INGENIERÍA ELECTRÓNICA

Herramienta para la identificación de procesos y simulación de redes neuronales mediante una agenda digital programable

Javier A. Minotta-Hurtado*, Eval B. Bacca-Cortés*§

** Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia*
§ e-mail: evbacca@univalle.edu.co

(Recibido: Marzo 2 de 2007 - Aceptado: Noviembre 9 de 2007)

Resumen

Este artículo describe el diseño, implementación y prueba de la herramienta denominada UV-SRNA-PDA (simulador de redes neuronales artificiales de la Universidad del Valle para una agenda digital programable), que está orientada a la simulación de redes neuronales artificiales y a la identificación de procesos industriales complejos. Esta aplicación trabaja sobre una agenda digital programable (PDA) *Palm T5* usando un sistema de adquisición de datos diseñado para tal fin. Dos tipos de redes neuronales fueron implementados: el perceptrón y el perceptrón multicapa (MLP) usando como algoritmos de aprendizaje los siguientes: propagación hacia atrás, gradiente descendente, velocidad de aprendizaje variable y momentum. Las pruebas se realizaron sobre plantas de primer y segundo orden (sólo esta última es reportada en este artículo), obteniendo su modelo neuronal y validando sus resultados en dos plataformas conocidas: MATLAB y UV-SRNA 2.0 (versión de la UV-SRNA-PDA para PC). Estas pruebas arrojaron un error de entrenamiento promedio de $5.62 \times 10^{-3} \pm 3.55 \times 10^{-4}$ y un error de validación promedio de $4.56 \times 10^{-3} \pm 5.95 \times 10^{-4}$. En ambos casos, los resultados son mejores o comparables con los de las otras herramientas de simulación. Sin embargo, el tiempo de entrenamiento típico en UV-SRNA-PDA fue de 900 s en comparación con los 3 s para MATLAB y 8 s para la UV-SRNA 2.0

Palabras clave: Redes neuronales artificiales, Perceptrón, MLP, GDA, PDA, Identificación de sistemas.

ELECTRONICS ENGINEERING

A tool for process identification and neural network simulation by means of a personal digital assistant

Abstract

This paper describes the design, implementation and testing of the software tool designated as UV-SRNA-PDA (Universidad del Valle's artificial neural network simulator for a personal digital assistant), which is oriented to artificial neural networks simulation and complex industrial processes identification. This application works on a PDA (personal digital assistant) *Palm T5* using a customized data acquisition system. Two kinds of artificial neural networks were implemented: perceptron and multi-layer perceptron (MLP) using as learning algorithms the following: backpropagation, descendent gradient, variable learning rate and momentum. To test the proposed system, first and second order systems were selected (only the latter is reported here), finding their neural models and validating their results with MATLAB and UV-SRNA 2.0 (PC version of UV-SRNA-PDA). These tests achieved an average training error of $5.62 \times 10^{-3} \pm 3.55 \times 10^{-4}$ and an average validating error of $4.56 \times 10^{-3} \pm 5.95 \times 10^{-4}$. In both cases, the results were better or comparable with those from other software simulation tools. However, the typical training time on the UV-SRNA-PDA was 900 s compared to 3 s in MATLAB and 8 s in UV-SRNA 2.0.

Keywords: Artificial neural networks, Perceptron, MLP, GDA, PDA, System identification.

1. Introducción

La potencia de cálculo de los dispositivos móviles como celulares y agendas electrónicas personales (PDA) ha mostrado un alto crecimiento. Actualmente, son muchas las aplicaciones con fines científicos que se han implementado. Establecer el modelo de un sistema o proceso, requiere encontrar la relación entre sus entradas y salidas. Esta relación puede estar definida por una ecuación matemática cerrada o no. La mayoría de los procesos industriales, presentan altas no linealidades, lo cual dificulta encontrar una expresión matemática cerrada que describa su comportamiento.

La inteligencia computacional brinda una herramienta para encontrar la relación entre las entradas y salidas de un proceso, cuyo modelo matemático es difícil de hallar. Las redes neuronales artificiales son tal herramienta; poseen muchos algoritmos de aprendizaje cuya diferencia está dada por la topología de red que emplean, la forma de actualizar los pesos de la red, el tipo de aprendizaje empleado (supervisado o no supervisado), como lo explican, entre otros, Narendra & Parthasarathy (1990) y Bose & Liang (1996). Sin embargo, este trabajo se enfoca en la implementación de dos topologías bien conocidas: el perceptrón y el perceptrón multicapa con algoritmo de aprendizaje de propagación hacia atrás (*backpropagation*). Este último tiene la característica de ser un aproximador universal de funciones, lo cual es muy útil para que se pueda establecer el modelo de un proceso desde un punto de vista sistémico (Heikkonen & Lampinen, 1999).

Las redes neuronales y los dispositivos móviles son una combinación que posee varias ventajas: permiten el desarrollo de aplicaciones con capacidad de generalización y aprendizaje (estas aplicaciones están ejecutadas por un sistema de cómputo pequeño y de bajo costo en relación con un computador portátil), y de software para equipos periféricos de uso específico y de dimensiones reducidas. En la industria maderera (Lampinen et al., 1998), la detección de los defectos en las maderas es un problema difícilmente modelable por técnicas convencionales, ya que depende de la forma, color y origen biológico de la madera; además, esta labor es realizada en campo y comúnmente en sitios de difícil acceso.

Q-OPT es una herramienta genérica para el modelado de procesos industriales (Heikkonen & Lampinen, 1999), que basándose en la capacidad de generalización y mapeado del algoritmo de propagación hacia atrás, permite modelar un sistema en términos de pequeños modelos neuronales. Esta herramienta requiere un computador personal (PC), y debido a su característica modular, la potencia de cálculo que requiere es alta.

La medida y estimación dinámica del peso es otra área interesante donde aplicar las redes neuronales; en este caso, no como parte de la medida en sí, sino con el fin de compensar las no linealidades. Un sistema típico es el de un ascensor de cuerda, el cual levanta un peso que debe ser constantemente medido para determinar la potencia del motor (Heikkonen & Lampinen, 1999).

En la industria de la seguridad, el sistema de detección SNOOPE de SAIC (*Science Applications International Corporation*) (Lisboa, 1992) es un sistema portable que emplea redes neuronales para la detección de explosivos en los equipajes de los aeropuertos. La combinación de portabilidad y redes neuronales es también clara en el área de la supervisión ambiental, en la cual la presencia de diferentes niveles de compuestos químicos y valores de las variables ambientales pueden determinar desde condiciones de seguridad favorables hasta un nivel de vida aceptable (DoD ARO/MURI, 1998). Otra aplicación industrial que requiere portabilidad y algoritmos basados en inteligencia computacional es la detección de anomalías en equipos industriales como columnas de destilación, calderas, molinos y rodillos industriales (Ploix & Dreyfus, 1996). En términos generales, se puede observar que la portabilidad, en combinación con una aplicación de inteligencia computacional, es muy útil en diferentes etapas como el modelado, control, mantenimiento y diagnóstico de procesos. La importancia del uso de agendas electrónicas como un elemento de cómputo, comunicación y adquisición de datos en estas etapas es bien

reconocida por las empresas de automatización e instrumentación. En comparación con un computador portátil, las agendas ofrecen una solución compacta (desde el punto de vista de servicios), portable, y de dimensiones y peso reducidos para ser ubicada en los centros de cableado.

Normalmente, el trabajo de modelado usando redes neuronales artificiales se realiza en un PC. Sin embargo, gracias al desarrollo de la aplicación UV-SRNA 2.0 (López, 1998) y el empleo de dispositivos móviles como las agendas electrónicas, el diseño o ajuste de controladores inteligentes puede ser llevado a cabo en el mismo lugar del proceso, lo cual es muy conveniente ya que justo es en ese instante y lugar que el proceso está influenciado por todas las variables ambientales y señales eléctricas, neumáticas o naturales propias del proceso, dando como resultado una relación entrada / salida mucho más aproximada a la entregada por procedimientos convencionales de identificación de procesos.

