HERRAMIENTA DE SIMULACIÓN PARA VERIFICAR EL EFECTO DE LA PRECISIÓN EN LA IMPLEMENTACIÓN DE REDES NEURONALES

N. TOLOSANA[1], N. MEDRANO[2], B. MARTÍN[3]
[1]*Dpto. de Ingeniería Mecánica. Centro Politécnico Superior, Universidad de Zaragoza. 50015 - Zaragoza. España.*
[2]*Dpto. de Ing. Electrónica y Comunicaciones. Facultad de Ciencias, Universidad de Zaragoza. 50009 - Zaragoza. España.*
[3]*Dpto. de Ing. Electrónica y Comunicaciones. Escuela Universitaria de Ingeniería Técnica Industrial, Universidad de Zaragoza. 50009 - Zaragoza. España.*

Las Redes Neuronales Artificiales son elementos de procesado capaces de resolver problemas complejos, de alta no linealidad, con datos incompletos y ruido. La fuerte dependencia de su eficacia en la resolución de problemas con la precisión en que son implementadas puede ser evidenciada mediante programas de simulación específicos como el que se presenta en este trabajo, resultando de gran interés en cursos relacionados con la realización electrónica de estos sistemas.

1. Introducción

Las Redes Neuronales Artificiales (RNA) son sistemas de procesamiento basados en el modo de trabajar del sistema nervioso de los seres vivos [1]. Por sus características, son capaces de resolver problemas complejos, de alta nolinealidad, con datos incompletos y ruido. Una de las propiedades más importantes de estos sistemas es su capacidad de abordar tareas sin necesidad de disponer de un modelo previo de comportamiento del problema. Sus áreas de aplicación son muy diversas, como la predicción, clasificación de patrones o estimación funcional, lo que las hace útiles en reconocimiento de texto escrito (OCR), oral (*speech recognition* y *user recognition*), visión artificial, monitorización de procesos complejos [2], predicción de consumo [3], o modelización [4], entre otros muchos ámbitos.

En la mayoría de las ocasiones la implementación se realiza como un programa informático ejecutándose sobre un ordenador, de manera que se dispone de todos sus recursos (microprocesador de altas prestaciones, gran cantidad de memoria, etc.). Sin embargo, cuando se precisa un sistema portátil, pequeño y de bajo consumo, la solución pasa por desarrollar sistemas electrónicos específicos, usualmente en punto fijo, basados en microcontroladores o FPGAs. Tal es el caso de los sistemas de disparo del air-bag de los vehículos, o dispositivos de preprocesado de señales de sensores. En tales circunstancias es de gran importancia conocer los requisitos del sistema necesarios para desempeñar correctamente las funciones, tanto de memoria, capacidad de E/S, consumo o precisión en los cálculos.

La mayoría de los cursos y bibliografía en que se tratan estos sistemas centran sus esfuerzos en la presentación del paradigma computacional que las redes neuronales suponen, así como de los modelos de mayor aplicación, como el perceptrón multicapa (MLP) o los mapas auto-organizados. Cuando se aborda la cuestión de la implementación, normalmente se contempla su realización por programa, siendo la electrónica un mero repaso de productos comerciales o procedentes de centros de investigación. En la actualidad existen muy pocos recursos para analizar la operación de un sistema neuronal implementado en precisión limitada. Con este propósito se ha desarrollado un programa que muestra los efectos asociados al uso de punto fijo (4, 8, 16, 32 bits...) en la aplicación de mapas auto-organizados y MLP con aprendizaje por retropropagación de errores [1] en problemas concretos. Este software, al que se le ha dado en nombre de *Netie*, está siendo empleado en cursos de doctorado dedicados a la implementación electrónica de RNA.

2. Representación numérica en punto fijo

La codificación binaria en punto fijo consiste en la distribución de los n bits que constituyen el número en dos partes, una correspondiente a la parte entera y otra a la parte decimal. Así, el número decimal 3.25867 podría representarse en punto fijo y complemento a dos para 8 y 16 bits como muestran las dos primeras columnas de la Tabla 1.

