RED NEURONAL ESTRUCTURADA EN EL ESPACIO DE ESTADOS COMO MODELO DE CAJA GRIS

Jesús M. Zamarreño
Dpto. de Ingeniería de Sistemas y Automática, Universidad de Valladolid, jesusm@autom.uva.es
Institute of Sustainable Processes, University of Valladolid

Alejandro Merino
Departamento de Ingeniería Electromecánica, Universidad de Burgos, alejandromg@ubu.es

Resumen

La Red Neuronal en el Espacio de Estados (RNEE) ha demostrado muy buenas propiedades en el modelado de sistemas dinámicos. En este artículo, proponemos una evolución de dicha red neuronal cuando la información sobre la estructura interna del sistema está disponible mediante algún tipo de modelo. Con esta información se puede obtener un modelo de caja gris que representa de forma más fidedigna el sistema modelado. Este modelo ha sido bautizado como Red Neuronal Estructurada en el Espacio de Estados (RNEEE). Se presenta un ejemplo sobre un caso de estudio en simulación.

Palabras clave: Modelo en espacio de estados, Red neuronal, Modelo de caja gris.

1. INTRODUCCIÓN

Habitualmente, en identificación de sistemas con redes neuronales, se suelen utilizar dos tipos: redes perceptron multicapa con información de entradas/salidas pasadas, y redes recurrentes. En muchos casos, las redes perceptron multicapa no son capaces de capturar la información completa del estado del proceso [9]. Además, debido a la necesidad de incluir un número indeterminado de entradas y salidas pasadas como entrada a la red, en este tipo de redes el número de pesos crece rápidamente. El comportamiento de los sistemas dinámicos puede ser representado por redes recurrentes de una forma más natural [3], [4], [6].

En concreto, la red neuronal dinámica en la que se basa este trabajo, RNEE (Red Neuronal en el Espacio de Estados), fue presentada originalmente en [9], donde algunas de sus propiedades más importantes fueron analizadas como la estabilidad asintótica de puntos de equilibrio y el estudio de la respuesta temporal.

La RNEE es una gran herramienta para modelado de procesos no lineales como ha sido mostrado en varios casos [2, 8, 10, 11] donde sus ventajas en el modelado de procesos dinámicos no lineales fueron demostradas a través de sus buenos resultados. Las principales ventajas de esta aproximación al modelado son su capacidad para representar cualquier dinámica no lineal, y lo que podría denominarse como *modelado paralelo*: el modelo representa la relación causa-efecto de la dinámica del proceso sin considerar entradas pasadas y/o salidas pasadas. La relación dinámica se modela a través de la capa de estados, que calcula el estado interno de la red usando solamente entradas al modelo y valores de estado internos del periodo de muestreo anterior. Debido a la existencia de conexiones recurrentes, se desarrolló un algoritmo de entrenamiento híbrido, presentado en [1], donde la parte estocástica basada en [7] fue mejorada para acelerar su convergencia.

La arquitectura de la RNEE (ver figura 1) consiste en cinco bloques, y cada bloque representa una capa neuronal. De izquierda a derecha, el número de neuronas (o elementos de proceso) en cada capa es n, h, s, $h2$ y m. La tercera capa representa el estado del sistema (la dinámica). Como puede verse en la figura, existe una realimentación de la capa de estado a la capa precedente, lo que significa que el estado actual depende (de una forma no lineal) del estado en el periodo de muestreo anterior. Las capas segunda y cuarta modelan el comportamiento no lineal: de las entradas a los estados y de los estados (y posiblemente entradas) a las salidas, respectivamente. Las capas primera y quinta aportan transformaciones lineales a las entradas y salidas, respectivamente.

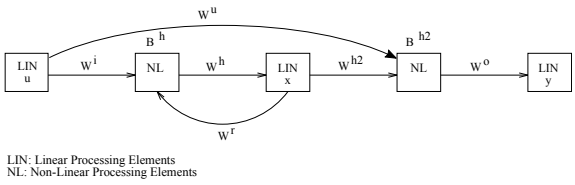

Figura 1: Arquitectura genérica de red neuronal en el espacio de estados

La RNEE se implementa a través de la siguiente representación matemática, suponiendo un mode-

lo discreto con periodo de muestreo unidad:

$$\hat{\vec{x}}(t+1) = W^h \cdot f_1(W^r \cdot \hat{\vec{x}}(t) + W^i \cdot \vec{u}(t) + B^h) \quad (1)$$
$$\hat{\vec{y}}(t) = W^o \cdot f_2(W^{h2} \cdot \hat{\vec{x}}(t) + W^u \cdot \vec{u}(t) + B^{h2}) \quad (2)$$

donde los parámetros son las matrices de pesos, W, y los vectores de bias, B:

- W^i, W^h, W^r, W^{h2}, W^u, W^o son matrices de dimensión $h \times n$, $s \times h$, $h \times s$, $h2 \times s$, $h2 \times n$ y $m \times h2$, respectivamente.
- B^h and B^{h2} son vectores de bias con h y $h2$ elementos respectivamente.
- f_1 y f_2 son dos funciones (no lineales, en general) que se aplican elemento a elemento a un vector o una matriz. Usualmente son de tipo sigmoide.