Figura 1. Sistema dinámico.

Este trabajo tiene como objetivo presentar la herramienta UV-SRNA-PDA, sus ventajas, limitaciones y su utilidad en la identificación de procesos industriales, como una de sus más inmediatas aplicaciones. De acuerdo con Proakis & Manolakis (1996), los sistemas dinámicos reales poseen la estructura que se muestra en la Figura 1, y pueden ser modelados usando la siguiente expresión:

$$y_p(k) = f(y_p(k-1), y_p(k-2),..., y_p(k-n), u(k-1), u(k-2),..., u(k-n)) \quad (1)$$

donde y_p es la salida del sistema a estimar, $y_p(k-1), \ldots, y_p(k-n)$ son las salidas retardadas del sistema y $u(k),\ldots, u(k-n)$ son las entradas al sistema presente y retardadas, respectivamente.

Se tiene entonces que el sistema depende exclusivamente de las entradas y salidas en instantes anteriores. De acuerdo a las características del sistema, se cuenta con un modelo general para la identificación haciendo uso de las redes neuronales artificiales (Bose & Liang, 1996), el cual se muestra en la Figura 2. La red neuronal recibe las entradas del sistema, realiza su estimación, y entonces el algoritmo de aprendizaje toma la salida real del sistema y la salida de la red neuronal para determinar el error y ajustar adecuadamente los parámetros de la red neuronal.

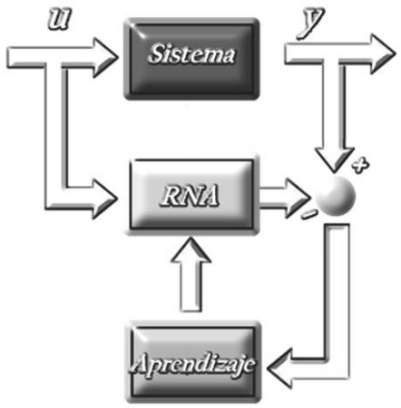

Figura 2. Modelo general de identificación de sistemas con redes neuronales artificiales (RNA).

Este modelo de identificación se implementó en el desarrollo de la herramienta UV-SRNA-PDA (Minotta & Correa, 2006), empleando J2ME (*Sun Microsystems*, 2000) como lenguaje de programación, el cual se compone de una configuración estándar para dispositivos inalámbricos que se conoce como CLDC (*Sun Microsystems*, 2001). CLDC es un conjunto de APIs (*Application Program Interfaces*) básicas para construir aplicaciones para dispositivos móviles. Con el fin de añadir nuevos requerimientos, surgió MIDP (*Sun Microsystems*, 2002). Mientras que CLDC es la configuración básica de J2ME, MIDP extiende CLDC y añade nuevos requerimientos y APIs para dispositivos que soportan MIDP.

Como se muestra en la Figura 3, la herramienta UV-SRNA-PDA realiza la identificación de sistemas industriales, obteniendo los datos que los representan mediante el sistema de adquisición de

datos diseñado e implementado para tal fin, el cual los envía a través de un protocolo de comunicación definido hacia la PDA. La aplicación UV-SRNA-PDA, aparte de ser una herramienta de simulación de redes neuronales artificiales, posee las herramientas necesarias para realizar tanto la identificación del sistema mediante un modelo neuronal configurable como su posterior validación.

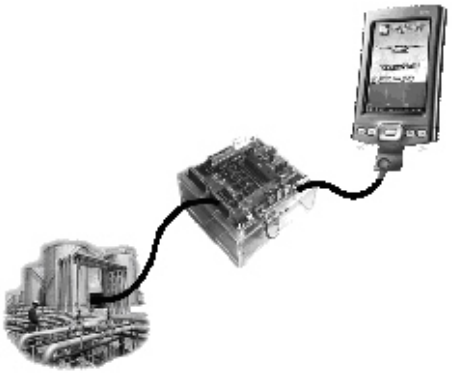

Figura 3. Diagrama general del sistema.

La herramienta UV-SRNA-PDA hace parte del trabajo de grado realizado por Minotta & Correa (2006) y actualmente se encuentra en proceso de registro ante el Ministerio del Interior (División de Derechos de Autor). Una vez este trámite esté completo, se podrá acceder a ella a través de la página del grupo *Percepción y Sistemas Inteligentes* de la Universidad del Valle.

En el presente artículo, se presentan inicialmente las características del sistema de adquisición de datos implementado; luego, se presenta la estructura de la aplicación UV-SRNA-PDA y la forma en la cual se realiza la identificación de sistemas; por último, se presentan los resultados de las pruebas realizadas en la identificación de sistemas de segundo orden.

2. Descripción del hardware y del software de la herramienta UV-SRNA-PDA

2.1 Sistema electrónico de adquisición de datos

Con el fin de permitir la identificación del proceso de interés, se diseñó e implementó un sistema de adquisición de datos de 8 entradas analógicas con una resolución de 10 bits cada una, cuatro entradas y salidas digitales aisladas cada una por opto-acoplador, una salida analógica con 10 bits de resolución, comunicación en serie a través de una interfaz RS-232 con velocidades de hasta 115 200 bps y tiempos de muestreo entre 1 ms y 250 ms. El sistema es controlado por un microcontrolador *ATMega32* (ATMEL, 2005), el cual trabaja con un micro-kernel de tiempo real de distribución gratuita llamado *FreeRTOS* (FreeRTOS, 2005).

En la Figura 4 se aprecia el diagrama de bloques del sistema de adquisición de datos, donde se observa la parte digital conformada por sus

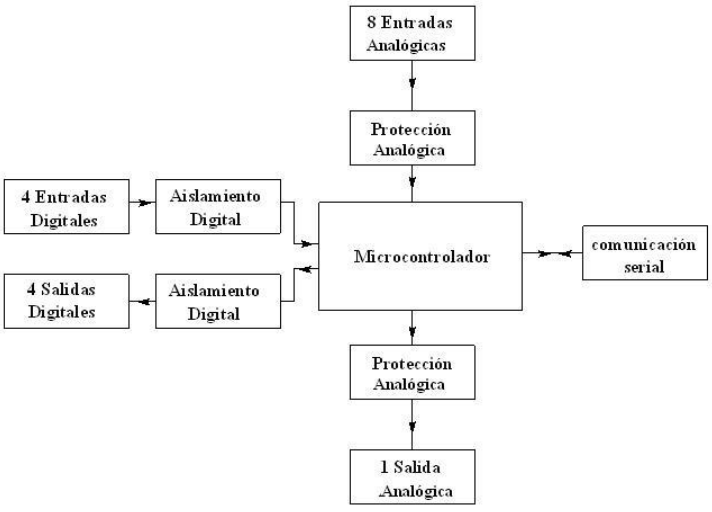

Figura 4. Diagrama de bloques del sistema de adquisición de datos.

entradas y salidas, la cual se comunica con el microcontrolador que realiza la gestión del sistema.

En la Figura 5 se muestra la estructura del sistema de adquisición de datos, y en ella se pueden ver las secciones que lo conforman, como son la etapa de potencia, la comunicación serial, la unidad de procesamiento, las entradas y salidas y los conectores de expansión.

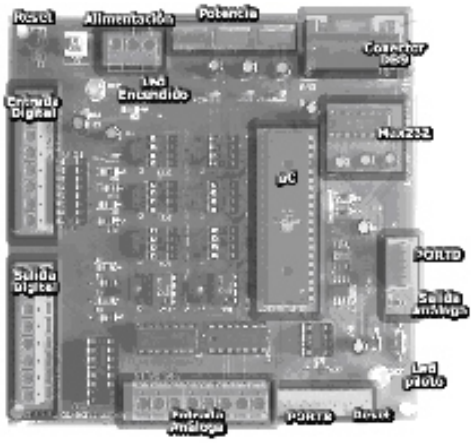

Figura 5. Estructura real del sistema de adquisición.

