptive Systems in Control and Signal Processing 1986, International Federation of Automatic Control, (1986)

[98] Williams, R.J. y Zipser, D. . "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". Neural Computation 1, 270-280. (1989)

[99] Wolpert, D. H. "A mathematical theory of generalization", Complex Systems 4, 151-249 (1990)

[100] Ydstie, B. E. , "Forecasting and Control using Adaptive Connectionist Networks", Computers chem. Engng. Vol 14, No. 4/5, pp. 583-599 (1990)

[101] Zeigler, B. P. . "Theory of Modeling and Simulation", Wiley, New York. (1976)

4. Descripción del Algoritmo

En este capítulo se va a describir el algoritmo utilizado desde un punto de vista informático, para aproximar los aspectos matemáticos y formales del capítulo anterior a la programación práctica que los desarrolle.

4.1. Funciones a realizar

En este apartado se presentan los distintos pasos que deben darse en la implementación práctica del algoritmo.

Las tareas a realizar por el proceso son:

- Creación de una red mínima inicial.
- Creación de nuevas redes por ampliación de otras.
- Ajuste del agrupamiento en una red (centros y anchos).
- Ajuste de los procesadores lineales a los datos muestrales.
- Utilización de redes ya definidas sobre muestras para simulación.

De todas estas la última es separable, por lo que puede realizarse aparte de las demás, constituyendo un desarrollo independiente.

4.2. Interfase

En este apartado se describe la interacción entre el usuario y el proceso, es decir, los datos o información que aquel debe proporcionar a este.

El usuario puede elegir entre ajustar un modelo o usar uno preajustado. Como alternativa puede usarse un programa diferente para posprocesar la red (ver apartado de continuidad en el capítulo anterior).

En el caso de generar un modelo deben proporcionarse las secuencias que van a utilizarse para los ajustes parciales y para la validación y ajuste definitivo. Asimismo se podrá indicar cuál es la precisión requerida, para ahorrar complejidades innecesarias. Una vez ajustado, el modelo será archivado, para lo cual el usuario dará un nombre que permita identificarlo posteriormente.

En caso de simulación con un modelo ya ajustado, este deberá ser proporcionado por el usuario citando su nombre, que también debe preparar las secuencias cuyas entradas se van a suministrar. Por último proporcionará un identificador para archivar las secuencias de resultados.

4.3. Formato de los datos

El conjunto básico de datos a proporcionar son las secuencias de ajuste, validación y uso.

A efectos de reducir los problemas numéricos es conveniente, con anterioridad, escalar las variables a orden de magnitud unidad. Esta transformación afín preserva en su caso la linealidad.

Puesto que se puede tratar de conjuntos de datos bastante grandes, no es lógico proporcionarlos interactivamente, sino mediante ficheros confeccionados al efecto. El formato de uno de estos ficheros se describe a continuación.

Una primera línea con el número de variables independientes (M) y el número de variables dependientes (N) separados por un espacio en blanco:

$$M\ N$$

A continuación cada secuencia de valores compuesta de tantas líneas como instantes de tiempo comprenda. Cada línea incluirá el valor de todas las variables para cada instante en formato libre (separados mediante espacios):

$$x_1 \ldots x_i \ldots x_M\ y_1 \ldots y_j \ldots y_N$$

donde x_i son las variables independientes y y_j las dependientes.

La conclusión de la secuencia se identificará con una línea en la que hay una palabra cualquiera; por ejemplo:

fin

A continuación puede comenzar la siguiente secuencia con el mismo formato utilizado para la anterior, y así sucesivamente. El fichero se concluirá con el fin de la última secuencia.

4.4. Estructura de la información

Como ya se ha definido los datos de entrada son principalmente unas secuencias de entrada y salida del sistema en cuestión.

Los datos de salida son, en el caso de ajuste, la estructura y parámetros de una red que funciona como modelo del sistema. Esto comprenderá la topología y los pesos (datos de conexiones).

La información que maneja el algoritmo admite la siguiente clasificación:

- Valor numérico de cada variable del sistema en un instante dado.

- Centro en cada conexión de agrupamiento

- Ancho en cada conexión de agrupamiento

- Peso en cada conexión de procesador lineal

- Identificación de los procesadores enlazados por cada conexión

- Valor de salida de cada procesador

- Error asociado a cada procesador de clasificación

Además habrá variables auxiliares (dimensiones particulares de un problema, etc.).

