Modelación de Sistemas Dinámicos Utilizando Redes Neuronales en su Forma Canónica

Luz Mestre Torres, Gonzalo Acuña Leiva
Universidad de Santiago de Chile, Departamento de Ingeniería Informática,
Santiago, Chile
lmestre@diinf.usach.cl, gacuna@usach.cl

Abstract

Dynamic system modelling has been mainly performed using recurrent neural networks (RNN). However, training algorithms must be specific for each model thus making RNN not easy to use. In this work, the canonical transformation of RNN, as proposed by Dreyfus and Idan, is used thus allowing the utilization of traditional and simple training algorithms like backpropagation. An application for a CSTR dynamic process is included showing very good one-step-ahead (OSA) and multiple-prediction-output (MPO) prediction results.

Keywords: Recurrent Neural Networks, Canonical Form, Dynamic Systems, Complex Systems.

Resumen

La modelación de sistemas dinámicos con redes neuronales sugiere el uso de redes recurrentes. Sin embargo, los algoritmos de entrenamiento para dichas redes deben ser específicos para cada modelo, lo cual complica su uso. Este trabajo aplica el método propuesto por Dreyfus e Idan donde se transforma una red neuronal recurrente, obtenida a partir de un modelo de ecuaciones diferenciales, a una red en su forma canónica, lo cual permite el uso de algoritmos de entrenamiento tradicionales como retropropagación del error. Se realiza una implementación computacional de dicho método, la cual se utiliza para modelar en un sistema dinámico no lineal, donde se logran muy buenos resultados en la predicción, utilizando tanto predicción a un paso (OSA) como predicción múltiple (MPO).

Palabras claves: Redes Neuronales Recurrentes, Formas Canónicas, Sistemas Dinámicos, Sistemas Complejos.

1 Introducción

Una Red Neuronal está formada por un conjunto de unidades de procesamiento denominadas neuronas. Cada neurona actúa como un elemento procesador independiente, las entradas y los pesos de interconexión son procesados por una función suma, cuyo resultado es mapeado por una función de transferencia no lineal, la cual es la salida de la neurona[3].

El conocido algoritmo de retropropagación, para entrenamiento de redes neuronales multicapa, sólo puede aprender relaciones estáticas en una red neuronal. También se puede usar retropropagación del error para realizar predicciones no lineales de una serie de tiempo estacionaria. Aún así, esta red representa sólo un modelo estático, cuyos parámetros tienen valores fijos. Para que una red modele un sistema dinámico, debe poseer memoria[3][9].

La capacidad de caja negra de las redes recurrentes de tiempo discreto para modelar procesos dinámicos ha sido extensamente estudiada. En la mayoría de los casos, los modelos dinámicos son modelos de entrada y salida, consistentes de una red prealimentada, cuya salida alimenta la entrada con uno o varios retardos. Tal arquitectura puede ser entrenada fácilmente utilizando el algoritmo de retropropagación. Sin embargo, en el modelamiento pueden surgir arquitecturas mucho más complicadas, con dinámica dentro de la red. En este caso existen dos alternativas: se puede implementar un algoritmo de entrenamiento para la arquitectura específica, o transformar la red en su equivalente forma canónica, la cual puede ser entrenada usando retropropagación del error[7].

En 1993, Nerrand et al.[7] define la forma canónica de una red neuronal, la cual permite transformar cualquier red recurrente en una red con recurrencia externa. Dicha forma canónica, permite clarificar la presentación de una red y hacer más fácil la implementación de los algoritmos de entrenamiento. En 1998 Acuña et al.[1], utiliza redes neuronales para elaborar modelos para la estimación en línea de la concentración de biomasa, utilizando la forma canónica presentada por Nerrand.

En 1998, Dreyfus e Idan[4] plantean una metodología basada en grafos para obtener esta forma canónica. Dicho método consiste en transformar un conjunto de ecuaciones de tiempo discreto a la forma canónica, lo cual se puede realizar en tres etapas: encontrar el orden del sistema, es decir, encontrar el mínimo número de variables que describen completamente el modelo; segundo encontrar el vector de estado; y finalmente transformar la ecuación de tiempo discreto en las ecuaciones que rigen las variables de estado.

En este trabajo se aplica la metodología para transformar una red recurrente a la forma canónica definida por Nerrand et al.[7]. Se realiza una implementación computacional del método propuesto por Dreyfus e Idan[4], que permite obtener la forma canónica de cualquier red recurrente expresada como un conjunto de ecuaciones de estado. Finalmente se utiliza dicha implementación para modelar un sistema dinámico no lineal, utilizando redes neuronales.

