RECURRENT NEURAL NETWORK IN A FIELD PROGRAMMABLE GATE ARRAY FOR POSITION CONTROL OF MOBILE ROBOT

REDES NEURONALES RECURRENTES EN DISPOSITIVOS LOGICOS PROGRAMABLES PARA EL CONTROL DE UN ROBOT MOVIL

Ing. Johnny Omar Medina Durán*, MSc. Julián Ferreira Jaimes*
PhD. Oscar Eduardo Gualdrón Guerrero**

*** Universidad Francisco de Paula Santander**, Dpto. de Ing. Electrónica
Av. Gran Colombia, Colsag, Norte de Santander, Colombia.
**** Universidad de Pamplona**
Ciudadela Universitaria. Pamplona, Norte de Santander, Colombia.
Tel.: 57-7-5685303, Fax: 57-7-5685303, Ext. 156
E-mail: {jomedina, jferreir}@bari.ufps.edu.co, oscar.gualdron@unipamplona.edu.co,

Abstract: This paper presents the implementation of a Recurrent Neural Network (RNN) for position control of mobile robot in an unknown. It uses a Nexys Digilent Card X3S1000 based in a Field Programmable Gate Array (FPGA). A class of neural network called Jordan determines actions to be executed. A Matlab tool called NNTool was used for development, training and simulation of ANN mathematical model. ANN in computer has been implemented in hardware using Matlab/Simulink and Xilinx System Generator (XSG). In order to validate the ANN performance, a Mobile robot is used for get information about its near environment. This robot employs four DC Motors with gearbox each one.

Resumen: En este trabajo se presenta la implementación de una Red Neuronal Recurrente (RNN) para el control de desplazamiento de un robot móvil en un ambiente desconocido, en una tarjeta de desarrollo Nexys de Digilent que contiene un FPGA (Field Programmable Gate Array) X3S1000. Las acciones a ejecutar por parte del robot móvil las determina una red neuronal artificial, específicamente un a red de Jordan. Se utilizo para la creación, el entrenamiento y la simulación del modelo computacional de la RNA, la herramienta de redes neuronales de Matlab, llamada NNTool. Esta red computacional es traducida a un modelo realizable en hardware, descrito mediante bloques en Matlab/Simulink y Xilinx System Generator (XSG). El desempeño de la RNA se valida en un robot móvil que tiene sensores infrarrojos para obtener la información del medio ambiente por el cual se desplaza y cuatro motoreductores como actuadores.

Keywords ANN, Mobile robot, FPGA, Xilinx System Generator, NNTool, ISE Fundation

1. INTRODUCCION

Las RNN son ideales para problemas tales como la identificación y clasificación de patrones secuenciales con distintas probabilidades de ocurrir a través del tiempo. El comportamiento no lineal que las interconexiones de retroalimentación ocasionan en las RNN las hace óptimas para resolver estos

problemas, de hay que se haya optado por escoger este tipo de red para controlar el sistemas de navegación del robot ya que la naturaleza de sus movimientos es secuencial.

La complejidad de este tipo de redes es alta en comparación con una red feedforward, ya que en esta última la red sólo es capaz de transmitir la información hacia las capas siguientes resultando en un efecto de propagación hacia atrás en el tiempo. Las redes neuronales recurrentes, en cambio, realizan el intercambio de información entre neuronas de una manera mucho más compleja y por sus características, dependiendo del tipo de algoritmo de entrenamiento que se elija, pueden propagar la información hacia delante en el tiempo, lo cual equivale a predecir eventos. Esta es una característica muy importante para ciertas aplicaciones, ya que la capacidad de predicción de eventos significativos basada en las entradas anteriores al sistema le proporciona un beneficio importante a la seguridad del mismo.

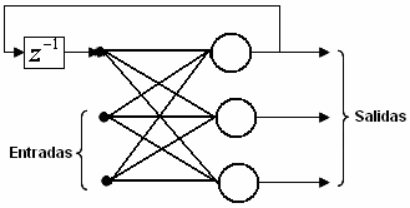

Fig. 1. arquitectura básica de una RNN

La arquitectura básica de una RNN se muestra en la Figura 1. Una característica importante es la inclusión de delays (z-1) a la salida de las neuronas en las capas intermedias; las salidas parciales smn(t+1) se convierten en valores smn(t), un instante de tiempo anterior, y así se retroalimentan a todos los componentes de la red, guardando información de instantes de tiempo anteriores. Puede observarse cómo todos los nodos están interconectados entre sí y también con algunos nodos anteriores a ellos a través de conexiones con delays antes de cada capa, o memorias temporales. El diagrama ha sido simplificado para no incurrir en excesiva complejidad, pero cada una de las capas está representada por cierto número de neuronas.