La cantidad necesaria de cada uno de estos tipos de datos es:

- Valor de una variable en un instante dado (n_V) Para cada variable tantas como instantes haya entre todas las secuencias.

$$n_v = n_{\text{var}} \sum_{\text{sec.}} n_{inst}$$

- Centro en una conexión de agrupamiento (n_M). Tantos como haya de estas conexiones, lo que equivale al número de procesadores de clasificación multiplicado por el número de variables explicativas o de entrada (valores actuales o retrospectivos de alguna variable). Esto se sumará para cada red.

$$n_M = \sum_{red} n_{exp} n_{clas}$$

- Ancho en conexiones de agrupamiento (n_σ). Los mismos que los anteriores.

$$n_\sigma = \sum_{red} n_{exp} n_{clas}$$

- Pesos en conexiones de ajuste lineal (n_l). Tantos como haya de estas; hay que tener en cuenta que entra una constante además de las variables.

$$n_l = \sum_{red} (n_{exp} + 1) n_{clas}$$

Es posible que tras el ajuste algunos de estos pesos sean nulos, por lo que este número quedaría como una cota superior.

- Identificación de procesador enlazado en una conexión (n_{punt}). Tantas como conexiones haya. El número de conexiones necesario para agrupamiento ya se ha definido. Las necesarias para las combinaciones lineales también ha sido calculado anteriormente. El número de procesadores lineales es igual al de los que evalúan clasificación. Además habrá que conectar todos estos a sus salidas. Lógicamente cada procesador, sea de clasificación o de ajuste lineal, se corresponderá a una salida.

$$n_{punt} = \sum_{red} \left(n_{exp} n_{clas} + (n_{exp} + 1) n_{clas} + 2 n_{clas} \right)$$

Debido a la posible eliminación de conexiones lineales este número será también una cota superior.

- Valor de salida de cada procesador (n_p). Este número se obtiene sumando para cada red el número de procesadores de clasificación, ajuste y salida. Además hay que tener en cuenta a los que hacen las veces de entrada, tomando los valores de las correspondientes variables explicativas (o la constante).

$$n_p = \sum_{red} (n_{exp} + 1 + n_{clas} + n_{clas} + n_{sal})$$

- Error asociado a un procesador de clasificación (n_{er}). Tantos como procesadores de clasificación en el total de las redes consideradas.

$$n_{er} = \sum_{red} n_{clas}$$

Estos datos elementales se agruparán en conjuntos, a saber:

- Los valores de las variables en instantes, que comprenden los de todas las variables en un momento dado, y en secuencias cuando se considere una de estas completa.

- Los que caractericen conexiones (centros, anchos, pesos) se agruparán con los de otras conexiones que vayan al mismo procesador.

- El valor de salida o el error está asociado a cada procesador. Estos se agruparán en capas que efectúen la misma función (entrada, clasificación, combinación lineal y salida) y estas a su vez formarán cada red.

Esta clasificación de datos puede verse en la fig. 4-1.

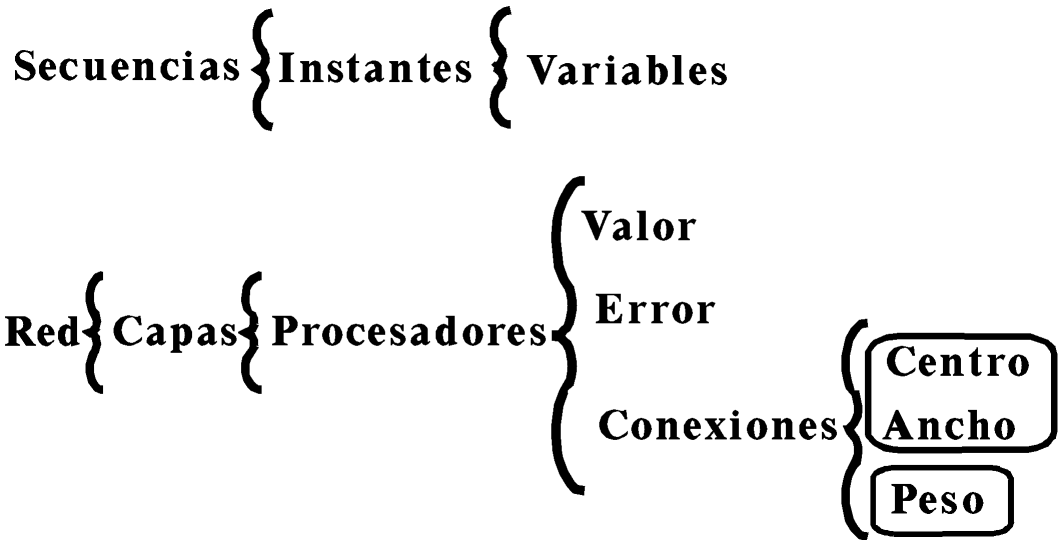

Fig. 4-1. Estructura de los datos.