2 La Forma Canónica de un Modelo no Lineal de Tiempo Discreto

2.1 Introducción

La dinámica de una red recurrente de tiempo discreto puede ser descrita por una ecuación de diferencias finitas de orden *N*, la cual puede ser expresada por un conjunto de *M* ecuaciones de diferencias de primer orden, que contengan *N* variables de estado junto a *M* variables de entrada.

Un sistema lineal descrito por su función de transferencia, puede ser representado de muchas formas en el espacio de estado. Todas estas representaciones son equivalentes, pero algunas tienen propiedades que las hacen más fácilmente tratables que otras. Tales representaciones son llamadas formas canónicas. Para modelos no lineales, el término forma canónica no tiene un significado universal. Se considerará que un modelo de tiempo discreto se encuentra en su forma canónica si:

$$z(n+1) = j\left[z(n), u(n)\right] \tag{1}$$

$$y(n+1) = y\left[z(n+1)\right] \tag{2}$$

donde *u(n)* es el vector de entradas externas, *z(n)* es el mínimo conjunto de *v* variables necesarias para caracterizar completamente el modelo de estado al tiempo *n+1*, si el estado del modelo y *u(n)* son conocidas al tiempo *n*, *y(n)* es el vector de salida.

La parte dinámica de la forma canónica puede ser representada mediante una red neuronal prealimentada descrita por la función φ, cuyas entradas son las variables de estado al tiempo *n*. La salida en el tiempo *n+1* es determinada a partir de las variables de estado al tiempo *n+1* y puede ser representada por una red prealimentada que implementa la función ψ.

Así, cualquier red recurrente puede ser transformada a su forma canónica la cual consiste en una red con recurrencia externa, como se muestra en la figura 1. Las entradas a la red son las variables manipuladas(entradas externas) y los valores de las variables de estado de salida las cuales son realimentadas y retardadas en una unidad. Las salidas

corresponden por una parte a aquellas variables de estado medibles, es decir, que pueden tener un valor deseado, las restantes salidas de la red son las variables de estado no medibles.

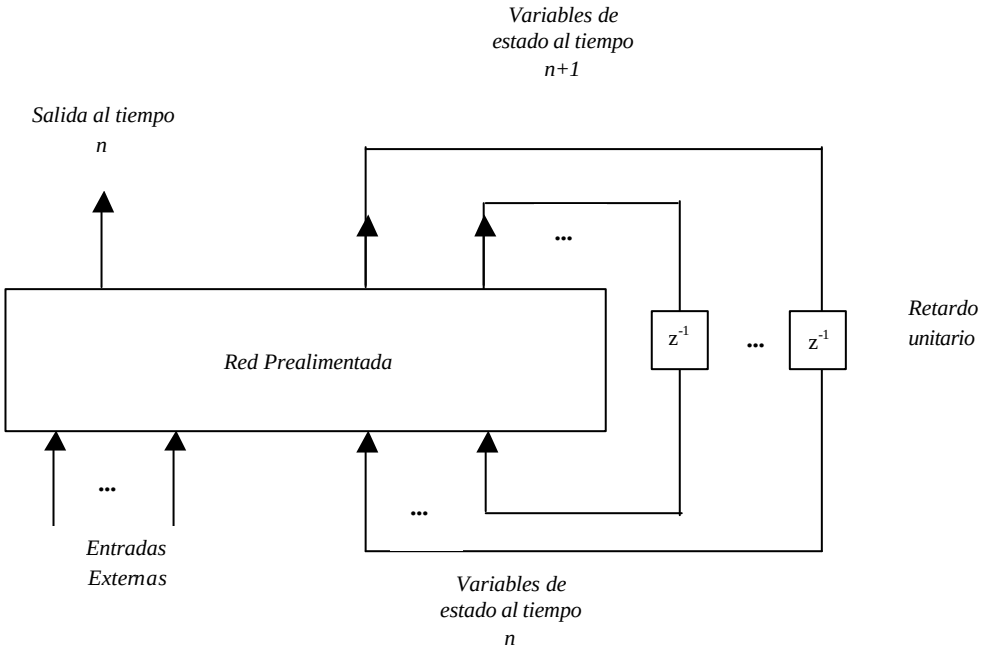


Figura 1: *Forma canónica de una red neuronal recurrente.*

Para transformar un conjunto de ecuaciones discretas a la forma canónica, se puede utilizar una representación mediante un grafo dirigido, la cual se describe en la sección 2.2. En las secciones 2.3 y 2.4 se describe las etapas de esta transformación: el cálculo del orden del modelo y la determinación el vector de estado.