Las ecuaciones 1 y 2 representan un modelo en espacio de estados discreto no lineal donde las funciones no lineales f_1 y f_2 modelan las no linealidades de los estados y las salidas, respectivamente.

Este modelo de red neuronal puede representar cualquier sistema dinámico no lineal descrito por

$$\vec{x}(t+1) = \mathbf{f}(\vec{x}(t), \vec{u}(t)) \quad (3)$$
$$\vec{y}(t) = \mathbf{g}(\vec{x}(t), \vec{u}(t)) \quad (4)$$

donde $\vec{x} \in \Re^s$, $\vec{u} \in \Re^n$, $\vec{y} \in \Re^m$ son los vectores de estado, entrada y salida, respectivamente, y $\mathbf{f}$ y $\mathbf{g}$ son funciones no lineales.

Sin embargo, en los trabajos presentados hasta ahora con esta arquitectura no se utilizó información estructural sobre los procesos para el diseño de la RNEE más allá del número de entradas, salidas, y, eventualmente, número de estados. En este sentido, la red neuronal se ha considerado como un modelo de caja negra, donde las relaciones específicas entre los estados y las entradas y entre las salidas y los estados se deducen indirectamente a través del proceso de entrenamiento. En este trabajo, la información estructural del modelo ha sido incluida en la estructura de la RNEE, usando estas relaciones conocidas para obtener un modelo de caja gris más adecuado para representar el proceso.

El resto del artículo está estructurado de la siguiente manera: en la sección 2 se desarrolla el soporte matemático que describe la RNEEE (Red Neuronal Estructurada en el Espacio de Estados) y en la sección 3 se muestra un caso de estudio sencillo donde se aplica el modelado con la RNEEE. Terminamos el artículo con unas conclusiones y trabajo futuro.

2. RED NEURONAL ESTRUCTURADA EN EL ESPACIO DE ESTADOS

Si tomamos como base las ecuaciones 3 y 4, las relaciones no lineales pueden ser expresadas para cada estado y cada salida como

$$x_i(t+1) = f_i(\vec{x}(t), \vec{u}(t)) \quad (5)$$
$$y_j(t) = g_j(\vec{x}(t), \vec{u}(t)) \quad (6)$$

donde $i = 1, \ldots, s$, $j = 1, \ldots, m$.

Con estas dos últimas ecuaciones (5 and 6), cada bloque no lineal de la figura 1 puede ser dividido en s y m subbloques, respectivamente (ver figura 2).

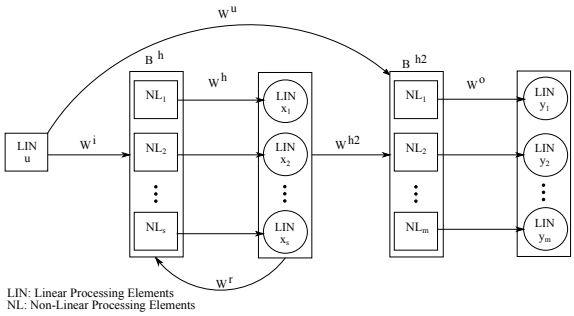

Figura 2: Arquitectura de red neuronal estructurada en el espacio de estados

El número de neuronas en cada bloque para la primera capa no lineal es $h_1, h_2, \ldots, h_s$, siendo $h = h_1 + h_2 + \ldots + h_s$. El número de neuronas en cada bloque para la segunda capa no lineal es $h2_1, h2_2, \ldots, h2_m$, siendo $h2 = h2_1 + h2_2 + \ldots + h2_m$. De esta forma, la matriz W^h puede escribirse como una matriz particionada por bloques con s particiones columna:

$$W^h = \begin{pmatrix} W^h_{11} & W^h_{12} & \cdots & W^h_{1s} \\ W^h_{21} & W^h_{22} & \cdots & W^h_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^h_{s1} & W^h_{s2} & \cdots & W^h_{ss} \end{pmatrix} \circ \begin{pmatrix} J_{1,h_1} & O_{1,h_2} & \cdots & O_{1,h_s} \\ O_{1,h_1} & J_{1,h_2} & \cdots & O_{1,h_s} \\ \vdots & \vdots & \ddots & \vdots \\ O_{1,h_1} & O_{1,h_2} & \cdots & J_{1,h_s} \end{pmatrix} \quad (7)$$

donde W^h_{ij} es una submatriz con una fila y h_j columnas, con $i, j = 1, \ldots, s$. J_{1,h_j} es una matriz de todo unos de dimensión *1* x h_j y O_{1,h_j} es una matriz nula de dimensión *1* x h_j. El operador $\circ$ es

el producto Hadamard (también conocido como producto Schur o producto elemento a elemento).