4.5. Flujo de datos

De las dos clases de datos, los de secuencia se van a usar en el ajuste de los de red, introduciéndose las entradas en los procesadores destinados al efecto y comparándose las salidas con las proporcionadas por la red en el caso de ajuste.

Los datos asociados a conexiones se usarán para obtener las entradas al procesador correspondiente, a partir de las cuales se obtendrá su valor de salida, y de este, en su caso el error asociado. Durante el ajuste, se utilizarán

los errores, en el caso de ser supervisado para ir modificando los parámetros de las conexiones. También juegan un papel los errores en las ampliaciones de la red. En el caso de ajuste no supervisado los parámetros de las conexiones variarán de acuerdo con las entradas solamente.

Así, en la fase de ajuste, se usarán todos los datos, y se irán modificando los correspondientes a las redes.

En la fase de simulación, no se utilizarán datos de variables de respuesta del sistema, y sólo se irán modificando los valores de salida de los procesadores para obtener la predicción de la red.

La operación normal de la red empieza con la asignación de los valores correspondientes a los procesadores con funciones de entradas y termina al calcularse la salida de los procesadores cuya respuesta representa la predicción de la red.

El ajuste supervisado toma estos resultados y modifica en consecuencia los parámetros correspondientes. El no supervisado modifica los parámetros de clasificación (centros y anchos) independientemente de la salida final de la red.

Las distintas tareas a realizar por el proceso pueden pues agruparse en las siguientes fases:

- generación de redes

- ajuste de parámetros

- comparación de las distintas redes

La obtención de resultados de la red es una fase comprendida en la de ajuste, como es habitual en los algoritmos PPD.

El algoritmo de ampliación de la red dispone de las tres vías de expansión comentadas en el capítulo anterior, que va desarrollando sucesivamente.

El algoritmo de crítica y comparación establece qué ampliación se toma en cuenta en cada paso. También decide cuando se considera alcanzado el modelo final.

El algoritmo de ajuste permite, para una determinada topología de red, alcanzar un mínimo de error.

La obtención de resultados permite calcular una salida en tiempo *n+1* respecto del punto en que está la red. La obtención de una secuencia de salida será por iteración de este proceso.

4.6. Secuencia de trabajo

A continuación se describe, a modo de seudocódigo, la secuencia de operaciones que debe realizar el algoritmo. Posteriormente se detallan las partes señaladas con (♦).

En modo de ajuste el algoritmo sigue el esquema del cuadro 4-1.

```
Lee secuencias
Lee precisión requerida
Crea red (♦)
Ajusta red (♦)
Repite:
    Amplia red (♦)
    Ajusta ampliaciones
    Elige ampliación
hasta que la generalización empeore o se alcance la precisión requerida
Ajuste supervisando agrupamiento (de la mejor red con la muestra completa)
    (♦)
Archiva red
→FIN
```

Cuadro 4-1. Secuencia global del algoritmo

La operación Crea red tiene la estructura indicada en el cuadro 4-2.

La operación Ajusta red responde al esquema indicado en el cuadro 4-3.

```
Crea procesador de salida constante
Crea procesadores asociados a las variables independientes
Crea un procesador de clasificación por cada variable dependiente
Conecta estos procesadores a los asociados a variables independientes
Crea un procesador lineal por cada variable dependiente
Conecta estos a los de variable independiente y al constante
Crea un procesador de salida por cada variable dependiente
Conecta estos procesadores a los correspondientes lineales y de clasificación
→RETORNA
```

Cuadro 4-2. Secuencia de la fase de creación de la red elemental.

```
Repite:
    Obtiene la predicción para las secuencias
    Modifica los centros en las conexiones de clasificación
hasta que se llega a una situación estacionaria
Calcula las varianzas de cada elemento de clasificación para definir los
    anchos
Repite:
    Obtiene la predicción para las secuencias
    Modifica los pesos de los procesadores lineales
hasta que se llega a límite de iteraciones o situación estacionaria
```

Cuadro 4-3. Secuencia del proceso de ajuste de una red.