Para la identificación de sistemas es necesario tener una cantidad de señales tanto de entrada, como de salidas suficientes, con el fin de adquirir los datos necesarios para la buena representación del funcionamiento del sistema. En el procesamiento de todas estas señales se presenta el efecto de *cuello de botella* cuando se pretende comunicar los datos a través de una interfaz serial con restricciones de velocidad de comunicación y sobre todo manteniendo el tiempo de muestreo adecuado.

Este inconveniente se solucionó a través de dos herramientas: el empleo de un micro-kernel de tiempo real como el *FreeRTOS* que permite la ejecución de varios procesos en el microcontrolador y el empleo de una cola de datos como interfaz entre los módulos de adquisición y transmisión de los mismos a la PDA.

En la Figura 6 se encuentra el diagrama de bloques de las tareas implementadas en el software de control del sistema de adquisición, cuyos principales componentes son:

***Buffer* RXD y TXD.** Son estructuras de datos donde se almacenan los datos provenientes de la PDA y las tramas enviadas a ésta. El *buffer* TXD es una estructura de datos tipo COLA.

Tarea de ADC. De acuerdo a la configuración existente, realiza la adquisición de datos y los almacena en la cola.

Tarea de transmisión. De acuerdo a la configuración existente, transmite los datos almacenados en el *buffer* TXD.

Tarea de recepción y configuración. Lee las tramas válidas almacenadas en el *buffer* RXD, determina qué comando ha sido recibido y de acuerdo a éste configura el sistema o activa las tareas de adquisición y de transmisión que inicialmente estan pausadas.

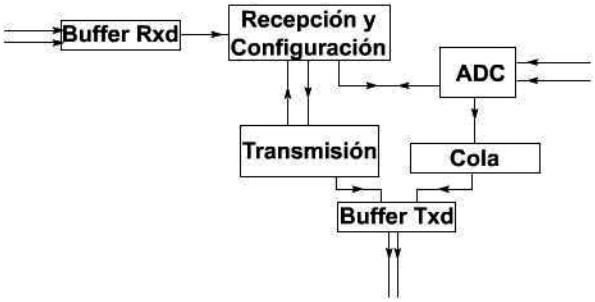

Figura 6. Diagrama de bloques de las tareas del sistema de adquisición.

2.2 Descripción de la herramienta UV-SRNA-PDA

UV-SRNA 2.0 es un software que recopila todas las características de procesamiento de un simulador de redes neuronales para uso académico. Esta herramienta incluye los siguientes algoritmos de aprendizaje: perceptrón, propagación hacia atrás, Hopfield, Kohonen, Levenberg-Marquardt, LVQ, RBF, ART y redes estocásticas. Sin embargo, esta herramienta no está diseñada para labores de identificación de procesos a través de cualquier sistema de adquisición de datos estándar. Esta tarea debe hacerse por aparte. UV-SRNA-PDA es un trabajo que no cubre todos estos algoritmos de aprendizaje, sólo los dos primeros, pero con un módulo de identificación basado en el algoritmo

de propagación hacia atrás con tasa de aprendizaje variable y momentum, los cuales fueron implementados siguiendo las pautas dadas por Haykin (1999). UV-SRNA, en su versión para PDA, posee las siguientes especificaciones (Minotta, 2006):

- interfaz gráfica amigable al usuario.
- soporte de los algoritmos de aprendizaje perceptrón y perceptrón multicapa con gradiente descendente, tasa de aprendizaje variable y momento.
- almacenamiento de archivos de patrones, validación y almacenamiento de redes neuronales compatibles con la herramienta UV-SRNA2.0.
- comunicación con el sistema de adquisición de datos a través de un protocolo previamente definido y a una velocidad de hasta 115.2 kbps.
- soporte de demos para las redes neuronales perceptrón y perceptrón multicapa.
- visualización de los errores de entrenamiento y validación.
- visualización de las señales adquiridas de procesos reales.
- módulo de identificación de procesos.

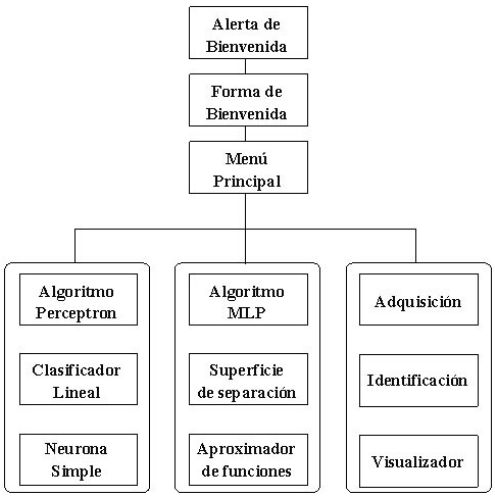

Figura 7. Diagrama de navegación general de la aplicación.

La herramienta UV-SRNA-PDA, cuenta con un esquema basado en formas o pantallas que se presentan según se requiera. Se hace uso de herramientas para la interfaz de usuario como botones y menús de texto desplegables que normalmente se encuentran disponibles a través de CLDC y MIDP para hacer la aplicación más versátil y facilitar su uso.

En la Figura 7 se muestra el diagrama de navegación de la aplicación y la bienvenida y presentación de la herramienta se muestra en la Figura 8. Posteriormente, se cuenta con un menú principal que posee tres módulos: el primero alberga las herramientas relacionadas con el perceptrón, como son el algoritmo propiamente dicho, los demos de clasificador lineal y neurona simple (Figura 9). El segundo módulo (Figura 10) posee las herramientas para el perceptrón multicapa (MLP) como son el algoritmo de aprendizaje en sí, los demos de superficies de separación, el aproximador universal de funciones, el módulo para la gestión de la adquisición de datos, el visualizador de señales y la identificación de procesos.

Figura 8. Presentación de la aplicación.

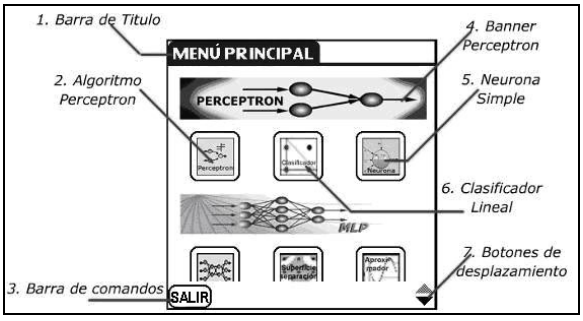

Figura 9. Módulo del perceptrón.

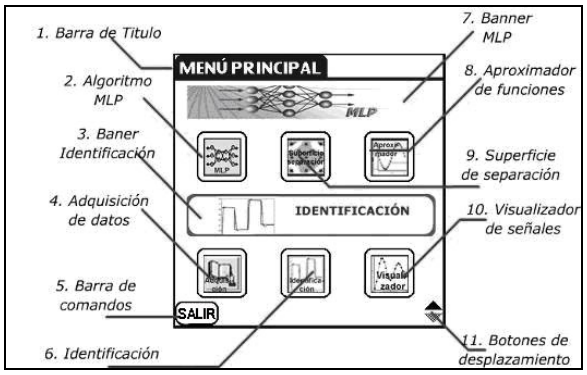

Figura 10. Módulo del perceptrón multicapa y la identificación de sistemas.

Los demos son parte de la aplicación original UV-SRNA para PC, cuyo objetivo pedagógico es la apropiación de los conocimientos básicos de la operación de una red neuronal. En UV-SRNA-PDA se tienen los dos demos descritos a continuación:

a) Demo para el perceptrón

Neurona artificial. En ella se asignan tres valores de entrada y sus pesos correspondientes, y se elige su función de activación. La estructura y la salida generada por la neurona se muestran en la Figura 11.