De la misma forma, la matriz W^o puede ser estructurada como una matriz particionada por bloques con m particiones columna:

$$W^o = \begin{pmatrix} W^o_{11} & W^o_{12} & \cdots & W^o_{1m} \\ W^o_{21} & W^o_{22} & \cdots & W^o_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ W^o_{m1} & W^o_{m2} & \cdots & W^o_{mm} \end{pmatrix} \circ \begin{pmatrix} J_{1,h2_1} & O_{1,h2_2} & \cdots & O_{1,h2_m} \\ O_{1,h2_1} & J_{1,h2_2} & \cdots & O_{1,h2_m} \\ \vdots & \vdots & \ddots & \vdots \\ O_{1,h2_1} & O_{1,h2_2} & \cdots & J_{1,h2_m} \end{pmatrix} \quad (8)$$

donde W^o_{ij} es una submatriz con una fila y $h2_j$ columnas, con $i, j = 1, \ldots, m$. $J_{1,h2_j}$ es una matriz de todo unos de dimensión *1* x $h2_j$ y $O_{1,h2_j}$ es una matriz nula de dimensión *1* x $h2_j$.

Si es posible disponer de información *a priori* de las ecuaciones de primeros principios o cualquier otro tipo de relación que permita establecer dependencias entre variables, un modelo de *caja negra* podría ser demasiado genérico para representar al proceso. Este conocimiento puede ser usado para restringir aun más la arquitectura del modelo, de tal forma que resultaría en un modelo de *caja gris* que puede representar al proceso de forma más fidedigna.

Consideremos que tenemos información sobre las dependencias individuales de cada componente del vector $\vec{x}$ e $\vec{y}$ con respecto a las componentes de los vectores de estados anteriores y entradas. Estas dependencias podrían ser explícitamente incorporadas en el modelo a través de matrices (0, 1): M^r, M^i, M^{h2}, M^u con dimensiones s x s, s x n, m x s, m x n, respectivamente. Estas matrices de relación representan la dependencia entre variables, donde una entrada $a_{ij} = 1$ (fila i y columna j) para cualquier matriz de relación A representa que la variable i depende de la variable j. Estas cuatro matrices (0, 1) se muestran en las tablas 1, 2, 3 y 4.

Tabla 1: Matriz M^r con dependencias entre estados actuales y pasados.

	$x_1(t)$	$x_2(t)$	$\cdots$	$x_s(t)$
$x_1(t+1)$	M^r_{11}	M^r_{12}	$\cdots$	M^r_{1s}
$x_2(t+1)$	M^r_{21}	M^r_{22}	$\cdots$	M^r_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_s(t+1)$	M^r_{s1}	M^r_{s2}	$\cdots$	M^r_{ss}

Tabla 2: Matriz M^i con dependencias entre estados y entradas.

	$u_1(t)$	$u_2(t)$	$\cdots$	$u_n(t)$
$x_1(t+1)$	M^i_{11}	M^i_{12}	$\cdots$	M^i_{1n}
$x_2(t+1)$	M^i_{21}	M^i_{22}	$\cdots$	M^i_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_s(t+1)$	M^i_{s1}	M^i_{s2}	$\cdots$	M^i_{sn}

Tabla 3: Matriz M^{h2} con dependencias entre salidas y estados.

	$x_1(t)$	$x_2(t)$	$\cdots$	$x_s(t)$
$y_1(t)$	M^{h2}_{11}	M^{h2}_{12}	$\cdots$	M^{h2}_{1s}
$y_2(t)$	M^{h2}_{21}	M^{h2}_{22}	$\cdots$	M^{h2}_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$y_m(t)$	M^{h2}_{m1}	M^{h2}_{m2}	$\cdots$	M^{h2}_{ms}

A partir de estas matrices, podemos expandir cada elemento y generar una matriz particionada por bloques donde cada elemento es una submatriz J o O de dimensiones apropiadas.

Por ejemplo, para M^r, si $M^r_{ij} = 1$, la expandimos a $W^{r,flag}_{ij} = J_{h_i,1}$. Si $M^r_{ij} = 0$, la expandimos a $W^{r,flag}_{ij} = O_{h_i,1}$. Aplicamos esto para $i, j = 1, \ldots, s$. De esta forma, definimos $W^{r,flag}$ como una matriz particionada por bloques con s particiones fila:

$$W^{r,flag} = \begin{pmatrix} W^{r,flag}_{11} & W^{r,flag}_{12} & \cdots & W^{r,flag}_{1s} \\ W^{r,flag}_{21} & W^{r,flag}_{22} & \cdots & W^{r,flag}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^{r,flag}_{s1} & W^{r,flag}_{s2} & \cdots & W^{r,flag}_{ss} \end{pmatrix} \quad (9)$$