Las RNN son más eficaces para resolver problemas con no-linealidades temporales significativas. Son especialmente útiles en aplicaciones tales como el reconocimiento de patrones secuenciales, cambiantes en el tiempo, ya que las capacidades de predicción y mapeo de las RNN así lo permiten.

En los sistemas biológicos, los cuales fueron las bases conceptuales de las redes neuronales, el número de interconexiones entre todas las neuronas es muy grande. Las RNN pueden aproximarse más a ese comportamiento que los demás tipos de redes, pero por lo general la complejidad intrínseca de éstas requiere tiempos de procesamiento muy superiores. Las características variables de sus estados internos a través del tiempo también hacen muy importante considerar en qué momento deben actualizarse los pesos: al final de cada época (*epoch-wise training*) o continuamente.

2. ANTECEDENTES

2.1 Red Neuronal Artificial

La idea de la RNA surge del comportamiento del cerebro de los mamíferos. Una neurona del cerebro de un mamífero puede solamente ejecutar 100 operaciones por segundo, pero a su vez ejecutar miles de veces más instrucciones que un computador digital ("10.000 trillones de operaciones por segundo") en los cuales los programas se ejecutan a cientos de Megahertz o en unos cuantos Gigahertz. La razón de esta diferencia radica en el desempeño, ya que el cerebro mamífero es una red altamente paralela, la cual tiene miles de neuronas funcionando simultáneamente, mientras un computador digital puede realizar solo una operación en un ciclo de reloj. Una RNA es un sistema que simula esas cualidades de la red neuronal biológica, esta puede ejecutarse en un computador digital, pero entonces, el poder que ellas ganan con el paralelismo, se pierde en el tiempo requerido para ejecutar todas las conexiones paralelas serialmente. Por esta razón se hace necesaria la implementación en hardware de las RNAs para aprovechar al máximo el paralelismo que las caracteriza.

2.2 Dispositivos Lógicos programables

Los *Programmable Logic Device* (PLD's) son circuitos integrados en los que se pueden programar ecuaciones lógicas Booleanas, tanto combinacionales como secuénciales. Existen actualmente una gran variedad de estos chips, y algunos de ellos pueden contener hasta 5'000.000 compuertas lógicas. Está formado por una matriz de puertas AND conectada a otra matriz de puertas OR más biestables. Cualquier circuito lógico se puede

implementar, por tanto, como suma de productos.
Los dispositivos lógicos programables con mayor cantidad de recursos son los arreglos de compuertas programables en campo (FPGA) y los dispositivos lógicos programables complejos (CPLD).

Las FPGA, introducidas por Xilinx en 1985, También se denominan arreglos de celdas lógicas (LCA). Consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

Los CPLDs constituyen una extensión del concepto de los dispositivos lógicos simples (SPLDs) a un nivel de integración más alto. En lugar de fabricar dispositivos con matrices programables cada vez mayores, los CPLDs están formados por la interconexión de muchos bloques lógicos, cada uno de ellos similar a una pequeña PAL.

Las FPGAs y CPLDs dada su arquitectura física descrita con anterioridad, ofrecen características tales como alta velocidad de procesamiento, procesamiento concurrente y diseño jerárquico, que hacen a estos dispositivos ideales para la implementación de RNA.

3. RNN PARA EL CONTROL DE UN ROBOT MOVIL SEGUIDOR DE LINEA.

Para verificar el comportamiento de las RNAs implementadas en PLDs se tomo como escenario de prueba una de las categorías comunes en competencias de robótica, son robots que pueden seguir una línea negra, ubicada sobre una superficie blanca. Esta aplicación nos permite desarrollar estrategias de control para el desplazamiento de robot móvil.

El sistema de control o máquina de inferencia que determina las acciones a realizar para el desplazamiento del robot móvil sobre una línea, debe estar preparado para afrontar cruces con ángulos de 90 grados, y se puede desarrollar con una red neuronal realizando los procedimientos de:

- Obtención y análisis de los datos.
- Selección y entrenamiento de la RNA.
- Desarrollo del modelo en Xilinx System Generator (XSG) de la RNA.
- Implementación de la RNA utilizando ISE Foundation.