Esta representación precisa escalado tras determinadas operaciones dado que, por ejemplo, un producto de dos números de n bits da como resultado un número de 2n bits. Sin embargo, la realización de circuitos electrónicos que lleven a cabo operaciones con números en esta representación resulta relativamente sencillo [5].

3. Representación numérica en punto flotante

Es el tipo de codificación de números empleada por los ordenadores convencionales. Permite obtener un rango dinámico de representación numérica, sin la necesidad de escalar los operandos de una operación. Similar a la empleada en la notación científica, consta de dos partes, la mantisa M y el exponente e, de forma que cualquier número N se representa

$$N = M \times \beta^{e} \tag{1}$$

donde β es la base del exponente, común a todos los números en esta representación, por lo que se encuentra implícita (2 en codificación binaria). Los n bits que representan de esta forma un número real se reparten entre la mantisa y el exponente. En relación a la representación en punto fijo, a igualdad en número de bits esta codificación posee menos precisión, aunque permite el manejo de grandes rangos numéricos. Así, el número 3.25867 podría escribirse con 16 bits como muestra la tercera columna de la Tabla 1. En esa configuración, el primer bit corresponde al signo de la mantisa (10 bits) y el exponente (6 bits) se representa en complemento a dos.

La implementación electrónica de operaciones matemáticas en punto flotante resulta compleja y costosa en espacio, por lo que los sistemas electrónicos que no se basan en PC suelen hacer uso de microcontroladores o DSP trabajando en punto fijo.

8 bits punto fijo (3 parte entera, 5 decimal)	*16 bits punto fijo (3 parte entera, 13 decimal)*	*16 bits punto flotante (10 mantisa, 6 exponente)*
011.01000 → 3.25	011.0100001000111 → 3.258667	0110100001000010 = $0.110100001 \times 2^{000010}$ → 3.2578

Tabla 1: Representación del número 3.25867 en diferentes precisiones.

4. Aproximación funcional

Una de las aplicaciones más extendidas de los sistemas neuronales es el ajuste de funciones. En este caso, resulta de gran interés determinar los requisitos mínimos que el sistema sobre el que se va a implementar la red necesitan para su correcto funcionamiento. Además, mediante su simulación es posible determinar otras características como los valores de parámetros de aprendizaje, número de capas y neuronas por capa o tipo de función de salida por neurona. La Figura 1 muestra la salida de *Netie* para entrenamientos de un MLP para un problema de aproximación funcional (la función *sen(x)*), para precisiones en punto flotante (64 bits), punto fijo en 8 y 16 bits así como el comportamiento de la red entrenada en punto flotante, posteriormente aplicada en punto fijo para precisiones de 16 y 8 bits. Como se puede apreciar en la figura, dependiendo del error permisible en la respuesta de la red es posible escoger diferentes soluciones. Además, puede observarse cómo una red entrenada sobre PC (64 bits, punto flotante) da en general mejores resultados al ser trasladada a punto fijo que si el propio entrenamiento se lleva a cabo en esta representación.

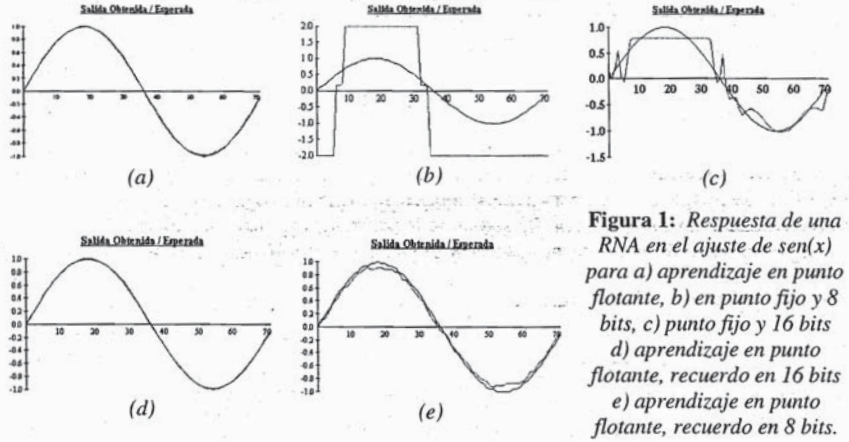
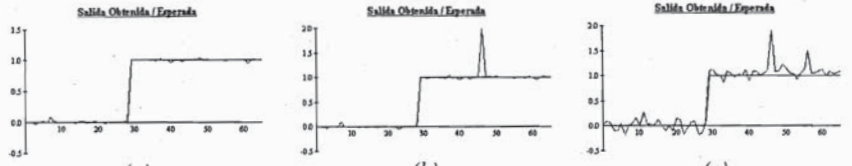