y W^r puede ser estructurada como:

$$W^r = \begin{pmatrix} W^r_{11} & W^r_{12} & \cdots & W^r_{1s} \\ W^r_{21} & W^r_{22} & \cdots & W^r_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^r_{s1} & W^r_{s2} & \cdots & W^r_{ss} \end{pmatrix} \circ \begin{pmatrix} W^{r,flag}_{11} & W^{r,flag}_{12} & \cdots & W^{r,flag}_{1s} \\ W^{r,flag}_{21} & W^{r,flag}_{22} & \cdots & W^{r,flag}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^{r,flag}_{s1} & W^{r,flag}_{s2} & \cdots & W^{r,flag}_{ss} \end{pmatrix} \quad (10)$$

donde las submatrices nulas *podan* conexiones que no son relevantes según indique la tabla 1 y, por tanto, los pesos correspondientes no necesitan ser entrenados. Eso significa que la dimensión del problema de optimización para el entrenamiento de la

Tabla 4: Matriz M^u con dependencias entre salidas y entradas.

	$u_1(t)$	$u_2(t)$	$\cdots$	$u_n(t)$
$y_1(t)$	M^u_{11}	M^u_{12}	$\cdots$	M^u_{1n}
$y_2(t)$	M^u_{21}	M^u_{22}	$\cdots$	M^u_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$y_m(t)$	M^u_{m1}	M^u_{m2}	$\cdots$	M^u_{mn}

red neuronal se reduce significativamente.

Para M^i, si $M^i_{ij} = 1$, la expandimos a $W^{i,flag}_{ij} = J_{h_i,1}$. Si $M^i_{ij} = 0$, la expandimos a $W^{i,flag}_{ij} = O_{h_i,1}$. Aplicamos esto para $i = 1,\ldots,s$, $j = 1,\ldots,n$. De esta forma, definimos $W^{i,flag}$ como una matriz particionada por bloques con s particiones fila:

$$W^{i,flag} = \begin{pmatrix} W^{i,flag}_{11} & W^{i,flag}_{12} & \cdots & W^{i,flag}_{1n} \\ W^{i,flag}_{21} & W^{i,flag}_{22} & \cdots & W^{i,flag}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^{i,flag}_{s1} & W^{i,flag}_{s2} & \cdots & W^{i,flag}_{sn} \end{pmatrix} \tag{11}$$

y W^i puede ser estructurada como:

$$W^i = \begin{pmatrix} W^i_{11} & W^i_{12} & \cdots & W^i_{1n} \\ W^i_{21} & W^i_{22} & \cdots & W^i_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^i_{s1} & W^i_{s2} & \cdots & W^i_{sn} \end{pmatrix} \circ \begin{pmatrix} W^{i,flag}_{11} & W^{i,flag}_{12} & \cdots & W^{i,flag}_{1n} \\ W^{i,flag}_{21} & W^{i,flag}_{22} & \cdots & W^{i,flag}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^{i,flag}_{s1} & W^{i,flag}_{s2} & \cdots & W^{i,flag}_{sn} \end{pmatrix} \tag{12}$$

De la misma forma, para M^{h2}, si $M^{h2}_{ij} = 1$, la expandimos a $W^{h2,flag}_{ij} = J_{h2_i,1}$. Si $M^{h2}_{ij} = 0$, la expandimos a $W^{h2,flag}_{ij} = O_{h_i,1}$. Aplicamos esto para $i = 1,\ldots,m$, $j = 1,\ldots,s$. De esta forma, definimos $W^{h2,flag}$ como una matriz particionada por bloques con m particiones fila:

$$W^{h2,flag} = \begin{pmatrix} W^{h2,flag}_{11} & W^{h2,flag}_{12} & \cdots & W^{h2,flag}_{1s} \\ W^{h2,flag}_{21} & W^{h2,flag}_{22} & \cdots & W^{h2,flag}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^{h2,flag}_{m1} & W^{h2,flag}_{m2} & \cdots & W^{h2,flag}_{ms} \end{pmatrix} \tag{13}$$

y W^{h2} puede ser estructurada como:

$$W^{h2} = \begin{pmatrix} W^{h2}_{11} & W^{h2}_{12} & \cdots & W^{h2}_{1s} \\ W^{h2}_{21} & W^{h2}_{22} & \cdots & W^{h2}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^{h2}_{m1} & W^{h2}_{m2} & \cdots & W^{h2}_{ms} \end{pmatrix} \circ \begin{pmatrix} W^{h2,flag}_{11} & W^{h2,flag}_{12} & \cdots & W^{h2,flag}_{1s} \\ W^{h2,flag}_{21} & W^{h2,flag}_{22} & \cdots & W^{h2,flag}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ W^{h2,flag}_{m1} & W^{h2,flag}_{m2} & \cdots & W^{h2,flag}_{ms} \end{pmatrix} \tag{14}$$