3.1 Obtención y análisis de los datos

Es importante que el diseñador conozca muy bien el sistema a controlar, pues la información o base de conocimiento para el entrenamiento determina el tipo de red neuronal a implementar. En el caso particular del robot móvil se tienen 6 sensores infrarrojos que suministran la información para el seguimiento de la línea negra y para el desplazamiento cuenta con 4 motorreductores que permiten cuatro movimientos básicos: giro a la izquierda, giro a la derecha, hacia adelante y hacia atrás, a partir de cuatro señales de control.

El comportamiento básico del robot móvil se explica a continuación, En la figura 2a se observa que los sensores frontales se encuentran sobre la línea de tal manera que para este la orden será que se desplace hacia el frente, en la figura 2b el robot ha encontrado un cruce y se ha salido de la linea para lo cual la acción que debe tomar el controlador será moverse hacia delante esperando encontrar el cruce, en la figura 2c el robot encuentra un cruce a la derecha para lo cual el controlador debe tomar la decisión de girar a la derecha, en la figura 2d el estado que entregan los sensores es el mismo que el que entregan en la figura 2b, sin embargo el controlador a diferencia de en el caso de la figura 2b que era avanzar hacia delante la decisión que debe tomar es rotar hacia la derecha como lo venia haciendo, denotando que el sistema no es combinacional sino secuencial .

Ya que el sistema es secuencial se puede explicar con un diagrama de estados como el presentado en la figura 3 el cual nos presenta claramente las deciciones que deben ser tomadas por el controlador neuronal.

Fig. 2. Comportamiento robot seguidor de línea

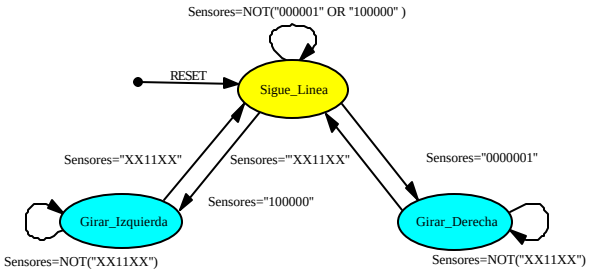

Fig. 3. Comportamiento robot seguidor de línea

Con ayuda de la figura 2 y 3 se analizan las posibles combinaciones de los sensores y se determina la acción a ejecutar por el robot movil como se muestran en la tabla 1, donde SI, S2, S3, S4, S5 y S6 son los sensores, m1, m2, m3, y m4 son las señales de control de los motores.

3.2 Selección y entrenamiento de la RNA

La red neuronal seleccionada es la de Jordán ya que es una red recurrente que permite adaptarse a sistemas digitales secuenciales. Esta red se muestra en la figura 4, con 8 bits de entrada, una sola capa con 6 neuronas y se usa como función de transferencia el limitador *hardlin* fuerte.

Para el entrenamiento de la RNN se utilizó el NNtool de Matlab donde una vez realizado el entrenamiento se obtuvieron los siguientes pesos: [-5 2 -1 -1 2 2 5 2; 3 -2 1 0 0 -1 -3 -1; 4 -4 2 2 3 2 -1 -3; -2 2 0 0 -3 -1 0 2; 3 -2 1 0 0 -1 -3 -1; -2 2 0 0 -3 -1 0 2], y valores de polarización [0; -1; 0; -1; -1; -1].

La RNN en el entrenamiento converge de una manera muy rápida, alcanzando un porcentaje de error 0, lo cual indica que todas las entradas fueron correctamente clasificadas.

Tabla 1: Relación entrada-salida de la RNA para el seguidor de línea

Entradas								Salidas					
A	B	1	2	3	4	5	6	1	2	3	4	A	B
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	1	0	0	0
0	0	0	0	0	1	1	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	1	0	1	0	1	0	0	0
0	0	0	0	1	1	0	0	1	0	1	0	0	0
0	0	0	0	1	1	1	0	1	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	1	1	0	0	1	0	1
0	0	1	0	0	0	0	1	1	0	0	1	0	1
0	0	1	0	0	0	1	0	1	0	1	0	0	0
0	0	1	0	1	1	0	0	1	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	1	0	1	0
1	0	1	0	0	0	0	0	0	1	1	0	1	0
1	0	1	0	0	0	0	1	0	1	1	0	1	0
1	0	0	0	0	0	0	1	0	1	1	0	1	0
1	0	0	1	0	0	0	0	0	1	1	0	1	0
1	0	0	1	1	0	0	0	0	1	1	0	1	0
1	0	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	0	1	1	0	0	0	1	1	0	1	0
1	0	0	0	0	1	0	0	0	1	1	0	1	0
1	0	0	0	0	1	1	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	1	0	0	1	0	1
0	1	0	0	0	0	0	1	1	0	0	1	0	1
0	1	1	0	0	0	0	0	1	0	0	1	0	1
0	1	1	0	0	0	0	1	1	0	0	1	0	1
0	1	0	0	0	0	1	0	1	0	0	1	0	1
0	1	0	0	0	1	1	0	1	0	0	1	0	1
0	1	0	0	0	1	0	0	1	0	0	1	0	1
0	1	0	0	1	1	0	0	1	0	1	0	0	0