2.2 Representación de un Modelo Dinámico Mediante un Grafo

A continuación se presenta una analogía entre los elementos de un grafo y los de un sistema dinámico. Sea un grafo dirigido G formado por un conjunto de aristas E y un conjunto de vértices V. Se define v_i como un vértice que representa una variable x_i; e_{ij} es una arista dirigida desde un vértice v_j a otro v_i, representa una relación entre dos variables; el número de aristas paralelas desde un vértice v_j a otro v_i es igual al número de retardos diferentes $t_{ij,k}$; $\{R_i\}$ denota el conjunto de aristas salientes del vértice v_i; M_i es el largo de la arista entrante a v_i de máximo largo; $c(v_i)$ es el número de ciclos, es decir, caminos que empiezan y terminan en el mismo vértice, incluido el vértice v_i; $c(e_{ij})$ es el número de ciclos que incluyen la arista e_{ij}; y finalmente se define A_{ji} como el número de aristas e_{ij} desde el vértice v_j a otro v_i.

Utilizando esta representación, a partir del modelo dinámico se puede obtener directamente el grafo G_0, transformando las variables en vértices y manteniendo las relaciones en el tiempo a través de aristas.

2.3 Cálculo del Orden del Modelo

Se desea reducir el grafo inicial G_0 a un grafo G_1, el cual contiene sólo los vértices que representan variables de estado y tiene el mismo número de variables, pero no necesariamente las mismas que el modelo descrito por G_0. A partir de G_1 se puede obtener el orden del modelo. Las simplificaciones del grafo G_0 se basan en que las aristas del grafo que no están dentro de ciclos no son necesarias para determinar el orden de la red, y además, en que las relaciones estáticas no son necesarias para la determinación del vector de estado.

Si x_m depende de las entradas externas y x_j solamente, y x_j depende de las entradas externas y x_i solamente y además, no hay otra variable más que x_m que dependa de x_j, entonces la variable x_j puede ser eliminada del modelo por sustitución de x_m sin cambiar el orden del modelo. Nerrand et al.[7] probó que el orden v del modelo representado por el grafo G_1 es dado por:

$$v = \sum_i w_i \tag{3}$$

donde:

$$\forall v_i \in G_1, w_i = \begin{cases} M_i - min_{c_{ji} \in \{R_i\}} (M_j - \tau_{ji}), & si \quad M_i - min_{c_{ji} \in \{R_i\}} (M_j - \tau_{ji}) > 0 \\ 0 & en\ otro\ caso \end{cases} \quad (4)$$

Se considera que existe un valor w_i por cada vértice de G_1.

Cuando existen varias aristas paralelas, sólo la arista de máximo largo es relevante para determinar el orden del modelo.

A partir de la metodología propuesta por los autores se puede plantear el siguiente algoritmo para determinar el grafo G_1:

1. Eliminar todas las aristas de G_0 que no pertenecen a ningún ciclo. Borrar los vértices cuyas aristas entrantes son de largo cero, y recombinar sus aristas entrantes y salientes.
2. Mientras no haya más cambios posibles.

   Si un vértice tiene un conjunto de aristas paralelas entrantes solamente y un conjunto de aristas paralelas salientes solamente, borrar el vértice y combinar cada par en una arista cuyo largo sea la suma de los largos de las aristas combinadas.

   Fin mientras
3. Mientras no hayan cambios posibles.

   Si existen varias aristas paralelas entre dos vértices, borrarlas todas excepto la de mayor largo.

   Fin Mientras

2.4 Determinación del Vector de Estado

El vector de estado, puede ser expresado de la siguiente forma:

$$z(n) = \left[x_1(n-k_1)...x_1(n-k_1-w_1+1)\,x_2(n-k_2)...x_2(n-k_2-w_2+1)...x_{N_v}(n-k_{N_v}-w_{N_v}+1).\right]^T \quad (5)$$

donde k_i y w_i son enteros no negativos, N_v es el número de vértices de G_1, y w_i es el número de ocurrencias de la variable x_i en el vector de estado.

Si $w_i=0$, entonces la variable x_i del modelo no es una variable de estado y los correspondientes k_i no requieren ser calculados. En otro caso k_i denota el retardo de la más reciente ocurrencia de la variable x_i en el vector de estado $z(n)$ Los valores de w_i deben cumplir con la condición indicada en la ecuación (3).

En la forma canónica, el retardo entre dos sucesivas variables de estado provenientes del mismo vértice es igual a uno. De esta forma existen varias representaciones canónicas ya que los w_i pueden ser diferentes, siempre y cuando su suma sea igual al orden.