Finalmente, para M^u, si $M^u_{ij} = 1$, la expandimos a $W^{u,flag}_{ij} = J_{h2_i,1}$. Si $M^u_{ij} = 0$, la expandimos a $W^{u,flag}_{ij} = O_{h2_i,1}$. Aplicamos esto para $i = 1,\ldots,m$, $j = 1,\ldots,n$. De esta forma, definimos $W^{u,flag}$ como una matriz particionada por bloques con m particiones fila:

$$W^{u,flag} = \begin{pmatrix} W^{u,flag}_{11} & W^{u,flag}_{12} & \cdots & W^{u,flag}_{1n} \\ W^{u,flag}_{21} & W^{u,flag}_{22} & \cdots & W^{u,flag}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^{u,flag}_{m1} & W^{u,flag}_{m2} & \cdots & W^{u,flag}_{mn} \end{pmatrix} \tag{15}$$

y W^u puede ser estructurada como:

$$W^u = \begin{pmatrix} W^u_{11} & W^u_{12} & \cdots & W^u_{1n} \\ W^u_{21} & W^u_{22} & \cdots & W^u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^u_{m1} & W^u_{m2} & \cdots & W^u_{mn} \end{pmatrix} \circ \begin{pmatrix} W^{u,flag}_{11} & W^{u,flag}_{12} & \cdots & W^{u,flag}_{1n} \\ W^{u,flag}_{21} & W^{u,flag}_{22} & \cdots & W^{u,flag}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W^{u,flag}_{m1} & W^{u,flag}_{m2} & \cdots & W^{u,flag}_{mn} \end{pmatrix} \tag{16}$$

De esta forma, el sistema dinámico no lineal con información estructurada puede ser descrito por

$$\vec{x}(t+1) = \mathbf{f}(M^r\vec{x}(t), M^i\vec{u}(t)) \tag{17}$$

$$\vec{y}(t) = \mathbf{g}(M^{h2}\vec{x}(t), M^u\vec{u}(t)) \tag{18}$$

y las ecuaciones de la Red Neuronal Estructurada en el Espacio de Estados (RNEEE) permanecen igual que 1 y 2, pero las matrices W^h, W^o, W^r, W^i, W^{h2} y W^u son matrices particionadas por bloques tal y como están expresadas en las ecuaciones 7, 8, 10, 12, 14 y 16.

El procedimiento, por tanto, para diseñar la estructura neuronal, es, en primer lugar, deducir, a partir de un modelo de conocimiento del proceso, las relaciones entre estados y entradas, y entre

salidas y estados. Estas relaciones se reflejan en las matrices M^r, M^i, M^{h2}, M^u. A partir de estas matrices, el resto de pasos se pueden automatizar: generación de las matrices de pesos particionadas por bloques y entrenamiento de la red neuronal resultante con datos de proceso.

3. CASO DE ESTUDIO

3.1. DESCRIPCIÓN DEL DEPÓSITO CALEFACTADO

Usaremos un sistema de ejemplo sencillo donde ilustrar la aplicación de la RNEEE propuesta. El sistema, que se muestra en la figura 3, está compuesto de un depósito, cuyo contenido es calentado por medio de una resistencia eléctrica. El depósito tiene una entrada en la parte superior que permite al fluido entrar al depósito. El fluido abandona el depósito a través de una válvula en la parte inferior del depósito. La apertura de la válvula puede ser modificada para permitir extraer más o menos fluido del depósito. El fluido es calentado por medio de una resistencia eléctrica, y el flujo de calor aportado al fluido puede ser modificado con el voltaje aplicado a la resistencia. El nivel del tanque, h, el caudal de entrada W_i, la apertura de la válvula Op, y las temperaturas de entrada y salida T_i y T_o son variables medidas.

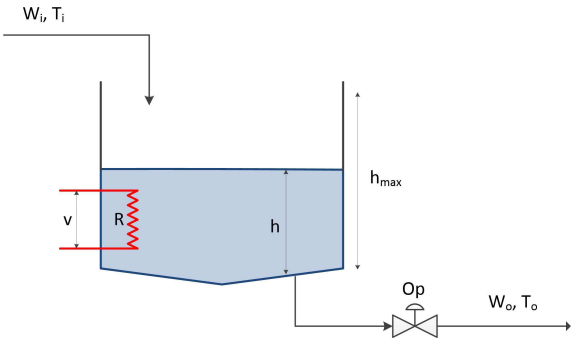

Figura 3: Ejemplo de depósito calefactado

Se supone condiciones de mezcla perfecta dentro del depósito, por lo que la temperatura es homogénea dentro del tanque e igual a la temperatura del fluido a la salida T_o. La densidad del fluido se considera constante y la entalpía específica del fluido se considera dependiente de la temperatura a través de una relación no lineal. Con estas suposiciones, las ecuaciones que modelan el comportamiento del sistema se muestran a continuación:

$$A \cdot \frac{dh}{dt} = W_i - W_o \quad (19)$$

$$A \cdot h \cdot \rho \frac{dH_o}{dt} = W_i \cdot \rho \cdot (H_i - H_o) + Q \quad (20)$$

$$H_i = f_{NL1}(T_i) \quad (21)$$

$$T_o = f_{NL2}(H_o) \quad (22)$$

$$W_o = k \cdot Op \cdot \sqrt{h} \quad (23)$$

$$Q = \frac{V^2}{R} \quad (24)$$

Siendo A el área transversal del depósito (m^2), ρ es la densidad del fluido (kg/m^3), h es la altura del depósito (m), W son los flujos volumétricos (m^3/s), T son las temperaturas del fluido (K), H son las entalpías específicas del fluido (J/kg), los subíndices $_i$ y $_o$ se refieren a los caudales de entrada y salida, Q es el flujo de calor aportado por la resistencia (W), V es el voltaje de la resistencia (V), R es el valor de la resistencia (Ω), f_{NL1} y f_{NL2} son ecuaciones no lineales que relacionan la temperatura con la entalpía del fluido (correlación directa e inversa, respectivamente), k es un parámetro de ajuste para el cálculo del caudal de salida y Op es la apertura de la válvula (%).

Evidentemente, si disponemos del modelo del sistema en forma de ecuaciones diferenciales, una alternativa a lo aquí planteado sería realizar un ajuste de los parámetros que figuran en el modelo, pero hay que tener en cuenta que en muchos casos reales hay una incertidumbre o desconocimiento de las relaciones explícitas entre variables que hacen inviable esa alternativa o directamente el modelo es demasiado complejo lo que aumenta significativamente la carga computacional de las simulaciones necesarias para ajustar los parámetros del modelo a los datos reales. El ajuste de parámetros de un modelo DAE (Differential and Algebraic Equations) complejo requiere un esfuerzo de desarrollo considerable y la solución del problema no es sencilla cuando el número de parámetros es elevado, mientras que el esfuerzo de desarrollo de la red es mucho menor. En este caso de estudio se dispone del modelo completo, pero en sistemas reales, con modelos matemáticos complejos, no se dispone de un modelo del sistema, sino de un modelo básico, que puede contener algunas ecuaciones diferenciales aproximadas, que a su vez contienen relaciones entre las entradas y salidas del sistema con los estados y ecuaciones que relacionan variables, que pueden ser ecuaciones de correlación o incluso cualitativas. Estas ecuaciones proporcionan cierta información del sistema y de sus relaciones, pero no son suficientes para representar de manera fidedigna el comportamiento del sistema; sin embargo, sí que se podría utilizar la información que proporcionan para diseñar una RNEEE.

El caso de estudio aquí presentado se ha querido mantener sencillo a propósito para ilustrar la técnica propuesta.

A la vista de las ecuaciones del sistema (o en otros casos por conocimiento físico de las relaciones entre variables), podemos establecer un modelo en el dominio discreto con las dependencias específicas para los estados h y H_o, y la salida de interés T_o:

$$h(t+1) = f_1(h(t), Op(t), W_i(t)) \quad (25)$$

$$H_o(t+1) = f_2(h(t), H_o(t), W_i(t), T_i(t), V(t)) \quad (26)$$

$$T_o(t) = g(H_o(t)) \quad (27)$$

A partir de estas relaciones, y estableciendo como vector de entrada $\vec{u} = [Op, W_i, T_i, V]'$, como vector de estados $\vec{x} = [h, H_o]'$ y vector de salida $\vec{y} = [T_o]$, podemos construir las matrices de relación M^r, M^i, M^{h2} y M^u:

$$M^r = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (28)$$

$$M^i = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad (29)$$

$$M^{h2} = \begin{pmatrix} 0 & 1 \end{pmatrix} \quad (30)$$

$$M^u = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \quad (31)$$

3.2. DISEÑO DE LA RNEEE

Las ecuaciones 25, 26 y 27 permiten establecer una red neuronal estructurada como la figura 4 donde se ve claramente que solamente existen aquellas conexiones que dicta el modelo de conocimiento, estableciendo una estructura neuronal más parecida a las relaciones que prevé el modelo de conocimiento planteado. Las relaciones entre variables siguen siendo modeladas por las capas no lineales como relaciones no lineales a ajustar, pero la estructura es más adecuada para representar el sistema, con muchos menos parámetros (pesos de la red neuronal) a ajustar que un modelo genérico. En concreto, si cada capa oculta no lineal constara de 10 neuronas, en total la reducción de parámetros sería de un 30 % aproximadamente.