El proceso Amplia red puede ser de 2 tipos: aumento del número de procesadores de clasificación o aumento del orden en los filtros de las entradas. El aumento de clasificación se describe en el cuadro 4-4 y el aumento de orden en el 4-5.

```
Añade un procesador de clasificación por cada salida
Inicializa sus conexiones en función del procesador de mayor error
Añade un procesador lineal por cada salida
Inicializa su conexiones a los mismos valores que el procesador de referencia
Conecta cada procesador de salida a los que le correspondan de los nuevos
    creados
```

Cuadro 4-4. Secuencia de la ampliación del número de procesadores ocultos.

```
Crea tantos procesadores como variables
Conecta estos procesadores a los de clasificación
Conecta los nuevos procesadores a los lineales
→RETORNA
```

Cuadro 4-5. Secuencia de la ampliación de orden.

El Ajuste supervisando agrupamiento tiene la forma indicada en el cuadro 4-6.

```
Repite:
    Obtiene la predicción para las secuencias
    Modifica los centros en las conexiones de clasificación
hasta que se llega a una situación estacionaria
Calcula las varianzas de cada elemento de clasificación para definir los
    anchos
Repite:
    Obtiene la predicción para las secuencias
    Modifica los pesos de los procesadores lineales
hasta que se llega a límite de iteraciones o situación estacionaria
Repite:
    Adapta la clasificación
    Repite:
            Obtiene la predicción para las secuencias
            Modifica los pesos de los procesadores lineales
    hasta que se llega a límite de iteraciones o situación estacionaria
hasta que el error residual no mejore
```

Cuadro 4-6. Secuencia del ajuste supervisando agrupamiento.

4.7. Grado de propiedades PPD del algoritmo

Este trabajo ha sido realizado enteramente mediante proceso en serie convencional. Cara a su posible implementación mediante procesado paralelo y distribuido real interesa saber hasta que punto el esquema utilizado responde a las características de este tipo de algoritmos. Una de ellas es el ajuste mediante ejemplos, que por ser un aspecto obvio del método no se comenta. Las demás se repasan a continuación, describiendo el grado de cumplimiento alcanzado.

4.7.1. Paralelismo

—Lee secuencias: por tratarse de una lectura es necesariamente procesado en serie.

—Crea red: añadir un elemento en una capa es independiente de los demás que se añadan; la creación de una capa está limitada por otras sólo en cuanto a los enlaces, pero estos se almacenan mediante identificadores, por lo que no resulta necesaria la existencia del elemento enlazado; el proceso es por tanto completamente paralelo.

—Ampliación del nº de procesadores de clasificación:

Añade un procesador de clasificación por cada salida: cada uno puede ir en paralelo.

Añade un procesador lineal por cada salida: ídem y en paralelo con los de clasificación.

Conecta cada procesador de salida a los que le correspondan de los nuevos procesadores creados: tampoco requiere las fases anteriores, por lo que puede asimismo ir en paralelo.

—Aumento de orden:

Crea tantos procesadores como variables: cada uno en paralelo con los demás

Conecta los nuevos procesadores a la red: puede ser paralelo a lo anterior.

—Operaciones en la fase de ajuste de la red:

Obtención de la predicción para las secuencias introducidas: primero deben cargarse las entradas de la red; en el programa presentado se requiere procesamiento en serie para los que representan predicciones anteriores de la red, porque van tomando cada uno el valor del anterior; la fase de clasificación permite el procesamiento en paralelo de cada elemento; también pueden funcionar en paralelo los procesadores lineales; además estos y los de clasificación pueden operar independientemente; cuando estos han concluido se evalúa la predicción de la red, que requiere una comparación de mínimo entre los procesadores de clasificación, para tomar la salida del lineal correspondiente; para el ajuste no supervisado sólo es necesario en realidad llegar hasta la fase de clasificación.

Modificación de los centros en las conexiones de clasificación: una vez elegido el mínimo, lo que se ha hecho en la fase anterior, es sólo ese el que se ajusta; si se interpreta como un ajuste multiplicado por 1 ó 0, todos los procesadores pueden ir en paralelo.