Así, se debe encontrar un conjunto de $2N_v$ enteros $[k_i, w_i]$. Para hacer esto, se utiliza un nuevo grafo, llamado grafo de restricciones de tiempo. Este grafo, llamado G_2, se deriva de G_0 al borrar todos los vértices y aristas que no son significativas con respecto a las restricciones de tiempo que las variables de estado deben satisfacer. La principal diferencia entre G_2 y G_1 es el hecho que para las limitaciones de tiempo, las aristas que no están dentro de ciclos pero expresan una relación entre ellos, se deben mantener porque son relevantes para escoger las variables de estado.

A partir de la metodología propuesta por los autores, se puede plantear el siguiente algoritmo para determinación de G_2:

1. Borrar todos los vértices cuyas aristas entrantes son de largo cero, y recombinar sus aristas entrantes y salientes.
2. Mientras no existan cambios posibles

   Si un vértice tiene solamente un conjunto de aristas entrantes y solamente un conjunto de aristas salientes, borrar los vértices y combinar las aristas en una sola arista cuyo largo es la suma de los largos de las aristas combinadas.

   FinMientras
3. Mientras no existan cambios posibles

   Si existen varias aristas paralelas entre dos vértices, borrar todas excepto la arista de mayor largo.

   FinMientras
4. Borrar todas las aristas que no tiene ambos vértices en al menos un ciclo

Este algoritmo es muy semejante a aquel utilizado para determinar G_1. Las variables de estado son obtenidas a partir los vértices del grafo resultante G_2, así dos enteros k_i y w_i ($k_i \geq 0$, $w_i > 0$) son asociados a cada vértice.

Las variables de estado del modelo, son los vértices que aparecen en G_2. Con esta información y además con el orden del modelo obtenido en la sección 2.3, se puede obtener el vector de estado, lo cual se describe a continuación.

Sea N_E el número de aristas en el grafo G_2 y e_{ji} una arista de G_2 de largo t_{ji}. Para la definición del vector de estado y la construcción del grafo G_2, debe ser posible calcular $x_j(n-k_j+1)$ desde una de las variables de estado, proveniente del vértice v_i, el cual está disponible al tiempo $n-k_j+1$. Estas variables deben haber sido calculadas en el vértice v_i al tiempo $n-k_j+1-t_{ji}$. Por consiguiente las siguientes relaciones se deben mantener si $t_{ji} \geq 0$

$$n - k_i - w_i + 1 + \tau_{ji} \leq n - k_j + 1 \leq n - k_i + \tau_{ji} \tag{6}$$

o equivalentemente:

$$k_j - w_i + \tau_{ji} \leq k_i \leq k_j + \tau_{ji} - 1 \tag{7}$$

Por lo tanto, un conjunto de $2N_e$ desigualdades con $2N_v$ variables enteras debe ser satisfechas.

Así el problema de encontrar las variables de estado y las ecuaciones de estado se puede tratar como un problema de optimización lineal entera: encontrar el conjunto de enteros w_i tal que su suma sea mínima, bajo el conjunto de restricciones expresadas por las desigualdades de la ecuación (7). El valor mínimo es conocido, y es igual a v.

Si la solución óptima no es la trivial, debe ser encontrada por métodos de optimización lineal como Simplex.

2.5 Implementación Computacional

Luego de analizar cada uno de los pasos para la obtención de la forma canónica de una red recurrente, se puede plantear el siguiente algoritmo:

1. Construir G_0 a partir del conjunto de ecuaciones de tiempo discreto. Este paso permite determinar las entradas al problema: el conjunto de aristas y el conjunto de vértices.
2. Construir G_1 a partir de G_0. A partir de G_0, se eliminan aristas y vértices de lo cual se obtiene el orden n del modelo, es decir, el número de componentes del vector de estado.
3. Construir G_2 a partir de G_0. A partir de G_0, se eliminan aristas y vértices de lo que se obtiene las variables que aparecen en el vector de estado.
4. Obtener el vector de estado, a partir de G_1 y G_2. Se resuelve un problema de optimización lineal que permite determinar el vector de estado.

A partir de este algoritmo se implementó en MATLAB 6.1 un programa que realiza la transformación de cualquier red neuronal recurrente, a una red en su forma canónica. Las entradas utilizadas son la matriz de número de aristas entre cada vértice y la lista de aristas con sus retardos. La salida obtenida es el vector de estado, a partir del cual se puede construir, comenzando por G_0, la forma canónica buscada.

El programa determina el grafo G_1, luego G_2 y luego obtiene el vector de estado resolviendo un problema de optimización lineal entera, para lo cual se utilizó la función milp (Mixed Integer Linear Programming, presente en Matlog Version 5, Toolbox Logistics Engineering para MATLAB).