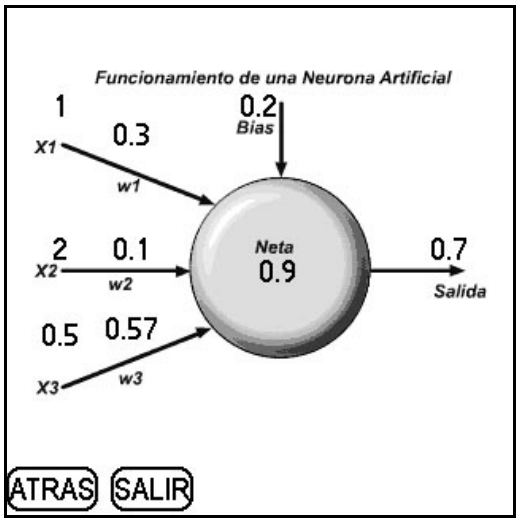

Figura 11. Demo del perceptrón para la neurona simple.

Clasificador lineal. Se obtiene una línea que parte en dos el plano y proporciona una solución a un problema determinado (que en el caso ilustrado en la Figura 12 es la solución de la AND lógica).

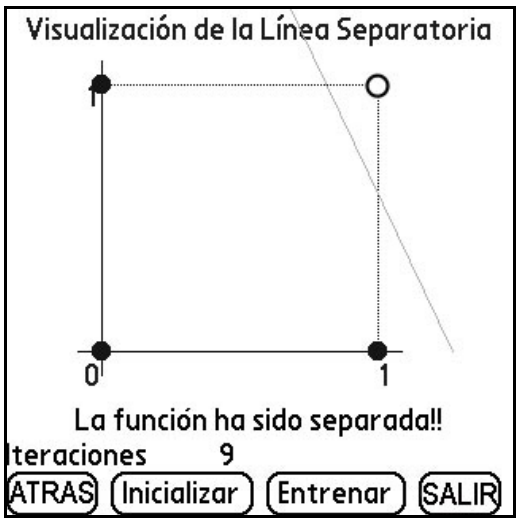

Figura 12. Demo del perceptrón para la separación lineal.

b) Demo del perceptrón multicapa

Superficie de separación. Se generan regiones en las cuales se realiza la separación de funciones más complejas que el perceptrón no puede solucionar (en el caso ilustrado en la Figura 13 es la función XOR).

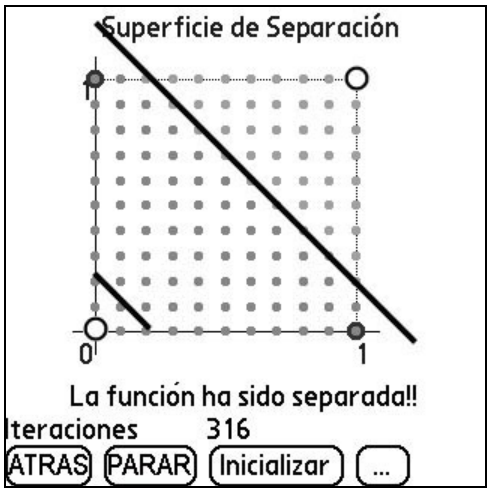

Figura 13. Demo del perceptrón multicapa para la superficie de separación.

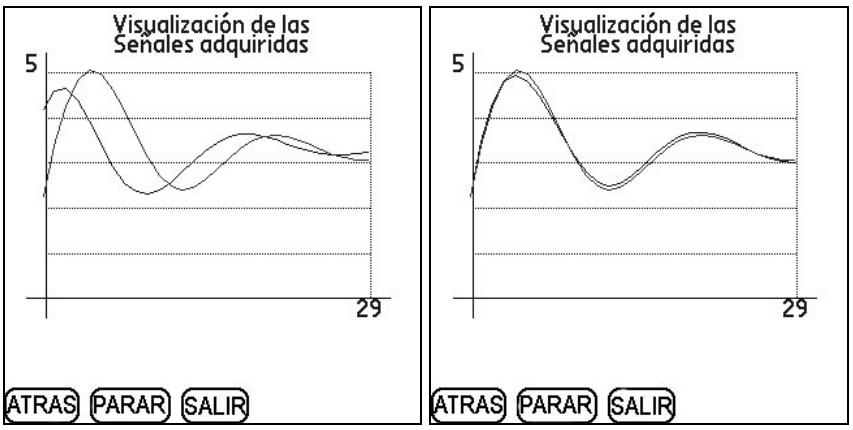

Figura 14. Demo del perceptrón multicapa para la superficie de separación.

Aproximador universal de funciones. La Figura 14 muestra el caso para una función del tipo $y(t) = Ae^{Kt}\text{sen}(\omega t)$ que es no lineal en t. Este es uno de los ejemplos para ilustrar que el perceptrón multicapa puede aprender el mapeo de cualquier función matemática.

2.3 Detalles de implementación de la herramienta sobre una PDA

La herramienta UV-SRNA-PDA fue implementada usando el lenguaje de programación JAVA, específicamente la API J2ME, debido a las ventajas con las que cuenta como portabilidad y manejo de equipos periféricos de uso específico encontrados en sistemas embebidos (como son las agendas electrónicas).

Con el fin de desarrollar aplicaciones sobre un dispositivo móvil (agendas o teléfonos celulares), existen tres opciones para la realización de las pruebas y depuraciones. La primera opción hace uso de un simulador (con el que cuenta la API de J2ME), el cual está conformado por una pantalla y comandos básicos. La segunda opción emplea un simulador distribuido por el fabricante del dispositivo móvil (en nuestro caso, el simulador del sistema operativo de PALM sería el adecuado), el cual muestra una interfaz de usuario muy similar a la encontrada en las agendas electrónicas de este fabricante, aunque su gran limitante es la falta de soporte para la emulación del hardware del dispositivo.

Finalmente, la última opción es la descarga de la aplicación directamente en el dispositivo. Esta opción requiere los siguientes pasos: la compilación de la aplicación, la creación del paquete que incluya la aplicación y sus recursos, la transformación de este paquete al lenguaje compatible con el dispositivo, y por último, la descarga de la aplicación al dispositivo a través de un cable de datos o un proceso de sincronización. En nuestro caso, y debido a que se empleó el puerto de comunicaciones de la agenda electrónica, fue necesario escoger la última opción. Sin embargo, si se desarrollan aplicaciones que no empleen hardware específico del dispositivo móvil, los simuladores son una buena opción.

La metodología de diseño de aplicaciones en dispositivos móviles está basada en su mayoría en establecer un esquema de navegación usando *pantallazos*. El cambio entre ellos está controlado por los eventos asociados a los botones, menús, listas de selección y demás elementos que componen una interfaz de usuario. J2ME soporta el uso de hilos, lo cual resulta especialmente útil para agilizar el trabajo de la aplicación, de la interfaz gráfica y de las interfaces con equipos periféricos de uso específico. En nuestro caso, fue necesario que la aplicación misma fuera un hilo (al albergar los eventos de los elementos de la interfaz de usuario), que el proceso de adquisición de datos fuera un hilo independiente, comunicándose a través de estructuras de datos con otro hilo dedicado a la graficación de datos provenientes

del puerto de comunicaciones, y finalmente, que cada proceso de cálculo relacionado con el entrenamiento de una red neuronal se ejecutara en un hilo independiente.

J2ME es una API en crecimiento, aunque no lo suficiente como para implementar herramientas científicas que exijan buena potencia de cálculo. Esta potencia de cálculo está representada en un compromiso entre el procesador y unas librerías matemáticas elaboradas eficientemente. La ventaja de abstracción que presenta J2ME en relación con el tipo de dispositivo móvil que se maneja, se convierte en un inconveniente aún no resuelto para la implementación de librerías matemáticas eficientes. Esto se debe a que su implementación requeriría un código nativo, lo que dada la variedad de dispositivos móviles es bastante dispendioso. En nuestro caso, la función exponencial y tangente sigmoidal fueron implementadas usando series y funciones recursivas, ya que éstas no se encontraban en la librería matemática de CLDC 1.1 y MIDP 2.0.