Cálculo de las varianzas de cada elemento de clasificación para definir los anchos: esta fase presenta las mismas características que las dos anteriores.

Modificación de los pesos de los procesadores lineales: El ajuste estocástico es independiente en cada uno; el gradiente aproximado también, a través de las conexiones.

—Ajuste supervisando agrupamiento, de la mejor red con la muestra completa. Presenta como novedad las siguientes partes:

Adaptar la clasificación: la adaptación de los centros se hace comparando los centros y anchos de cada uno con los de los demás, para buscar el más próximo y reamoldar las posiciones en función del error acumulado de ambos; así pues, hay tantas iteraciones como procesadores, y en cada una se ejecuta primero uno y luego todos los demás en paralelo.

Archiva red: Por tratarse de un proceso de almacenamiento es serie.

4.7.2. Cálculos locales

—Crea red: no se necesita ningún dato de otros procesadores.

—Ampliación del nº de procesadores de clasificación:

Añade un procesador de clasificación por cada salida: puesto que cada nuevo procesador se crea cercano al de mayor error, resulta necesario ciclar por los anteriores para saber cuál es; para la nueva posición del centro se usan los valores de aquel.

Añade un procesador lineal por cada salida: los pesos se toman iguales a los del elegido anteriormente.

Conecta cada procesador de salida a los que le correspondan de los nuevos: no requiere ningún dato de ningún procesador.

—Aumento de orden:

Crea tantos procesadores como variables: la creación de un procesador no precisa ningún dato.

Conecta los nuevos procesadores al resto de la red: tampoco necesita datos.

—Ajuste de la red:

Obtención de la predicción para las secuencias: los datos que necesita cada procesador son los que toma en sus conexiones; la fase de clasificación requiere una comparación de mínimo entre los procesadores de clasificación.

Modificación de los parámetros de posición de las conexiones de clasificación: una vez elegido el mínimo, lo que se ha hecho en la fase anterior, es sólo ese el que se ajusta en función de la entrada que ha tenido, sin necesidad de otros datos.

Calcular las varianzas de cada elemento de clasificación para definir los anchos: esta fase presenta las mismas características que las dos anteriores.

Modificación de los pesos de los procesadores lineales: el ajuste estocástico es independiente en cada procesador, existiendo una variable global que es el ancho de banda en cada momento; para el ajuste por gradiente incompleto todos los datos son accesibles por las conexiones.

—El ajuste supervisando agrupamiento de la mejor red con la muestra completa incluye además de fases ya comentadas la adaptación de la clasificación: se pueden usar para almacenamiento los procesadores que normalmente tienen sentido de entrada, por lo que las únicas variables globales son auxiliares, como por ejemplo el número de procesadores en cada capa.

4.7.3. Representación distribuida

El resultado del ajuste son una serie de representaciones lineales localizadas en un conjunto de subdominios. Los parámetros que rigen ambos aspectos están asociados a conexiones, por lo que se puede afirmar que los resultados están representados en la topología de la red con esos parámetros.

4.7.4. Capacidad de extrapolación

La capacidad de extrapolación del modelo resultante se pretende garantizar siguiendo el esquema de división de muestra para controlar el límite en orden y número de subdominios alcanzado.

4.8. Ejemplo

En este apartado se presenta la secuencia de ajuste sobre un ejemplo, a partir de unas secuencias ficticias creadas al efecto. El proceso se desarrolla de la siguiente forma:

Lee las secuencias. Se separa aproximadamente la mitad de los datos para el ajuste. Con el total se estimará la generalización, hasta llegar al modelo final, donde serán utilizados todos los datos para el último ajuste.

Lee la precisión requerida. Como los datos se han escalado entre 0 y 1 y se estima que una precisión del 1% obligará al algoritmo a apurar el ajuste, se le da el valor 0.01.

Crea la red inicial.

Ajusta esta red. El error cuadrático medio residual sobre la muestra de ajuste resulta 0.25 y sobre el total 0.34.

Repite:

Amplia la red

Ajusta ampliaciones

hasta que la generalización empeore.

La evolución puede contemplarse en la tabla 4-1, donde en cada fase, la red seleccionada es la marcada con un doble recuadro.

Ajuste supervisando agrupamiento de la mejor red con la muestra completa. Error residual final: 0.038

Archiva red. Se da un nombre para el fichero en que va a estar contenida.