Se realizaron pruebas para redes recurrentes conocidas como Elman y Jordan[6], obteniéndose los mismos resultados que Nerrand et al.[7].

3 Modelamiento de un Sistema Dinámico Utilizando Redes Neuronales en su Forma Canónica

A continuación se modela con redes neuronales un sistema dinámico. Para esto se considera una reacción exotérmica de primer orden en un Reactor Continuo de Tanque Agitado o Continuous Stirred Tank Reactor (CSTR)[5]. La entrada al sistema es la temperatura de la camisa de enfriamiento, y la salida corresponde al grado de avance de la reacción.

El sistema es descrito por las siguientes ecuaciones:

$$\dot{x}_1 = -x_1 + 0.072\,(1 - x_1)\exp\left(\frac{x_2}{1 + x_2 / 20.0}\right) \tag{8}$$

$$\dot{x}_2 = -x_2 + 0.576(1-x_1)\exp\left(\frac{x_2}{1+x_2/20.0}\right) + 0.3(u-x_2) \tag{9}$$

$$y = x_1 \tag{10}$$

donde x_1 es el grado de avance de la reacción y x_2 es la temperatura adimensional del contenido del reactor. La entrada u es una tasa de flujo adimensional del fluido de transferencia de calor a través de la camisa de enfriamiento.

A partir de las ecuaciones (8),(9) y (10), se puede inferir que $x_1(t+1)=f_1(x_1(t), x_2(t))$ y $x_2(t+1)=f_2(x_1(t), x_2(t), u(t))$. Utilizando la representación indicada en la sección 2.2 se puede construir el grafo G_0, el cual se describe en la figura 2.a).