Figura 1: *Respuesta de una RNA en el ajuste de sen(x) para a) aprendizaje en punto flotante, b) en punto fijo y 8 bits, c) punto fijo y 16 bits d) aprendizaje en punto flotante, recuerdo en 16 bits e) aprendizaje en punto flotante, recuerdo en 8 bits.*

5. Clasificación

La clasificación de patrones complejos es otro de los campos de aplicación típicos de las RNA. En este caso, lo que se busca no es una función de transferencia que proporcione una salida más o menos continua ante un conjunto de datos de entrada, sino un valor numérico que separe datos en diferentes grupos, estableciendo así una clasificación. Este tipo de aplicaciones requiere en general mucha menos precisión numérica. Escogiendo

convenientemente los valores de salida que identifican las diferentes clases, se considera que un patrón corresponde a una determinada clase si la salida que proporciona no se aleja más de una cierta cantidad del valor identificativo de dicha clase. En la Figura 2 se muestra la aplicación de un MLP-BP a la clasificación binaria de patrones (salidas posibles: 0 y 1). Suponiendo que un patrón pertenece a la clase 0 si la salida que proporciona está en el rango de -0.3 a 0.3, y a la clase 1 si está entre 0.7 y 1.3, el nivel de aciertos es del 100% para una aplicación en pto. flotante, 98.5% para el caso de 16 bits en punto fijo y del 95.5% para el caso de 8 bits, partiendo en todos los casos de una red entrenada en punto flotante y 64 bits.



(a) *(b)* *(c)*

Figura 2: *Comportamiento de una red neuronal entrenada en punto flotante para un ejemplo de clasificación binaria: a) recuerdo en punto flotante, b) en 16 bits, c) en 8 bits.*

6. Conclusiones

En este trabajo se presenta una aplicación de simulación de redes neuronales artificiales que permite comprobar el comportamiento de determinados modelos neuronales en función de la precisión con la que son aplicados, tanto en aprendizaje como en recuerdo. Ello permite estudiar los requisitos mínimos que son precisos para implementar una solución de este tipo para problemas concretos, así como comprobar su comportamiento ante cambios en la arquitectura neuronal (número de neuronas, capas, parámetros de aprendizaje). Esta aplicación está siendo utilizada en cursos de doctorado relacionados con la implementación electrónica de redes neuronales artificiales.

Agradecimientos

Este trabajo ha sido financiado por la Diputación General de Aragón, Proyecto núm. P82/98.

Referencias

[1] S. Haykin. *Neural Networks, a Comprehensive Foundation.* Macmillan, 1994.

[2] M. Taylor, P. Lisboa (eds.). *Techniques and Applications of Neural Networks.* Ellis Horwood, 1993.

[3] B. Martín, N. Medrano, J.A. Domínguez, I. Ramírez, J. Barquillas, J. Blasco, J.L. García. *Proc. of the 14th IASTED International Conference on Modelling, Identification and Control*, 218-222. Austria, Febrero 1995.

[4] N. Medrano, B. Martín. *Proc. of the 2000 IEEE International Symposium on Circuits and Systems, ISCAS'2000*, vol. II, 497-500. Ginebra (Suiza), Mayo 2000.

[5] I. Koren. *Computer Arithmetic Algorithms.* Prentice Hall, 1993.