INFORMÁTICA

SISTEMA PARA EL ENTRENAMIENTO PARALELO DE REDES NEURONALES DE PROPAGACIÓN HACIA ATRÁS

Resumen / *Abstract*

El objetivo principal de este trabajo es la exploración del uso de la programación paralela en la instrumentación de redes neuronales, específicamente este artículo trata del uso de tal procesamiento para acelerar la fase de entrenamiento de una red de propagación hacia atrás (BPN). Se describe la implementación y los resultados obtenidos con la red neuronal antes mencionada. El sistema está realizado de manera tal, que el usuario pueda ejecutarlo eficientemente con un amplio rango de aplicaciones neuronales. Para probar el algoritmo paralelo implementado se utilizó una aplicación neuronal real y los resultados obtenidos muestran lo beneficioso de usar el entrenamiento paralelo.

The main objective of the work presented here is to speed up artificial neural network training using parallel processing. Backpropagation neural network was selected for this research, since it has attracted most interest among neural network researchers.This article describes the implementations and experiments undertaken for selected model of neural network. The parallel program can execute efficiently for a large range of neural applications. One real neural application has been used in this work to test the implemented parallel algorithm on workstation cluster. The results show that it is beneficial to use parallel processing for neural network training, obtaining better performance on a large system than on a small one.

Palabras clave / *Key words*

Redes neuronales, propagación hacia atrás, BPN, entrenamiento paralelo, programación paralela, inteligencia artificial, entrenamiento acelerado

Neural networks, backpropagation, BPN, parallel training, parallel programming, artificial intelligence, fast training

***Ernesto Elejalde Sierra**, Ingeniero Informático, Datacimex, Corporación CIMEX, Ciudad de La Habana, Cuba*
e-mail:e_elejal@yahoo.es

***George Navarro Gómez**, Ingeniero Informático, Centro de Elaboración de Boyeros, CIMEX, Ciudad de La Habana, Cuba*
e-mail:georgen@cimex.com.cu

***Alejandro Rosete Suárez,** Ingeniero Informático, Doctor en Ciencias Técnicas, Profesor Auxiliar, Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:rosete@ceis.cujae.edu.cu

***Exiquio C. Leyva Pérez,** Ingeniero Químico, Doctor en Ciencias Técnicas, Profesor Titular, Centro de Estudios de Ingeniería de Sistemas, CEIS, Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:exiquio@ceis.cujae.edu.cu

Recibido: octubre del 2005
Aprobado: diciembre del 2005

INTRODUCCIÓN

El desarrollo actual de la ciencia se dirige al estudio de las capacidades humanas como una fuente de nuevas ideas para el diseño de las máquinas. Así, la inteligencia artificial es un intento por descubrir y describir aspectos de la inteligencia humana que puedan ser simulados mediante máquinas.

Las redes neuronales son una forma de emular otra de las características propias de los humanos: la capacidad de memorizar y asociar hechos. Si se examina con atención aquellos problemas que no pueden expresarse a través de un algoritmo se observa que todos tienen una característica común: la **experiencia**. Así, parece claro que una forma de aproximarse a este tipo de problemas consiste en la construcción de sistemas que sean capaces de reproducir la característica humana de acudir a la experiencia acumulada.[1]

Todo parece indicar que el cerebro funciona de forma paralela y distribuida, siendo las neuronas los elementos computacionales. Estas son relativamente simples y la complejidad del funcionamiento de este órgano parece que radica en los esquemas paralelos de interconexión. Sin embargo, debido a que las redes neuronales se encuentran aún en una fase de modelación teórica, la mayoría de las

redes hasta ahora han sido simuladas sobre máquinas secuenciales (arquitectura de Von Neuman) y no sobre máquinas paralelas,[2] que son las que realmente poseen la potencia de cálculo necesaria para realizar en un tiempo factible las operaciones requeridas, principalmente en la fase de entrenamiento.