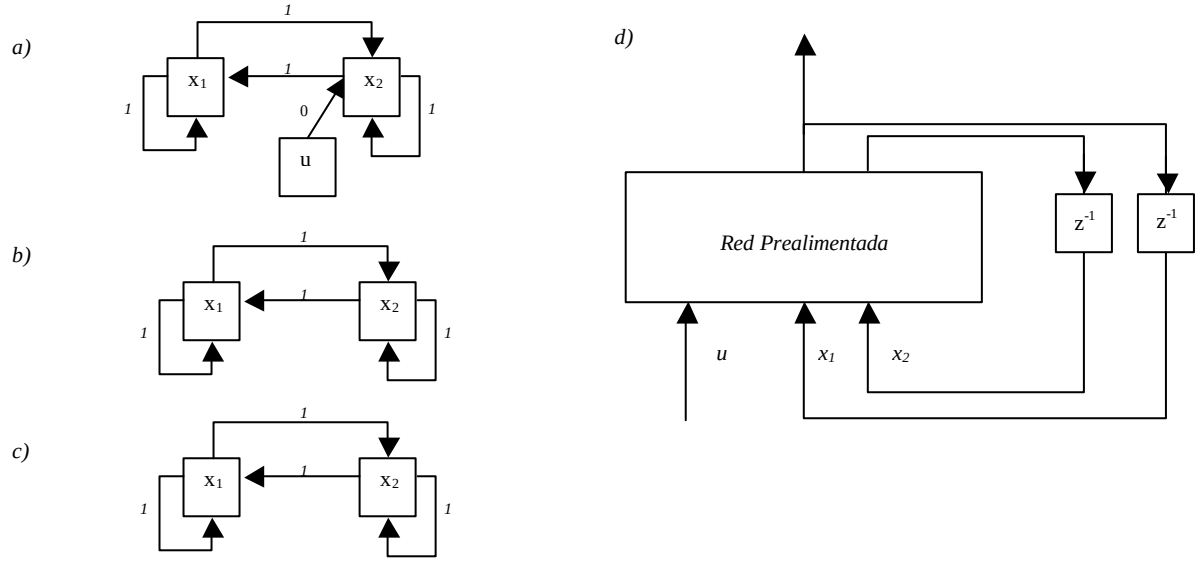


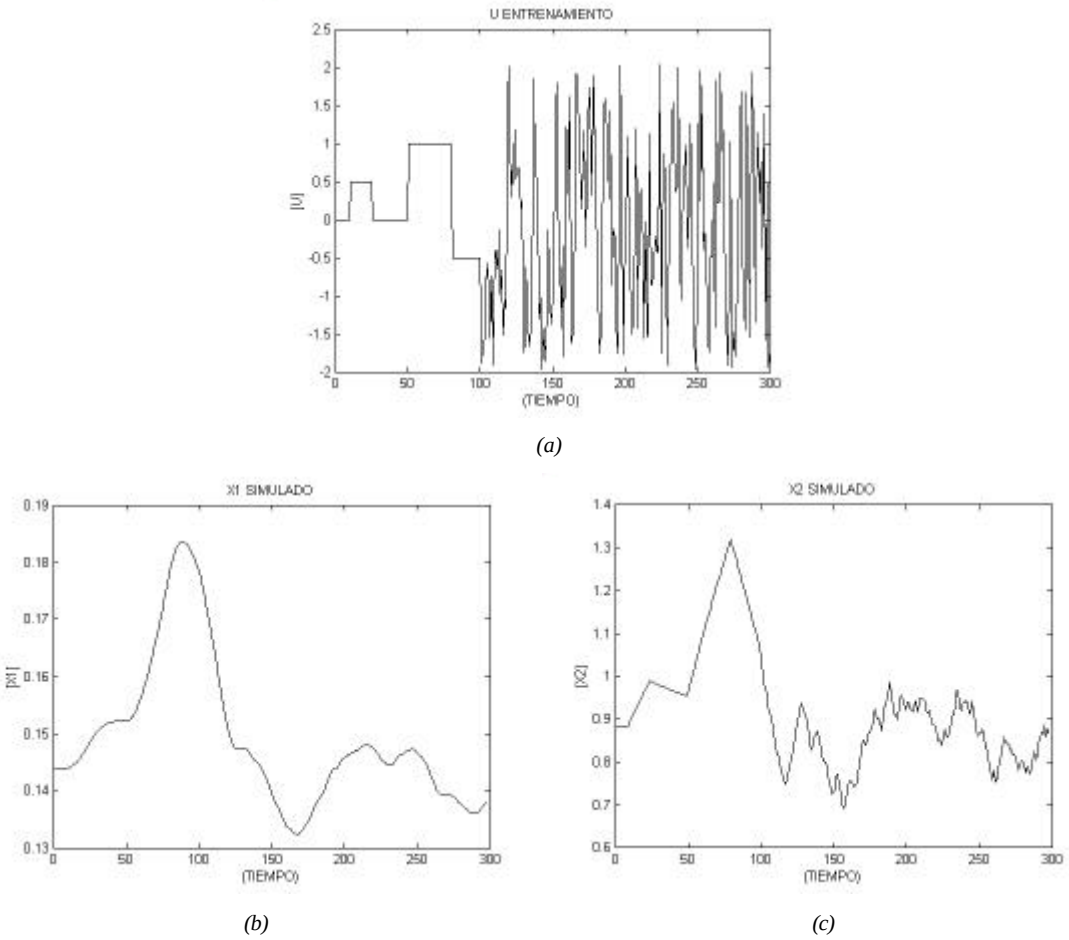
Figura 2: *Obtención de la forma canónica del modelo no lineal descrito en (8),(9) y (10): a) Grafo inicial G_0. b)Grafo G_1 del cual se determina que el orden del modelo es 2. c) Grafo G_2, el cual determina que las variables que aparecen en el vector de estado son x_1 y x_2. d) Forma canónica resultante.*

Los grafos G_1 y G_2 son construidos como se indica en las secciones 2.3 y 2.4 respectivamente, éstos son mostrados en la figura 2.b) y 2.c) respectivamente. En la construcción de G_1 la arista que une u con x_2 es eliminada, porque no pertenece a ningún ciclo. No existe ningún vértice que cumpla la condición de que todas las aristas entrantes sean de largo cero, por lo que no hay ninguna modificación. Los pasos 2 y 3 del algoritmo presentado en la sección 2.3 tampoco incorporan modificaciones. A partir de G_1 se obtiene que el orden del modelo es 2. En la construcción de G_2 indicado en la sección 2.4, los pasos 1, 2 y 3 no introducen modificaciones, y el paso 4 elimina la arista que une u con x_2. A partir de G_2 se obtiene que las variables que aparecerán en el vector de estado serán dos: x_1 y x_2. Resolviendo el problema de programación lineal que minimiza la suma de w_i, cuya suma es conocida, y es igual al orden del modelo, sujeto a la restricciones que introduce la ecuación (7), se obtiene que $w_1=1$, $w_2=1$, $k_1=0$ y $k_2=0$. Sustituyendo estos valores en la ecuación (5) se obtiene que el vector de estado es $z(n)=[x_1(n), x_2(n)]^T$.

La forma canónica obtenida se muestra en la figura 2.d). Las entradas de la red neuronal son x_1, x_2, u, y las salidas x_1, x_2. Las conexiones entre las entradas y salidas de la red pueden ser obtenidas a partir del grafo G_0 o también reemplazadas por capas ocultas[8].