3.3. DATOS EXPERIMENTALES

Para generar datos del proceso con los que entrenar la RNEEE, en primer lugar vamos a parametrizar el sistema como se indica en la tabla 5. El sistema ha sido simulado en entorno Matlab [5].

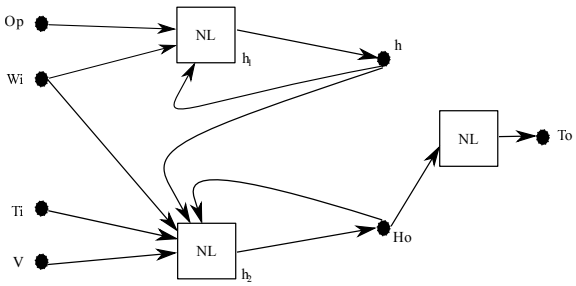

Figura 4: RNEEE que modela el depósito calefactado

Tabla 5: Parámetros del depósito calefactado.

Parámetro	Valor
A	2,25 dm^3
ρ	1 l/kg
k	10^{-3}
R	50 Ω

En cuanto a las entradas, supondremos que el caudal de entrada W_i se mantiene normalmente constante, pero generaremos experimentos con diversos valores del caudal en el rango $[2, 4]$ l/min. Realizaremos la misma suposición sobre la temperatura de entrada T_i, considerando valores en el rango $[13, 17]$ °C. En cuanto a la apertura de la válvula Op y el voltaje V aplicado a la resistencia, generaremos señales para identificación, con valores aleatorios (dentro de un intervalo $[10, 90]$ % para Op y de $[0, 220]$ V para V) que se mantienen constantes durante un tiempo aleatorio (fracción del tiempo de establecimiento del sistema: 200 segundos aproximadamente). El sistema parte de unas condiciones iniciales en las que la altura del depósito se encuentra en 10 cm y la entalpía de salida del fluido es de 63 kJ/kg, lo cual corresponde con una temperatura de 15 °C. El sistema está en equilibrio inicialmente. El periodo de muestreo de los datos es de 20 segundos.

Con estas premisas, generamos un experimento para obtener datos con los que entrenar la red neuronal tratando de cubrir varios puntos de funcionamiento. Estos datos pueden verse en la figura 5.

De forma similar, se genera un conjunto de datos para validación.

3.4. ENTRENAMIENTO DE LA RNEEE

Una vez definida la estructura de la red neuronal y generados los datos de entrenamiento, se procede al entrenamiento de la misma. Como se indicó en la sección 1, se ha utilizado un algoritmo estocástico cuyo objetivo es minimizar la suma de

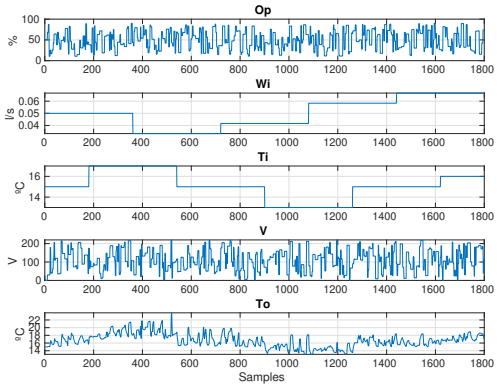

Figura 5: Datos de entrenamiento

errores al cuadrado entre la salida generada por la red y la temperatura real del sistema, teniendo como variables independientes el conjunto de pesos y bias de la red.

Repitiendo el entrenamiento 10 veces para cubrir posibles problemas de convergencia dependiendo de las condiciones iniciales, obtuvimos un RMSE (Root Mean Squared Error, raíz cuadrada del error cuadrático medio) de unos $0,4$ °C.

Seleccionando los pesos de la red neuronal que nos dio mejores resultados en el entrenamiento, procedimos a generar la salida de la red con los datos de validación, obteniendo en este caso un RMSE de $0,5$ °C. Como se puede apreciar en la figura 6, los resultados son excelentes excepto para temperaturas muy altas (por encima de 22 °C), zona de operación donde la red no dispuso de mucha información en el entrenamiento.

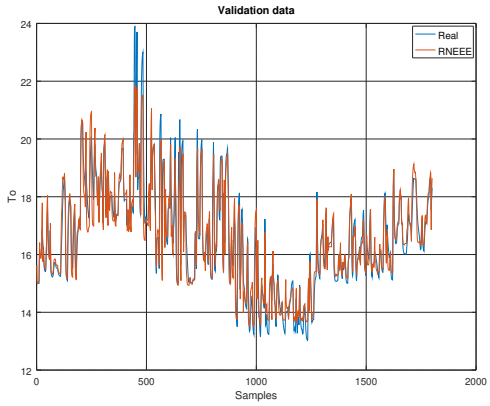

Figura 6: Resultados de la RNEEE sobre datos de validación

4. CONCLUSIONES Y TRABAJO FUTURO

En este artículo se ha mostrado una evolución de la Red Neuronal en el Espacio de Estados en la que se introduce información estructurada proveniente de un modelo de conocimiento (o cualquier otra información que se disponga sobre el sistema), obteniendo de este modo una red estructurada, pasando de un modelo de caja negra a uno de caja gris (las dependencias entre variables provienen de información por conocimiento del proceso y el aprendizaje de las relaciones se lleva a cabo con estructuras neuronales). Se ha mostrado cómo, partiendo de la información estructural del modelo, se pueden definir las matrices de pesos en las que muchos de sus elementos son cero.