2.4 Proceso de identificación de sistemas

Para la adquisición de datos, se emplea el módulo de comunicaciones de la herramienta UV-SRNA-PDA, el cual cuenta con diversas herramientas para realizar la comunicación entre la PDA y un sistema externo a través de un protocolo previamente definido y que se encuentra detallado en el trabajo de Minotta & Correa (2006). En la Figura 15 se muestran las interfaces de usuario para la configuración de la comunicación como son: la velocidad de transferencia de datos, el tipo de señal de entrada, el formato de las entradas analógicas (simple o diferencial; para un formato simple, se tienen 8 entradas analógicas, y para un formato diferencial, se tienen 2 entradas diferenciales), el tipo de señal de salida, y el número de canales usados para la adquisición.

Luego, se configura el tiempo de adquisición y el tiempo de muestreo de la señal, con el fin de determinar el número de muestras a tomar. Por último, se procede a realizar la adquisición de los datos, contando con una interfaz en la cual se visualiza la evolución del proceso de adquisición (Figura 16).

Figura 15. Configuración del sistema de adquisición.

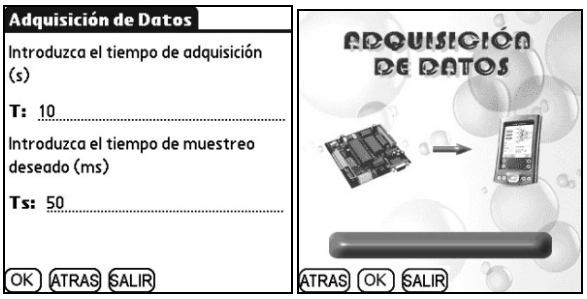

Figura 16. Configuración de los tiempos de adquisición y muestreo y de la adquisición de los datos.

Terminada la adquisición de los datos, se usa un menú para almacenar los patrones en un archivo con extensión **.pat* en la carpeta */UV-SRNA/*, la cual se encuentra en el sistema de archivos residente en la PDA (Figura 17).

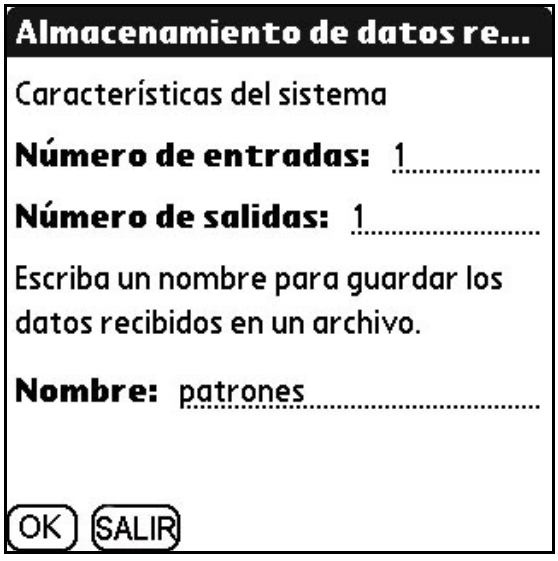

Figura 17. Almacenamiento de los datos adquiridos.

Se determina el número de entradas y salidas que se utilizaron, para así poder almacenar los datos con el formato adecuado. El número total de canales utilizados debe ser igual a la suma del número de entradas y el número de salidas seleccionadas. Antes de encontrar el modelo neuronal del proceso, se deben modificar los datos originales, de modo que los datos estén organizados de acuerdo al orden estimado del sistema para utilizar el mismo número de retrasos en las señales. En la Figura 18 se muestra la interfaz de la aplicación donde se realiza el cambio de formato de los datos para ser identificados. Este menú se encuentra disponible al elegir un archivo en la interfaz que visualiza el sistema de archivos del dispositivo y en el cual se pide el orden estimado del proceso.

Figura 18. Preparación de los datos para asignar un modelo neuronal al proceso.

En la Figura 19 se muestra la visualización de dos señales organizadas para realizar la identificación de un sistema de segundo orden. Se aprecia que existen tres señales graficadas: una corresponde a la entrada del sistema y las otras dos son las salidas, pero están atrasadas un instante de tiempo con relación a la otra. El proceso se repite dos veces, uno para adquirir los datos para el entrenamiento y otro para adquirir los datos para la validación.

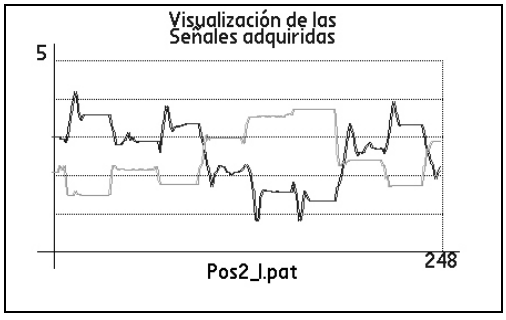

Figura 19. Visualización de las señales adquiridas.

Una vez con los datos en la PDA, se ejecuta el procedimiento de entrenamiento con el fin de encontrar el modelo neuronal del proceso. Para este fin, son necesarios los siguientes pasos:

Selección del archivo de patrones. Usando la interfaz para el manejo de archivos, se selecciona el archivo que contiene los patrones de entrenamiento deseados con extensión **.pat.* Normalmente, se muestra una alerta donde se especifican el número de patrones y el número de entradas y salidas, como se muestra en la Figura 20.

Figura 20. Características de los patrones de entrenamiento.

Configuración de parámetros de entrenamiento. Una interfaz gráfica dispone de campos de texto para ingresar los parámetros de configuración, que son: la tasa de aprendizaje α, el momentum β, el número máximo de iteraciones a realizar, el error deseado, el número de neuronas, las funciones de activación y el número de capas totales, como se observa en la Figura 21.

Figura 21. Parámetros de entrenamiento.

Entrenamiento. Se muestra una gráfica donde la abscisa corresponde al número de iteraciones y la ordenada al error cuadrático medio. Además, se visualiza el nombre del archivo que contiene los patrones de entrenamiento, el error cuadrático medio actual, el valor del parámetro α y el error cuadrático medio deseado (Figura 22) .

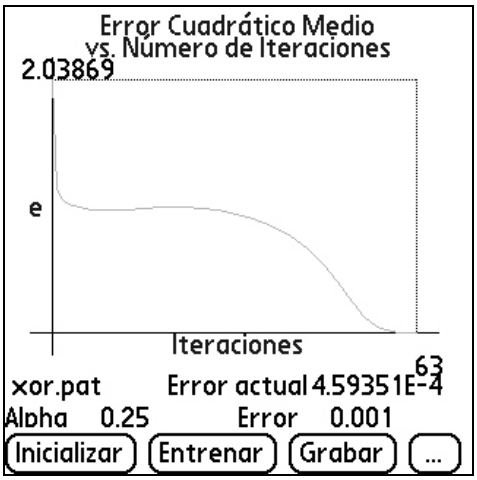

Figura 22. Evolución del error de entrenamiento.

Almacenamiento de la red neuronal. Al finalizar el entrenamiento, se almacena la red en el sistema de archivos del dispositivo como un archivo de extensión **.red*.

Validación de la red neuronal. La validación puede ser por teclado o por archivo. La validación por teclado es engorrosa para redes neuronales con un número amplio de entradas y, por lo general, se emplea para redes pequeñas. La validación por archivo es la que se emplea para la identificación de sistemas, para lo cual se necesitan los siguientes pasos:

a) Abrir la red neuronal y seleccionar el método de validación. Usando la interfaz gráfica, se selecciona el archivo que almacena la red neuronal cuya extensión debe ser **.red*. Esto muestra un mensaje de alerta con el número de neuronas en cada capa de la red neuronal. Luego se selecciona el método de validación (Figura 23).

b) Validación por archivo. Usando la interfaz de archivos, se selecciona el archivo que contiene los patrones de validación. Luego, se muestra una gráfica correspondiente a las salidas de la red tanto real como estimada, es decir, la salida que se encuentra en el archivo seleccionado y la salida que se genera después de realizar una propagación hacia adelante de la red MLP entrenada. Finalmente, se muestra el error cuadrático medio obtenido en el proceso de validación y la gráfica de evolución del error obtenido (Figura 24).