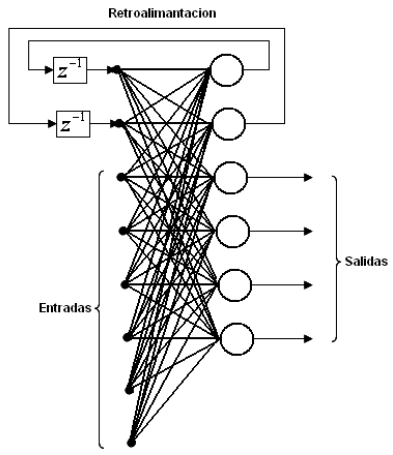

Fig. 4. Arquitectura RNA para el seguidor de línea

3.3 Desarrollo del modelo en XSG de la RNA

La estructura de la red neuronal mostrada en la figura 4, es implementada en XSG por medio de bloques sumadores, multiplicadores, constantes y bloques MCode, este último se utiliza para desarrollar la función de transferencia, estos bloques se muestran en la figura 5 que representa la implementación de una sola neurona. Como se puede ver cada una de las ocho entradas es multiplicada por el peso calculado en el entrenamiento y una vez realizada la suma de estos cuatro resultados se adiciona el valor de Bias y se le aplica la función de transferencia para obtener finalmente la salida de la RNA.

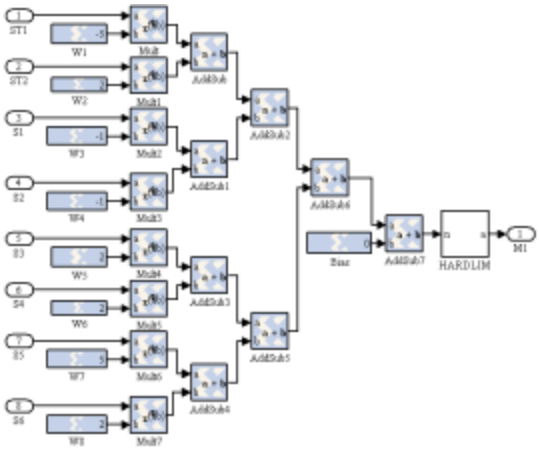

Fig. 5. Arquitectura interna de una neurona en XSG

Uno de los parámetros más importantes a configurar en los bloques multiplicadores, sumadores y constantes, es la precisión; la opción Full para la precisión, en los parámetros de configuración de estos bloques, permite obtener un funcionamiento óptimo sin problemas de sobre-flujo o saturación, con la consecuencia de un mayor consumo de recursos de la FPGA. Si en un diseño no se tienen limitaciones en recursos, resulta práctico utilizar esta opción, debido a que ahorra tiempo durante la puesta en marcha del diseño cuando se tienen un número grande de estos elementos, pero si por el contrario el ahorro de recursos es fundamental o los recursos disponibles son limitados, se debe seleccionar debidamente el tipo de salida, especificando el numero de bits, la posición del punto binario, el tipo de cuantización y el tipo de sobreflujo, que permitan representar correctamente las cantidades.