Dada las múltiples áreas de aplicación de las redes neuronales, el sistema implementado disminuye el tiempo de entrenamiento de estas redes, lo cual ayudaría a la utilización de este enfoque en las diferentes ramas de la economía del país.

Existen hoy en el mundo varios sistemas de gestión empresarial que presentan un elevado precio en el mercado, debido a que la gran mayoría de los procesos son altamente optimizados y mejorados mediante la incorporación de modelos no lineales y técnicas neuronales. El presente trabajo pretende aumentar, aún más, estas ventajas con la utilización de la programación paralela.

PROGRAMACIÓN PARALELA

Generalidades

Una computadora paralela es un grupo de procesadores que pueden trabajar juntos para resolver cooperadamente un problema de cálculo. Esta definición es bastante amplia como para incluir las supercomputadoras paralelas que tienen centenares o miles de procesadores, las redes de estaciones de trabajo (*workstations*), estaciones multiprocesadoras, y los sistemas empotrados.[3]

El rendimiento de una computadora depende directamente del tiempo requerido para realizar las operaciones básicas y la cantidad de dichas operaciones que se puedan llevar a cabo de forma concurrente. El tiempo de realización de una operación básica está limitado por los ciclos de reloj del procesador. Los tiempos de ciclo de reloj están disminuyendo con el desarrollo tecnológico, pero parecen estar acercándose a límites físicos como la velocidad de la luz;[4] es decir, que no se puede esperar por procesadores más rápidos cuando se quiere obtener grandes aumentos en la potencia de cálculo.

Estudio del rendimiento de los algoritmos paralelos

Para saber si se ha alcanzado o no un incremento de las prestaciones es importante estudiar cómo se debe evaluar un algoritmo paralelo concreto, tanto desde el punto de vista absoluto (¿cuán rápido es el algoritmo?) como desde el punto de vista relativo (¿cuán rápido es el algoritmo comparado con otro estándar?). Se mostrarán solo algunos de los parámetros más importantes.

Parámetros absolutos

• *Tiempo secuencial de un algoritmo*: Tiempo que tarda en ejecutarse el mejor algoritmo concebido para un solo procesador (lo que no significa ejecutar el algoritmo paralelo considerando un solo procesador).

• *Tiempo de ejecución con p procesadores*: Tiempo que tarda en ejecutarse el algoritmo utilizando *p* procesadores.

• *Tiempo de comunicación de un algoritmo paralelo ejecutado con p procesadores*: Tiempo que tarda el sistema multiprocesador en realizar transferencias de datos.

Parámetros relativos

• *Grado de paralelismo de un algoritmo*: Se define como el número de operaciones que se pueden hacer en paralelo en dicho algoritmo.

• *Incremento de velocidad (Speed-Up):* S_p. indica la ganancia de velocidad de un determinado algoritmo paralelo, ejecutado sobre *p* procesadores, cuando se compara con el mejor algoritmo secuencial. Se calcula como: $Sp = T_1 / T_p$, siendo T_1 el tiempo del mejor algoritmo secuencial que resuelve el problema y T_p el tiempo del algoritmo paralelo ejecutado sobre *p* procesadores.

• *Eficiencia*: E_p, mide el grado de utilización de un sistema mutiprocesador. Se calcula de la forma siguiente: $Ep = Sp / p$. Suele expresarse en porcentaje.

PARALELIZACIÓN DE REDES NEURONALES ARTIFICIALES

Todos los algoritmos de redes neuronales artificiales (RNA) presentan una fase de aprendizaje en la cual esta es entrenada con un juego de ejemplos de un problema determinado.[5] La red entrenada se utiliza entonces en un ambiente real para resolver instancias del problema que no estaban contenidas en los ejemplos. La fase de aprendizaje normalmente toma gran cantidad de tiempo computacional, a menos que sean problemas sencillos. Para problemas prácticos, dónde existen muchos datos, los tiempos de entrenamiento están en el orden de los días y las semanas, al ser ejecutada la aplicación en una sola computadora. este ha sido el obstáculo principal a la hora de implementar y utilizar las redes neuronales en aplicaciones reales y ha impedido también que su grado de aceptación sea aún más elevado.[6]

El problema del elevado tiempo de entrenamiento puede ser resuelto mediante la creación de algoritmos de aprendizaje más eficientes o implementando algoritmos para arquitecturas de computadoras paralelas.