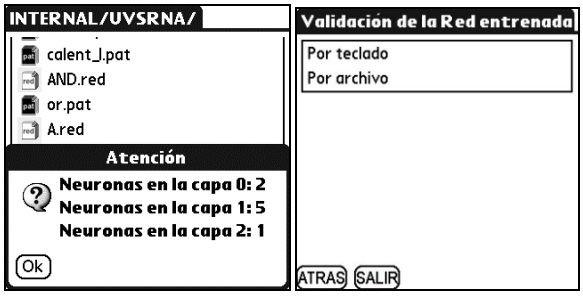

Figura 23. Validación de la red neuronal.

Figura 24. Visualización del error y gráfica de validación.

3. Pruebas y resultados

Con el fin de evaluar la validez de los resultados obtenidos con la herramienta UV-SRNA-PDA, se empleó un sistema de segundo orden con dinámica conocida. El objetivo de emplear un sistema con esta dinámica es comparar los resultados de la herramienta UV-SRNA-PDA con un sistema cuyo modelo sea bien conocido, de modo que permita determinar el adecuado funcionamiento de la herramienta, teniendo en cuenta un error cuadrático medio dado y un número máximo de iteraciones como criterios de parada. Los datos se obtuvieron con el sistema de adquisición mencionado, tanto para datos de entrenamiento

como de validación, y realizando la identificación con una red neuronal de tipo MLP cuya topología se muestra en la Figura 25.

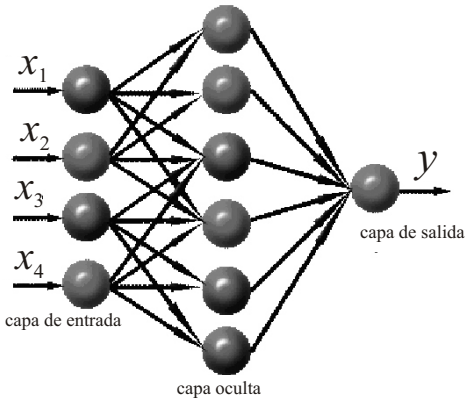

Figura 25. Topología de la red neuronal empleada.

La topología de la red neuronal empleada consta de los siguientes elementos:

- cuatro neuronas de entrada.
- cinco neuronas en la capa oculta.
- funciones de activación de tipo tangente sigmoidal.
- una sola neurona en la capa de salida con función de activación lineal.

El modelo neuronal obtenido se comparó con herramientas bien conocidas como MATLAB y UV-SRNA 2.0, con las cuales se realizó el mismo proceso de entrenamiento empleando la misma topología.

El sistema de segundo orden seleccionado fue el servomotor con control de posición *Feedback Instruments* (1992), que se muestra en la Figura 26.

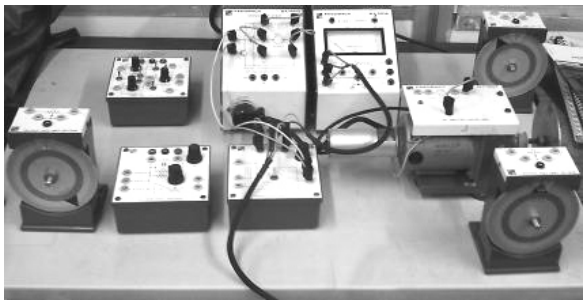

Figura 26. Servomotor con control de posición.

Este sistema responde a una dinámica de segundo orden cuando se encuentra realimentado y posee un disco con un potenciómetro (que cumple con la función de referencia manual midiendo la posición angular), un servomotor, una etapa de amplificación y unas etapas de control.

La salida del sistema está representada por otro disco similar al de la referencia manual, el cual se mueve e informa la posición angular indicada por la dinámica del proceso. La dinámica del sistema puede representarse matemáticamente por la expresión:

$$G(s) = \frac{0.2s + 1501.8}{2.6s^2 + 29.79s + 1520.61} \qquad (2)$$

Este sistema de segundo orden posee un tiempo de establecimiento aproximadamente igual a 1.9 s. Empleando el procedimiento de adquisición de datos para la identificación del sistema, descrito en secciones anteriores, con un tiempo de muestreo de 60 ms, se obtuvieron dos conjuntos de datos: el primero de los cuales se usó para el entrenamiento y el segundo para la validación del modelo neuronal. Estos conjuntos de datos poseen un total de 248 muestras cada uno, correspondiendo a la mayor cantidad de dinámicas del sistema, como se muestra en las Figuras 27 y 28. El orden de regresión de la red neuronal está definido según el orden estimado del sistema (Norgaard et al., 2003), que en el caso considerado es igual a 2.

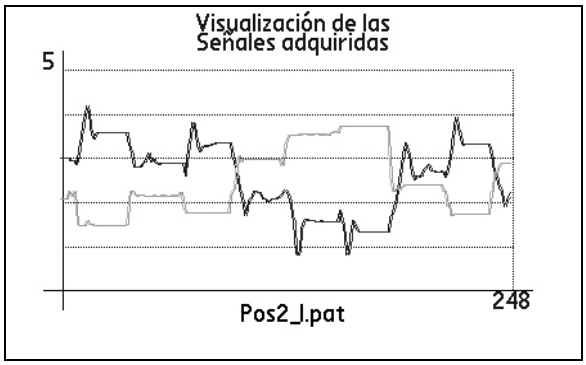

Figura 27. Datos de entrenamiento del sistema de posición.

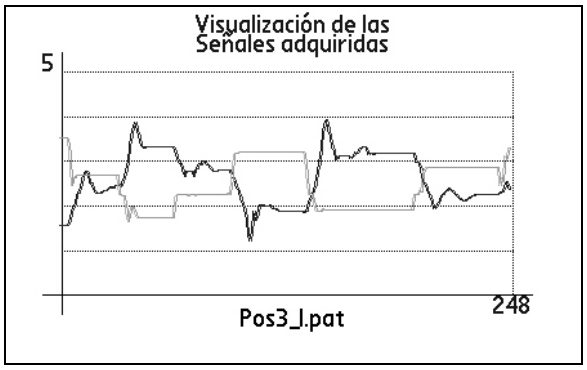

Figura 28. Datos para validación del modelo neuronal.

La identificación del sistema se obtuvo realizando el entrenamiento de la red de tipo MLP 10 veces en las tres herramientas de análisis (UV-SRNA-PDA, UV-SRNA para *Windows* y MATLAB). La condición de parada estuvo dada por un error cuadrático medio de 0.0045 o 500 iteraciones como máximo. Para cada red entrenada se realizó su correspondiente validación, con datos diferentes a los de entrenamiento, como se puede constatar en la Figura 28. En las Figuras 29, 30 y 31 se presentan la evolución del error de entrenamiento y la validación de uno de los modelos neuronales obtenidos para cada herramienta.

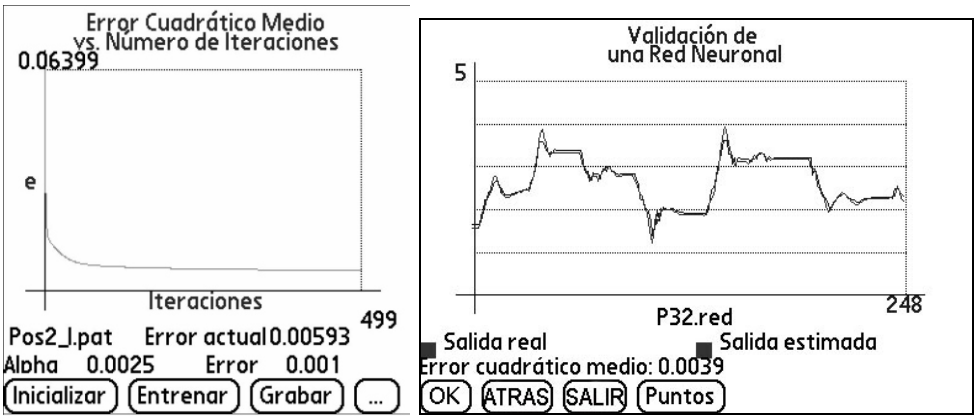

Figura 29. Evolución del error de entrenamiento y validación de un modelo neuronal entrenado en UV-SRNA-PDA.