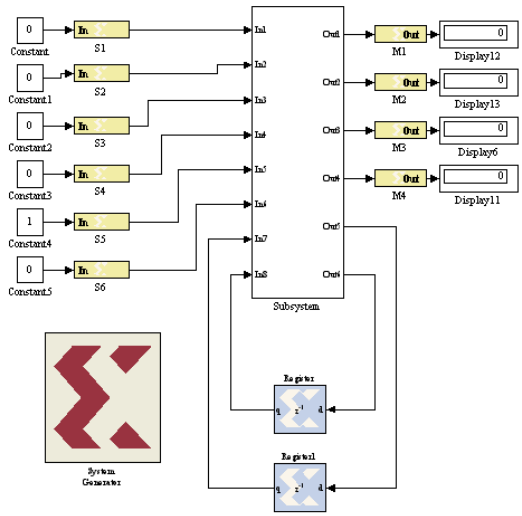

Fig. 6. Modelo en XSG de la red neuronal

Al analizar los pesos determinados con el NNtool de Matlab se puede ver que el rango de valores está entre -5 y 5 lo que permite utilizar datos en XSG con una resolución de cuatro bit con el fin de utilizar la menor cantidad de recursos.

En la figura 6 se observa el modelo completo en XSG del controlador del robot móvil seguidor de línea utilizando redes neuronales donde se identifican las entradas provenientes de los sensores, la red neuronal de una capa formada por las seis neuronas y las salidas que controlan la dirección de desplazamiento del robot móvil.

Antes de probarse en el sistema mecánico el controlador es simulado en Matlab. La figura 7 muestra los resultados arrojados por la simulación del sistema de control.

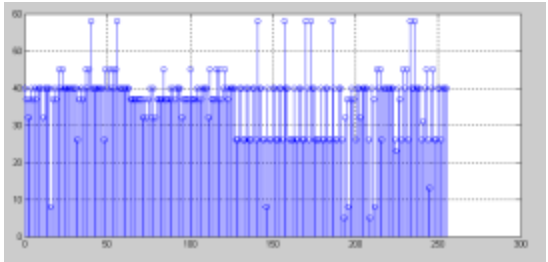

Fig. 7. simulación de entradas contra salidas

3.4 Implementación de la RNA utilizando ISE Foundation

Una vez el diseño ya ha sido simulado y se encuentra funcionando de la manera deseada, se debe realizar la asignación de los puertos del

hardware a implementar, es decir, hacer la asignación de pines de acuerdo con el FPGA que se utilizará. La asignación de pines se realiza en los Gateway de entrada y salida, el tipo de FPGA se indica cómo se muestra en la figura 8, ingresando en el bloque System Generate, desde este mismo bloque se genera el archivo de extensión **.BIT* necesario para programar la tarjeta de desarrollo desde el entorno de Xilinx ISE Foundation dado que XSG no brinda esta posibilidad.

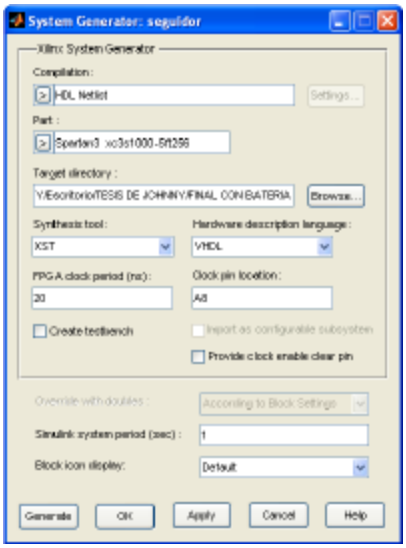

Fig. 8. Configuración de la FPGA para la implementación de la RNA

A la hora de realizar un diseño es necesario conocer la cantidad de recursos utilizados en la implementación, ya que ésta es una de las más importantes limitaciones de los dispositivos lógicos programables. En la figura 6 se pueden observar los recursos utilizados para la implementación de la RNA para el seguimiento de línea, como el número de LUTs, Flip Flops, Slices, relojes globales, y el porcentaje respecto a la capacidad total del dispositivo, de cada uno de estos, además de un valor muy importante que es el equivalente del diseño en compuertas, el cual permite conocer de una forma más general el consumo de recursos del diseño.

Analizando la figura 9, se puede ver que solo se utilizan 2887 compuertas equivalentes del millón disponibles en la FPGA XC3S1000, lo que nos indica que se pueden implementar sistemas de mayor complejidad, además de la utilización de dos elementos de memoria de 15360 disponibles.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2	15,360	1%	
Number of 4 input LUTs	232	15,360	1%	
Logic Distribution				
Number of occupied Slices	168	7,680	2%	
Number of Slices containing only related logic	168	168	100%	
Number of Slices containing unrelated logic	0	168	0%	
Total Number of 4 input LUTs	232	15,360	1%	
Number of bonded IOBs	11	173	6%	
Number of GCLKs	1	8	12%	
Number of RPM macros	48			
Total equivalent gate count for design	2,887			
Additional JTAG gate count for IOBs	528			

Fig. 9. recursos de hardware utilizados para la implementación

La resolución de tres bit utilizada en configuración de los bloques multiplicadores, sumadores y constantes en XSG permitió que la cantidad de recursos utilizados para la implementación sea 2887 compuertas que son menos del 3% de los recursos disponibles en la FPGA X3S1000 de tarjeta de desarrollo Nexys utilizada para implementar el controlador RNA para el desplazamiento del robot móvil.