Se han propuesto e implementado varias redes neuronales en computadoras paralelas, tanto en computadoras de propósito general como de propósito específico. Algunos ejemplos son:

• Simulador de redes neuronales de la Universidad de Stuttgart (Stuttgart Neural Network Simulator, SNNS), en el que una de las formas de asignación usadas consiste en asignar un elemento de procesamiento (EP) a cada segmento vertical de la red alimentada hacia delante, lo que incluye una neurona de cada capa.

• Simulación de una red BPN sobre la CM-1 (64k procesadores), por Belloch y Rosenberg. Utilizaron un procesador para simular cada neurona en la red. Un procesador para cada peso de salida y otro para cada peso de entrada.[7]

SOLUCIÓN Y ANÁLISIS DE RESULTADOS

En esta sección se describen los resultados obtenidos con la implementación paralela de una red neuronal de propagación

hacia atrás (BPN). Con el objetivo de probar el funcionamiento de dicha red se decidió entrenarla en el reconocimiento optico de caracteres (Optical Character Recognition, OCR). El objetivo de este trabajo no es desarrollar todo el software del OCR, solo entrenar la red que sería necesaria para una aplicación de esta índole.

Software desarrollado

El software implementado permite el entrenamiento de redes neuronales de propagación hacia atrás (BPN). La interacción de la aplicación con el usuario se hace mediante un fichero de entrada y uno de salida que facilita el uso en un amplio rango de aplicaciones reales.

Descripción de la implementación paralela de la red BPN

La figura 1 muestra cómo un cluster de computadoras puede ser usado para ejecutar una simulación paralela de una red BPN. Una copia del programa se ejecuta en la computadora que desempeña el papel de maestro y una copia en cada una de las computadoras esclavas. La colección de casos de entrenamiento es dividida en partes iguales entre las copias esclavas, a esta forma de división del problema se le denomina paralelismo de datos de entrenamiento.

El programa maestro genera aleatoriamente la colección inicial de pesos para la red, estos son difundidos a todos los nodos esclavos. Cada programa esclavo inicializa su red con estos valores y se ejecuta con los casos de entrenamiento que le fueron asignados anteriormente, acumulando los cambios en los pesos. Los programas esclavos envían al programa maestro los cambios acumulados, este los adiciona y aplica el ajuste a la colección inicial. Luego envía los nuevos pesos a todos los nodos esclavos para sus próximos ciclos de ejecución con los mismos casos de entrenamiento.

El entrenamiento termina cuando el error, calculado por el maestro, alcanza un valor previamente definido por el usuario de la red. Este proceso es totalmente equivalente al aprendizaje por lotes en un solo procesador.

Aplicación de prueba

El documento original a ser analizado se escanea y se salva en la computadora. El software de OCR divide la imagen en subimágenes, de forma tal, que cada una contenga una sola letra. Las subimágenes se traducen entonces de un formato de imagen a un formato binario donde cada 0 y 1 representa un píxel individual de la subimagen.

La información binaria se le pasa como entrada a una red neuronal que se ha entrenado para realizar la asociación entre los datos de la imagen de la letra y un valor numérico que corresponde con esta. La salida de la red neuronal se traduce ahora en un texto ASCII y se guarda en un fichero. El proceso completo se representa en la figura 2.