En el ejemplo propuesto se ve que no hay más que un dominio y por tanto una sola función de transferencia lineal, de primer orden en la propia variable dependiente y tercero en la independiente.

Tabla 4-1. Evolución de los errores residual y de generalización (primera y segunda fila respectivamente) en el ejemplo. Sólo se incluye el error de generalización de la red escogida en cada fase.

Fase del proceso	Ampliación de elementos de clasificación	Ampliación de orden en variables independientes	Ampliación de orden en variables dependientes
1ª	0.25	0.19	0.18 0.25
2ª	0.14	0.075 0.073	0.17
3ª	0.057	0.038 0.049	0.057
4ª	0.030	0.026 0.038	0.027

4.9. Formato de descripción de la red

El formato con el que se describe la red en el fichero que da como salida el programa es el siguiente:

Numero de entradas (variables independientes)

Numero de salidas (variables dependientes)

Precisión (error residual del último ajuste)

Capa i (donde cada capa es un grupo de procesadores identificado con los valores indicados en la tabla 4-2)

numero de elementos (de la capa)

Elemento j (cada uno de los procesadores numerados a partir de 0)

valor (último valor)

sigma (entrada al procesador)

c1 (error asociado al procesador, si es de clasificación)

c2 (número de puntos asociados al procesador, si es de clasificación)

numero_entradas (conexiones que entran a este procesador)

Entrada k

valor (último calculado)

peso (en el caso de procesadores de clasificación, centro cambiado de signo)

variacion (ancho para procesadores de clasificación)

ajuste (variable auxiliar)

conexion (0 indica aditiva; 1 indica multiplicativa)

orden (número de entradas a esta conexión)

Conexion m (cada entrada)

capa (del procesador de entrada)

numero (ídem)

Tabla 4-2. Identificadores de las capas.

Identificador	Tipo
0	Constante
1	Variables independientes (valores actuales y retrospectivos)
2	Variables dependientes (valores retrospectivos)
3	No se usa
4	Clasificación
5	Lineal
6	Salida

Cara a la interpretación de la red, la información más importante está en las conexiones. Cada procesador de clasificación define un subdominio cuyo centro está indicado en la conexión a cada variable de entrada. En las conexiones de la salida se asocia a cada uno de estos procesadores de clasificación uno lineal, que define una función de transferencia con los coeficientes definidos por sus pesos aplicados a las respectivas variables.

4.10. Criterios y limitaciones

En este apartado se comentan aquellos parámetros que se han fijado en la programación, así como los límites de dimensiones de los datos.

El número de entradas y salidas puede ser cualquiera. La longitud total de las secuencias está limitada a 100000 valores de cada variable, distribuidos en un máximo de 100 secuencias. Estas limitaciones son redefinibles en el programa, por lo que es inmediata su modificación.

Se admite que cada secuencia empieza en un punto estacionario, esto es, que cualquier valor previo de las variables es igual al inicial. Esta hipótesis puede ser necesaria en el cálculo de la predicción de los primeros puntos de la secuencia.

En el ajuste de los parámetros de las conexiones de una red se fijan dos criterios de terminación. El primero es un número máximo de iteraciones, que se fija en 10000. El segundo es dar los parámetros por estabilizados; se considera cumplido este criterio cuando la diferencia entre iteraciones consecutivas no excede en valor absoluto de 10^{-6}.

No existe limitación en cuanto al tamaño de las redes.

Tras el último ajuste se eliminan aquellas conexiones cuyos pesos sean inferiores a 0.001.

4.11. Características informáticas del programa

La programación se ha orientado a garantizar el funcionamiento correcto del algoritmo, pero no ha sido optimizada en el sentido de intentar una elevada eficacia computacional, por lo que las características que a continuación se mencionan son puramente indicativas.

Se ha utilizado el lenguaje C.

El tamaño total del programa (en torno a las 50 funciones repartidas en varios módulos) en código fuente está alrededor de los 60 Kb.

El tamaño del programa ejecutable depende de las opciones de compilación, no siendo superior a los 100Kb.

Para el almacenamiento de muestras y redes en memoria se utiliza el direccionamiento dinámico con punteros, basado en la función ***malloc***.

En cuanto a velocidad de respuesta, ejecutado en una estación de trabajo a 33 MHz, el programa, ajustando un modelo, sólo puede considerarse interactivo cuando las muestras son pequeñas. En fase de simulación siempre puede utilizarse interactivamente.