Se entrenó utilizando la herramienta MATLAB 6.1 una red con 300 datos normalizados entre 0 y 1 para cada entrada y salida utilizando *trainlm* como función de entrenamiento de la red, la cual actualiza los valores de pesos y bias utilizando optimización por el método de *Levenberg-Marquardt*. La red posee una capa oculta con 5 unidades, y se realizaron 500 iteraciones. La función de transferencia para la capa oculta es *tansig* (tangente hiperbólica sigmoide) y para la capa de salida *purelin* (función de transferencia lineal).

En la figura 3.a) se muestra la entrada al sistema, descrita como una secuencia de escalones, seguida de la suma de una señal binaria pseudo aleatoria (PRBS) variando entre –1 y 1 y una variable aleatoria conocida con distribución uniforme entre –1 y 1. Esto permite generar datos que poseen una amplia información de la dinámica del sistema y acerca de su comportamiento en el estado estacionario. Las variables de estado del proceso se muestran en las figuras 3.b) y 3.c), con valores iniciales: $x_1=0.144$ y $x_2=0.885$.



(a)

(b) *(c)*

Figura 3: *Datos de entrenamiento del la red neuronal correspondientes al modelo descrito en (8),(9) y (10): a) entrada u consistente de una secuencia de escalones y una señal PRBS. Variables de estado: b)* x_1*. c)* x_2

Para medir la precisión de la predicción realizada por la red neuronal se emplea una predicción un paso hacia adelante de las salidas del sistema (One Step Ahead OSA) y la salida predicha a partir de las salidas del modelo (Model Predicted Output MPO)[2]. OSA es la medida más comúnmente utilizada y puede ser expresada de una forma general como:

$$\hat{y}(t) = f(u(t-1),..., u(t-n_u), y(t-1),..., y(t-n_y)) \tag{11}$$

donde *f* es una función no lineal. En este caso $\hat{y}(t)$ generalmente entrega una buena predicción de *y(t)* sobre el conjunto de estimación. MPO es una medida de mayor calidad de la capacidad predictiva del modelo utilizado, ya que predice la salida a partir de predicciones en tiempos anteriores. Se puede definir por:

$$\hat{y}_d(t) = f(u(t-1),..., u(t-n_u), \hat{y}_d(t-1),..., \hat{y}_d(t-n_y)) \tag{12}$$

Simulando el proceso con el modelo de ecuaciones diferenciales y por otra parte con el modelo neuronal, utilizando condiciones iniciales x_1=0.1 y x_2=0.8 y la entrada sinusoidal indicada en la figura 4.a) se alcanzan muy buenos resultados tanto utilizando OSA como MPO, los cuales se observan en las figuras 4.b) y 4.c) respectivamente.