Resultados obtenidos con la BPN

Los resultados fueron obtenidos con una red BPN que presenta las siguientes características:

Casos de entrenamiento:	27, 54, 81
Velocidad de aprendizaje:	0,05
Neuronas capa entrada:	64
Factor momentum:	0,9
Neuronas capa oculta:	50
Desviación BIAS:	1
Neuronas capa salida:	27
Pesos iniciales (aleatorios)	entre -0,5 y 0,5
Función transferencial:	Sigmoidal
Error de entrenamiento:	< 0,1
Ganancia función sigmoidal:	1

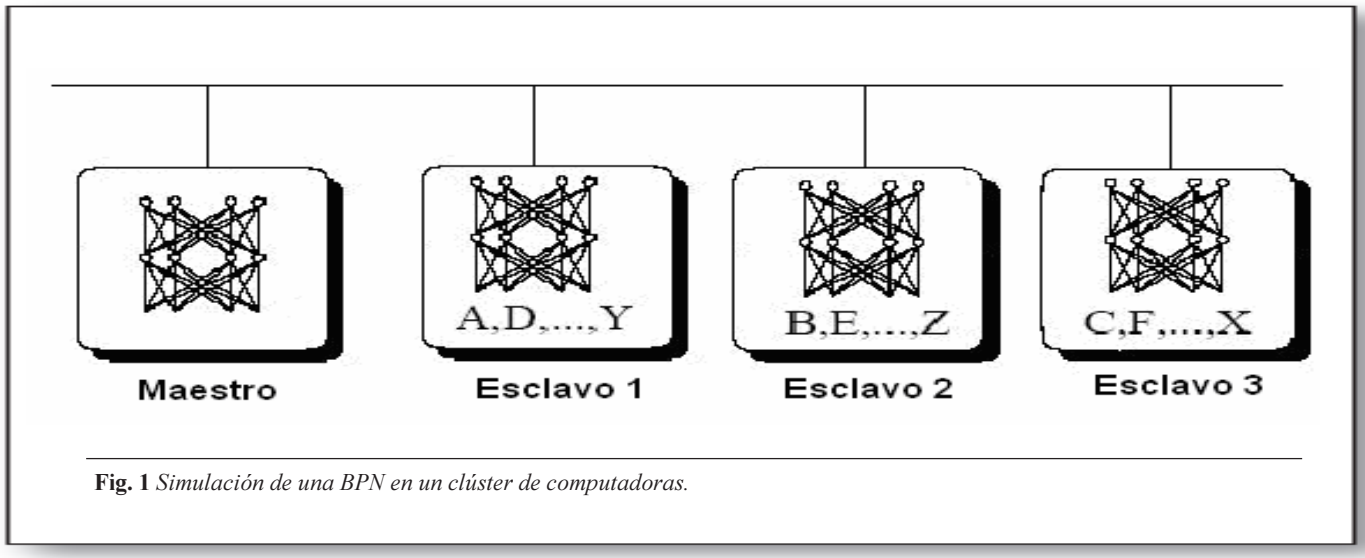

Fig. 1 *Simulación de una BPN en un clúster de computadoras.*

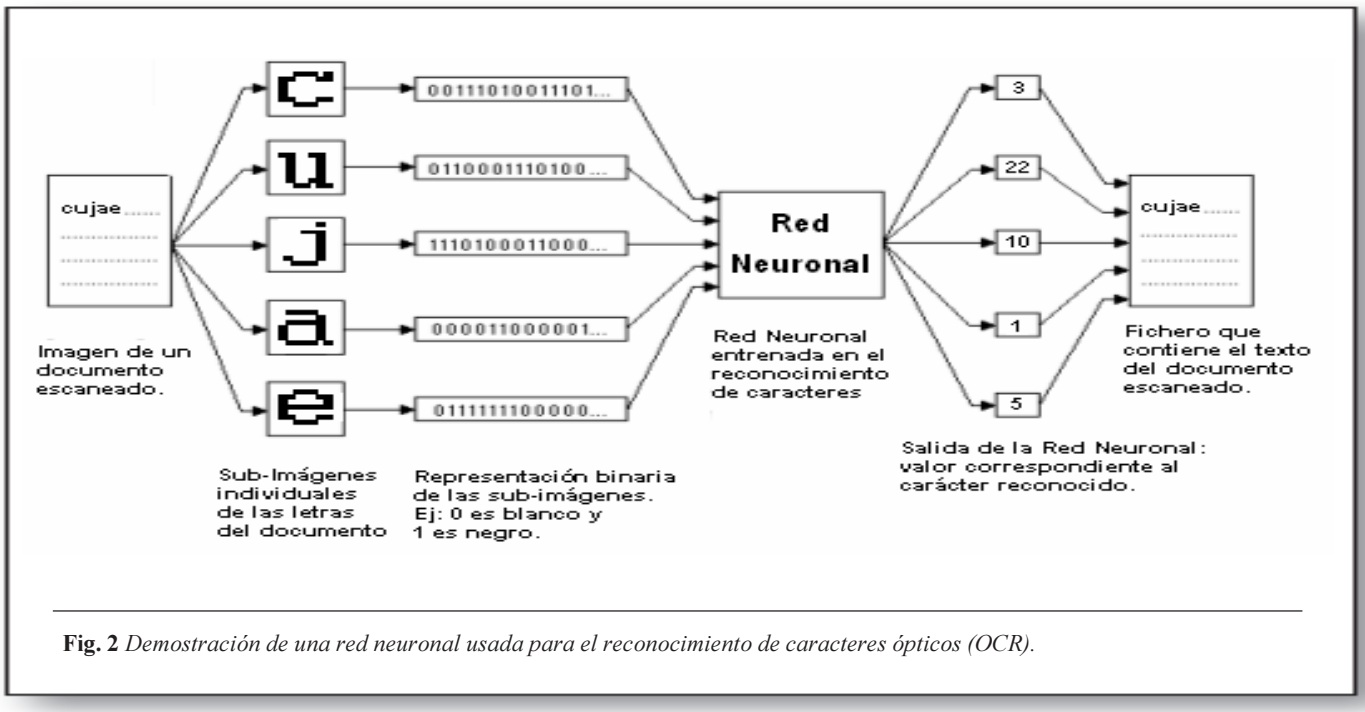

Fig. 2 *Demostración de una red neuronal usada para el reconocimiento de caracteres ópticos (OCR).*

La tabla 1 muestra los valores de *Speed-Up* y Eficiencia obtenidos para el mayor tamaño de problema seleccionado. La variable *n* representa la cantidad de casos de entrenamiento.

La figura 3, que agrupa los resultados de la tabla anterior y los correspondientes a los otros dos tamaños de problema seleccionados, muestra en el eje horizontal los números de procesadores y en el vertical los valores de *Speed-Up* para diferentes tamaños del problema. En esta se observa que con el aumento del número de procesadores para un tamaño de problema fijo, el *Speed-Up* aumenta hasta 4 procesadores y luego disminuye gradualmente. Esto indica que, para los tamaños de problemas seleccionados, con un aumento de más de 4 procesadores no se logra incrementar la velocidad de entrenamiento de la red neuronal debido a que el tiempo requerido en comunicación es superior a la reducción de tiempo obtenida por el procesamiento en paralelo. Manteniendo el número de procesadores fijo se observa que la métrica aumenta a medida que se incrementan las dimensiones del problema. Se aprecia que los mayores incrementos de velocidad se logran en los problemas de mayor tamaño.