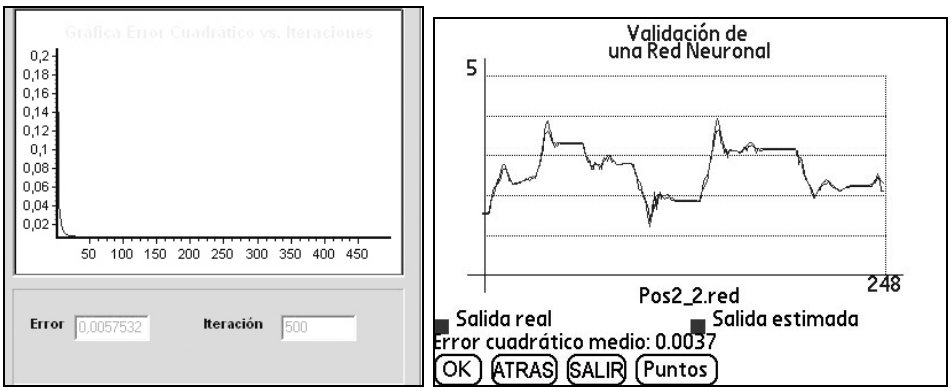

Figura 30. Evolución del error de entrenamiento y validación de un modelo neuronal entrenado en UV-SRNA 2.0

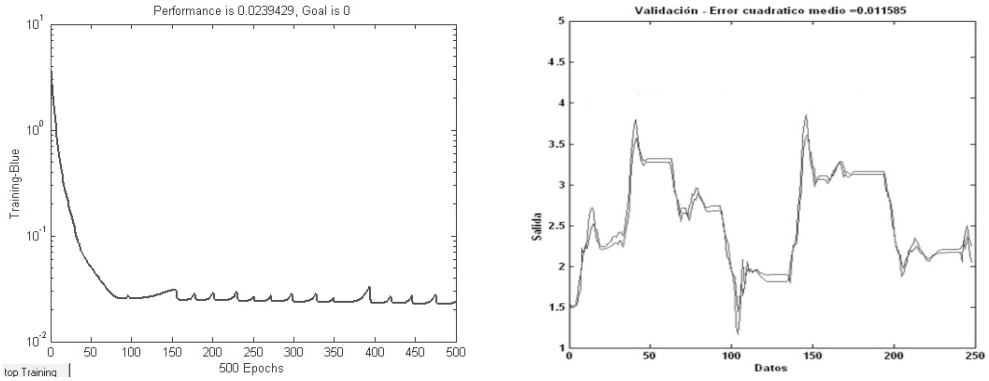

Figura 31. Evolución del error de entrenamiento y validación de un modelo neuronal entrenado en MATLAB.

4. Discusión de resultados

En la Figura 32 se presentan los resultados correspondientes al entrenamiento para la identificación del sistema de segundo orden. En esa figura, se observan los 10 entrenamientos para cada una de las herramientas en las abscisas y el error de entrenamiento final en las ordenadas.

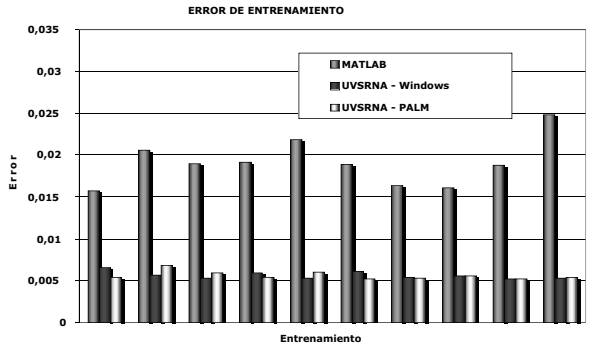

Figura 32. Error de entrenamiento en las pruebas.

Los diferentes errores cuadráticos medios que se muestran en la Figura 32 permiten afirmar que la herramienta UV-SRNA-PDA obtiene un modelo del sistema cuyas especificaciones de error cuadrático medio son comparables con los modelos arrojados por UV-SRNA 2.0 y MATLAB. Se obtuvo un error de entrenamiento promedio de $5.62 \times 10^{-3} \pm 3.55 \times 10^{-4}$ para la herramienta UV-SRNA-PDA, de $5.64 \times 10^{-3} \pm 3.20 \times 10^{-4}$ para la herramienta UV-SRNA para *Windows* y de $1.91 \times 10^{-2} \pm 2.00 \times 10^{-3}$ para MATLAB.

Se puede notar que para esta muestra de pruebas, el error cuadrático medio de las herramientas UV-SRNA-PDA y UV-SRNA 2.0 es menor que el de MATLAB.

Aunque el propósito de este artículo no es comparar la eficiencia de los algoritmos implementados sino dar a conocer una herramienta de inteligencia computacional portable, se puede afirmar que esta diferencia se debe al modo en que se procesan los datos en estos dos tipos de herramientas: en las primeras, el error cuadrático medio es calculado cada vez que un patrón es presentado a la red y en MATLAB (según la documentación del producto), se calcula por lotes de datos.

Como se discute más adelante, esto hace que el tiempo de entrenamiento sea muchísimo menor en MATLAB que en las otras herramientas. Sin embargo, es probable que las actualizaciones de los pesos realizadas en MATLAB hagan que los saltos en la superficie de error sean más grandes que en las otras herramientas, lo cual ocasiona que mínimos locales útiles para la identificación no sean considerados.

En la Figura 33 se presenta el resultado de las validaciones del sistema de segundo orden. En esa figura, se observa que tanto UV-SRNA-PDA como UV-SRNA para *Windows* presentan errores de validación similares y que MATLAB presenta un error de validación más alto.

Se obtuvo un error de validación promedio de $4.20 \times 10^{-3} \pm 4.62 \times 10^{-4}$ para la herramienta UV-SRNA para *Windows*, de $4.56 \times 10^{-3} \pm 5.95 \times 10^{-4}$ para la herramienta UV-SRNA-PDA y de $1.52 \times 10^{-2} \pm 2.29 \times 10^{-3}$ para MATLAB. Los promedios del error cuadrático medio tanto para entrenamiento como para validación, se reportan con un intervalo de confianza del 95%, teniendo en cuenta una distribución de probabilidad *t-Student*.

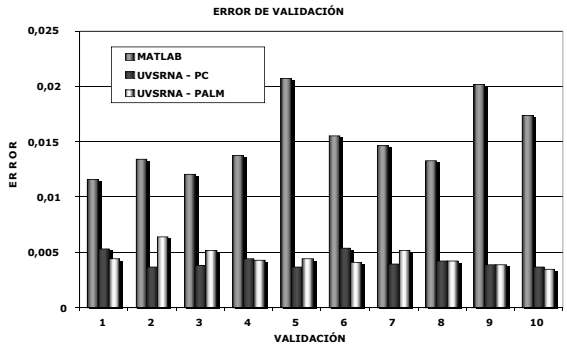

Figura 33. Error de validación de las pruebas.

MATLAB y UV-SRNA para *Windows* se ejecutaron en un computador con procesador *AMD Athlon 64* de 3 GHz, con 512 MB de memoria RAM DDR de bus de 400 MB / s y disco duro de 7200 rpm. UV-SRNA-PDA se ejecutó en una *PALM Tungsten T5* (Palm Inc., 2007), cuyo procesador es de 415 MHz y cuenta con 256 MB de memoria. Los tiempos de entrenamiento del modelo neuronal para la identificación del sistema de segundo orden fueron los siguientes: 3 s con MATLAB, 4s con UV-SRNA para *Windows* y 900 s con UV-SRNA para PALM.