Los resultados obtenidos en un caso de estudio sencillo son excelentes, comprobándose que la capacidad de la red neuronal en la representación de sistemas dinámicos no se ha visto alterada. De hecho, realizamos una comparación con una red sin estructurar (mantenemos las matrices de pesos con todos sus elementos), y los resultados eran similares.

Además de que la RNEEE tiene la misma capacidad de representación que un modelo RNEE equivalente, nos falta por ver si tiene alguna ventaja adicional además de necesitar menos parámetros efectivos. A pesar de tener que ajustar menos parámetros, nuestros resultados indican que los tiempos de entrenamiento son similares, debido a que a la hora de generar la salida de la red debemos reconstruir las matrices completas (aunque muchos de sus elementos sean cero) y la mayor carga computacional se da precisamente en la simulación de la red neuronal. Tampoco el número de elementos a cero de las matrices es tan grande como para poder considerar matrices dispersas, no obteniéndose ninguna ventaja computacional con ese supuesto.

Adicionalmente, pretendemos llevar a cabo una experimentación sobre un sistema real similar al mostrado como caso de estudio, para comprobar si la capacidad de representación sigue siendo buena sobre datos reales, teniendo en cuenta las limitaciones de aproximar cualquier sistema real por un modelo de representación del mismo. De esta forma, podríamos realizar una comparativa entre el modelo RNEEE y, por ejemplo, un modelo de conocimiento en forma de DAEs, en cuanto a los errores de modelado, carga computacional, tiempos de ajuste, etc.

Agradecimientos

Los autores agradecen el apoyo del Ministerio

de Economía y Competitividad/FEDER a través del proyecto DPI2015-67341-C2-2-R. Asimismo, el primer autor agradece el apoyo de la Junta de Castilla y León y EU/FEDER (CLU 2017-09).

English summary

STRUCTURED STATE SPACE NEURAL NETWORKS AS GREY-BOX MODELS

Abstract

State space neural networks (ssNN) has demonstrated very good properties when modelling dynamic systems in the past. In this paper we propose an evolution of the neural network when information about the inner structure of the system is available in the form any kind of model. With this information, a grey-box model is obtained that represents in a better way the system to be modelled. This model has been named structured state space neural network (sssNN). A simulated example is presented as a case study.

Keywords: State space model, Neural network, Grey-box model.

Referencias

[1] P. González-Lanza, J. Zamarreño, A hybrid method for training a feedback neural network, in: First International ICSC-NAISO Congress on Neuro Fuzzy Technologies NF 2002, Havana - Cuba, 2002.

[2] P. González Lanza, J. Zamarreño, A short-term temperature forecaster based on a state space neural network, Engineering Applications of Artificial Intelligence 15 (5) (2002) 459 – 464. doi:10.1016/S0952-1976(02)00089-1.

[3] M.M. Gupta, L. Jin, N. Homma, Static and dynamic neural networks. From fundamentals to advanced theory, John Wiley & Sons, New Jersey (2003).

[4] S. Haykin, Neural networks. A comprehensive foundation (2nd ed.), Prentice-Hall, New Jersey (1999).

[5] Mathworks, Matlab, `http://es.mathworks.com` (acceso 15 de mayo de 2019).

[6] O. Nelles, Nonlinear system identification. From classical approaches to neural networks and fuzzy models, Springer-Verlag, Berlin (2001).

[7] F. Solis, R. J.-B. Wets, Minimization by random search techniques, Mathematics of Operations Research 6 (1981) 19–30.

[8] J. Zamarreño, P. Vega, Identification and predictive control of a melter unit used in the sugar industry, Artificial Intelligence in Engineering 11 (4) (1997) 365 – 373. doi:10.1016/S0954-1810(96)00055-6.

[9] J. Zamarreño, P. Vega, State space neural network. Properties and application, Neural Networks 11 (6) (1998) 1099–1112.

[10] J. Zamarreño, P. Vega, Neural predictive control. Application to a highly non-linear system, Engineering Applications of Artificial Intelligence 12 (2) (1999) 149 – 158. doi:https://doi.org/10.1016/S0952-1976(98)00055-4.

[11] J. Zamarreño, P. Vega, L. García, M. Francisco, State-space neural network for modelling, prediction and control, Control Engineering Practice 8 (9) (2000) 1063 – 1075. doi:10.1016/S0967-0661(00)00045-9.