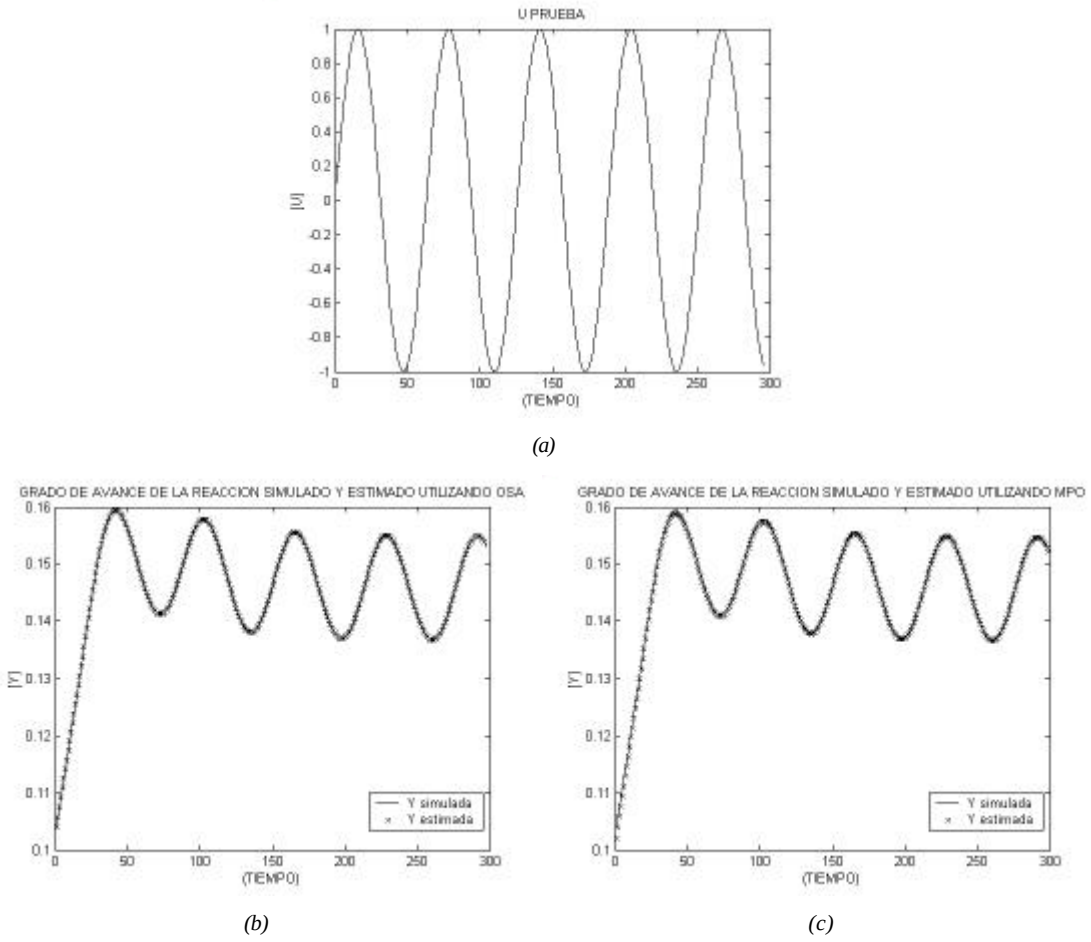


Figura 4: *Estimación del grado de avance de la reacción para el sistema descrito en (8),(9) y (10) utilizando la red en su forma canónica: a)Entrada para la fase de prueba. Salida simulada y estimada utilizando b) OSA. c) MPO.*

Se puede apreciar en la figura 4, que la red logra una muy buena calidad en la predicción, aún cuando la entrada al sistema utilizada en la fase de prueba es muy distinta a aquella empleada en el entrenamiento. Mas aún, la calidad de los resultados se mantiene para distintas condiciones iniciales de las variables de estado.

4 Conclusiones

Las redes recurrentes utilizan algoritmos de entrenamiento específicos para cada modelo, lo cual hace bastante complejo su uso. Otra forma de enfrentar este problema es llevar la red recurrente a su forma canónica, lo cual asegura poder utilizar un algoritmo de entrenamiento simple como retropropagación. Este trabajo utiliza una metodología basada en grafos para efectuar dicha transformación. Se ha modelado un sistema dinámico no lineal, el cual presenta una muy buena calidad de predicción utilizando predicciones un paso adelante (OSA) o a partir de condiciones iniciales (MPO), lo que ilustra la conveniencia de utilizar la transformación canónica propuesta, simplificando el entrenamiento de la red, lo que permite el modelamiento de complejos sistemas dinámicos no lineales.

Agradecimientos

Proyecto Fondecyt (Chile) 1010179 y Ecos- Conicyt (Francia-Chile) C99-B04.

Referencias

[1] Acuña, G., Latrille, E., Béal, C. and Corrieu, G. Static and Dynamic Neural Network Models for Estimating Biomass Concentration during Thermophilic Lactic Acid Bacteria Batch Cultures, *Journal of Fermentation and Bioengineering*, vol. 85, Nº6, pp.615-622, 1998.

[2] Billings, S.A., Jamaluddin , H.B. and Chen, S. Properties of Neural Networks with Applications to Modelling Non-Linear Dynamical Systems, *Int. J. Control*, vol. 55, Nº1, pp.193-224, 1992.

[3] Bishop, C. M, Neural Networks for Pattern Recognition, *Oxford*, New York, 1995.

[4] Dreyfus, G. and Idan, I. The Canonical Form of Nonlinear Discrete Time-Models, *Neural Computation*, vol 10(1), pp.133-164, 1998.

[5] Hernández E. and Arkun Y. Study of the Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamical Systems, *Computers chem Engng*, vol. 16, Nº4, pp.227-240, 1992.

[6] Hertz, J., Krogh, A., Palmer, R. Introduction to the Theory of Neural Computation, *Addison- Wesley*, 1990.

[7] Nerrand, O., Roussel-Ragot, P., Personnaz, L., Dreyfus, G., and Marcos, S. Neural Networks and Non- Linear Adaptive Filtering: Unifying Concepts and New Algorithms, *Neural Computation*, vol. 5, Nº2, pp.165-197, 1993.

[8] Oussar, Y. And Dreyfus, G. How to be a Gray Box: The Art of Dynamic a Semi-Physical Modelling, *Neural Networks*, vol. 14, 2001.

[9] Principe, J., Euliano, N., and Lefebvre W. Neural and Adaptive Systems: Fundamentals Through Simulations, *John Wiley & Sons, INC*, 1999.