Los resultados de entrenamiento y validación indican que la herramienta UV-SRNA-PDA que implementa un algoritmo de propagación hacia atrás con velocidad de aprendizaje variable y momentum, logra obtener un modelo del sistema de segundo orden comparable al obtenido con una herramienta bien conocida y usualmente tenida en cuenta como referente como es MATLAB. La diferencia en los tiempos de entrenamiento de las herramientas se debe a dos aspectos. El primer aspecto se refiere a la potencia del procesador en el cual se llevaron a cabo los cálculos, lo cual es un problema más tecnológico que algorítmico.

Un *AMD Athlon 64* posee 8400 MIPS mientras un procesador de la familia PXA tiene hasta 800 MIPS. El segundo aspecto tiene que ver con la madurez del desarrollo de las especificaciones CLDC 1.1 y MIDP2.0 con las cuales se implementó la herramienta, ya que éstas no poseen una librería matemática completa: funciones como la exponencial y la tangente sigmoidal fueron implementadas a nivel de aplicación al no existir su versión en las especificaciones mencionadas. Este último aspecto tiene como consecuencia que los cálculos de la presentación de los patrones a la red (*FeedForward*) y del error cuadrático medio sean más demorados en la herramienta UV-SRNA-PDA.

5. Conclusiones

En este trabajo se ha presentado una herramienta para la simulación de redes neuronales artificiales y la identificación de procesos complejos, a través de modelos neuronales entrenados usando el algoritmo de propagación hacia atrás con α variable y momentum [conocido como aproximador universal de funciones (Freeman & Skapura, 1993)].

UV-SRNA-PDA y su sistema de adquisición de datos son una herramienta portable que permite obtener un modelo de un proceso real directamente en el campo, lo cual posee la ventaja de extraer el modelo del proceso en su ambiente real de operación, bajo la influencia de factores externos como temperatura, ruido y carga.

Actualmente, es de especial interés la automatización de procesos industriales a través del control de variables tales como temperatura en calderas y fluidos, nivel de tanques de almacenamiento, presión, rotación de cilindros, velocidad en ejes de motores de corriente directa o alterna, para el diagnóstico de procesos y en seguridad ambiental y labores de inspección, entre otros. Estos procesos tienen en cuenta diversas variables, cuya interacción (lineal o no lineal) produce una respuesta. Encontrar el modelo matemático de esta interacción no siempre es posible y por ende una herramienta portable como la UV-SRNA-PDA permite la generación de un modelo neuronal del proceso.

UV-SRNA-PDA también es, debido a su portabilidad, altamente aplicable en la educación para la comprensión de la teoría de operación de las redes neuronales, lo cual la convierte en una buena herramienta de aprendizaje.

Los resultados obtenidos en las fases de entrenamiento (error cuadrático medio de $5.62 \times 10^{-3} \pm 3.55 \times 10^{-4}$) y de validación (error cuadrático medio de $4.56 \times 10^{-3} \pm 5.95 \times 10^{-4}$) muestran que UV-SRNA-PDA es una herramienta que permite identificar un proceso con buena exactitud, sin conocer de antemano el modelo matemático correspondiente, a pesar de no disponer de una adecuada implementación de las funciones matemáticas, sino de funciones iterativas implementadas en la capa de aplicación de J2ME.

Estos resultados permiten afirmar que la herramienta UV-SRNA-PDA es propicia para establecer el modelo de un sistema, cuya aplicación va desde la identificación de defectos en materias primas, diagnóstico de procesos industriales con el fin de determinar su buen o mal funcionamiento, seguridad ambiental, recolección de información financiera para la estimación de ganancias o pérdidas, inspección de equipos eléctricos, electrónicos, químicos o estructurales para la obtención de su buen funcionamiento, etc.

Aunque las condiciones de portabilidad y potencia de cálculo poseen un muy buen compromiso en los computadores portátiles, estos equipos son pesados, costosos y de dimensiones mayores que una agenda electrónica (*handheld*), características importantes en ambientes industriales donde el espacio en los gabinetes de control es reducido, los soportes no están diseñados para grandes pesos y las condiciones de disipación de potencia son restringidas, por lo cual, se prefiere el uso de las agendas electrónicas.

Actualmente, J2ME y sus especificaciones CLDC 1.1 y MIDP 2.0 están diseñadas para aplicaciones con baja exigencia de cálculo pero con un muy buen manejo de la interfaz de usuario, lo cual explica su empleo en el desarrollo de juegos.

Finalmente, es de esperar que con el incremento de la potencia de cálculo de los dispositivos móviles y la creciente madurez de las especificaciones CLDC 1.1 y MIDP 2.0, la programación de aplicaciones de uso científico como UV-SRNA-PDA permita en el futuro expandir su funcionalidad, al incrementar el número de topologías y algoritmos de aprendizaje, y permitiendo la implementación de un módulo de control inteligente de procesos basado en redes neuronales artificiales y un módulo de control basado en lógica difusa, entre otros.

6. Referencias bibliográficas

ATMEL. (2005). *8-bit AVR microcontroller with 32K bytes in-system programmable flash (ATmega32, Atmega32L).* http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf

Bose, N. K., & Liang, P. (1996). *Neural networks fundamentals with graphs, algorithms, and applications.* New York: McGraw-Hill.

DoD ARO/MURI (Department of Defense Multidisciplinary University Research Initiative). (1998). *The nose project.* http://www.wag.caltech.edu/muri/

Feedback Instruments Limited. (1992). *Operation manual of PCT-14 and PCT-10*.

Freeman, J. A., & Skapura, D. M. (1993). *Redes neuronales: algoritmos, aplicaciones y técnicas de programación*. Delaware: Addison-Wesley.

FreeRTOS. (2005). *RTOS implementation for AVR microcontrollers*. http://www.freertos.org.

Haykin, S. (1999). *Neural networks a comprehensive foundation*. Second edition. New York: Prentice Hall Inc.

Heikkonen, J., & Lampinen, J. (1999). *Building industrial applications with neural networks.* In Proceedings of the European Symposium on Intelligent Techniques, ESIT99, Chania, Greece. http://www.erudit.de/erudit/events/esit99/12642_P.pdf

Lampinen, J., Smolander, S., & Korhonen, M. (1998). Wood surface inspection system based on generic visual features. In: F. Fogelman-Soulié & P. Gallinari (editors), *Industrial applications on neural networks*, p. 35-42. Singapore: World Scientific Publishing.

Lisboa, P. J. G. (editor). (1992). *Current applications of neural networks*. London: Chapman & Hall.

López, J. (1998). *Herramienta de simulación de redes neuronales artificiales de la Universidad del Valle*. Tesis de maestría, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia.

Minotta, J. A. (2006). *Manual de usuario de la herramienta UV-SRNA para PDA*. Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia.

Minotta, J. A., & Correa, A. (2006). *Prototipo para la implementación de la herramienta UV-SRNA sobre una PDA*. Tesis de pregrado, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. I*EEE Transactions on Neural Networks* 1 (1), 4-27.

Norgaard, M., Ravn, O., Poulsen, N. K., & Hansen, L. K. (2003). *Neural networks for modelling and control of dynamic systems: a practitioner's handbook.* London: Springer-Verlag.

Palm Inc. (2007). *Palm products: Palm T | X handheld*.
http://www.palm.com/us/products/handhelds/tx/

Ploix, J. L., & Dreyfus, G. (1996). Knowledge-based neural modeling: principles and industrial applications. In: F. Fogelman-Soulié & P. Gallinari (editors), *Industrial applications on neural networks*. Singapore: World Scientific Publishing.
http://www.neurones.espci.fr/Articles_PS/ploix.pdf

Proakis, J. G., & Manolakis, D. K. (1996). *Digital signal processing, principles, algorithms, and applications.*Third edition. New Jersey: Prentice Hall, Inc.

Sun Microsystems. (2000). *The JAVA ME platform*. Sun Devoloper Network (SDN).
http://java.sun.com/javame/index.jsp

Sun Microsystems. (2001). *Connected limited DeviceConfiguation 1.1 (CLDC)*. Sun Devoloper Network (SDN).
http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html

Sun MicroSystems. (2002). *Mobile Information Device Profile (MIDP) 2.0* .
http://java.sun.com/products/midp/