TABLA 1
n = 81

Procesadores	Tiempo (min)	Speed-Up	Eficiencia (%)
1	276,51	1,00	100,00
2	182,01	1,52	75,96
4	101,07	2,74	68,40
6	151,57	1,82	30,41
8	190,19	1,45	18,17
10	212,49	1,30	13,01

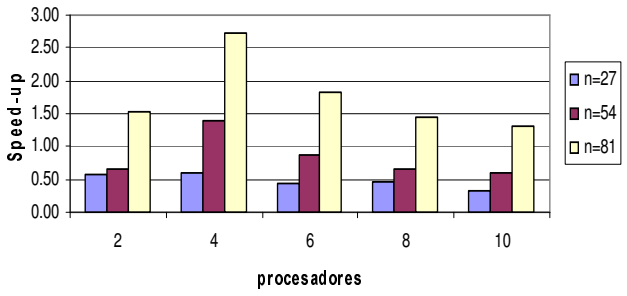

Fig. 3 *Valores de Speed-Up para la Red BPN.*

Hardware utilizado

La aplicación fue puesta a punto y probada utilizando el clúster de computadoras que se encuentra en el Centro Nacional de Bioinformática (BIOINFO) situado en la Academia de Ciencias de Cuba. La arquitectura del clúster es la siguiente:

- Un nodo maestro (dual PIII, 933 MHz, 1024 MB RAM, 120 GB HDD).
- Ocho nodos esclavos (dual PIII, 933 MHz, 512 MB RAM, 40 GB HDD).
- Un servidor de archivos (dual PIII, 933 MHz, 2048 MB RAM, 600 GB HDD, ATA RAID).

CONCLUSIONES

Con la realización de este trabajo se logró acelerar el entrenamiento de las redes neuronales BPN, mediante la utilización de la programación paralela, concluyéndose que:

- El paralelismo de casos de entrenamiento, utilizado para la red BPN, es más prominente cuando aumenta el tamaño de la colección de patrones de entrenamiento.
- En la arquitectura maestro-esclavo se pueden obtener buenos resultados, sin afectar el balanceo de carga, asignando una parte del problema a solucionar al procesador maestro, siempre que no se afecten sus tareas.
- Cuando se utiliza un clúster de computadoras con más de un procesador por máquina, los mejores resultados se obtienen cuando este se dedica solamente a ejecutar una tarea. De existir varias, las mismas compiten por los recursos de comunicación, lo que afecta el rendimiento general del hardware paralelo.
- En aplicaciones muy pequeñas, el entrenamiento secuencial puede llevarse a cabo con cierta eficiencia, sin embargo, la importancia del entrenamiento paralelo resulta prominente con el aumento de los datos y el tamaño de la red, lográndose reducciones significativas del tiempo total de entrenamiento. Por la gran cantidad de aplicaciones que presentan las redes neuronales, un sistema como el propuesto ayuda a la utilización de este enfoque en diferentes sectores del país.

REFERENCIAS

1. **ALFREDO, CATALINA:** *Introducción a las redes neuronales artificiales*., http://www.gui.uva.es/ login/13/redesn.html (28/2/03).
2. **GARAY, MIGUEL:** *Redes neuronales*. Centro de Estudios de Ingeniería de Sistemas (CEIS), Cujae, Ciudad de La Habana, 1997. \\ceis\clases\pregrado\5to\SII2\Red-Neu\Capitul1.doc (31/3/03).
3. **FOSTER, IAN:** *Designing and Building Parallel Programs*. http://www-nix.mcs.anl.gov/dbpp/ (18/12/03).
4. **VIDAL, ANTONIO:** *Presente y futuro de la computación paralela*, Ciudad de La Habana, 2003, \\ceis\DPPDValencia\Conceptos y Métodos de la Programación Paralela\sesion0 (10/1/04).
5. **FREEMAN, J. A. AND D. M. SKAPURA:** *Redes neuronales. Algoritmos, aplicaciones y técnicas de programación*, Addison-Wesley/Díaz de Santos, 1993.
6. **SUNDARARAJAN, N.; P. SARATCHANDRAN AND JIM TORRESEN:** *Chapter1 Introduction. Parallel Architectures for Artificial Neural Networks*, http://heim.ifi.uio.no/~jimtoer/ (27/1/04).
7. **KRONSJO, L. Y D. SHUMSHERUDDIN:** *Advances in Parallel Algoritms*, Blackwell Scientific Publications, Oxford, 1992.