4. CONCLUSIONES

Las redes neuronales recurrentes son especialmente útiles en aplicaciones tales como el reconocimiento de patrones secuenciales, cambiantes en el tiempo, ya que las capacidades de predicción y mapeo de las RNN así lo permiten.

La implementación de redes neuronales en dispositivos lógicos programables permite el desarrollo de sistemas de control en tiempo real para la navegación autónoma de robots móviles, debido a la arquitectura física que presentan estos dispositivos que permiten la implementación de sistemas concurrentes.

A la hora de implementar un diseño utilizando XSG en un PLD es importante configurar de forma adecuada la resolución de la data de cada uno de los bloques, dado que por defecto es 64 bits que causa un incremento en la cantidad de recursos utilizados siendo esta la mayor limitante de los PLD. En el caso de las RNA esta resolución la determinan los pesos calculados en el entrenamiento.

La utilización de Matlab permite crear y entrenar redes neuronales en un corto tiempo utilizando la herramienta NNtool, y Xilinx System Generator permite implementar la arquitectura de la RNA para luego programar el dispositivo lógico programable en un ambiente amigable para el diseñador.

REFERENCIAS

A. H. Bond and l. Gasser. An analysis of problems and research in DAI. In A. H. Bond and L. Gasser, editors, Readings in Distributed Artificial Intelligence, pages 3–36. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

Acosta, Maria Isabel. ZULUAGA, Camilo Alfonso.Tutorial sobre redes neuronales aplicadas en Ingeniería Eléctrica y su implementación en un sitio Web. [en línea]. Universidad Tecnológica de Pereira, 2000. p-5. http://ohm.utp..edu.co/neuronales.

Asada M., Stone P., Kitano H., Drogoul A., Duhaut D., Veloso M., Asama H., Suzuki S. .The RoboCup Physical Agent Challenge: Goals and Protocols for Phase I.. Kitano Horoaki (Eds). (1998). RoboCup .97: Robot Soccer World Cup I. Lectures Notes in Computer Science; Vol 1935: Lectures Notes in Artificial Intelligence. Berlin: Springer-Verlag.

Battiti, R. First and second order methods for learning: Between steepest descent and Newton's method, Neural Computation. Vol. 4, No. 2, 1992. p. 141-166.

Beale, E.M.L. A derivation of conjugate gradients, in F.A. Lootsma, Ed., Numerical methods for nonlinear optimization. London. Academic Press, 1972.

Brooks, R. "*Intelligence without reason*". IJCAI'91. Pág. 569-595. 1991.

Brooks, R. A., "*New Approaches to Robotics*", Science, Vol. 253. Pág.1227-1232. September 1991.

Dennis, J.E. Schnabel R.B. Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Englewood Cliffs. NJ: Prentice-Hall, 1983.

Fletcher, R. Reeves, C.M. Function minimization by conjugate gradients. En: Computer Journal, Vol. 7, 1964. p. 149-154.

Guzzo, M.A. Garguilo, E. Patiño, H.D. Diseño y desarrollo de un Neuro-chip basado en FPGAs, Universidad Nacional de San Juan, San Juan, Argentina. P-24

Hagan, M.T. Demuth, H.B. Beale, M.H. Neural Network Design, Boston, MA: PWS Publishing, 1996.

Hagan, M.T. Menhaj M. Training feed-forward networks with the Marquardt algorithm. En: IEEE Transactions on Neural Networks. Vol. 5, No. 6, 1999, p. 989-993.

Kartalopoulos, S.V. Biological neural networks. In Understanding Neural Networks an Fuzzy Logic Basic Concepts and Applications, The Institute of Electrical and Electronic Engineers, Inc., New York, 1996.

Moller, M.F. A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks. Vol. 6, 1993. p. 525-533.

Moctezuma Eugenio, Juan Carlos. Torres Huitzil, Cesar. Estudio sobre la implementación de redes neuronales artificiales usando Xilinx System Generator. Puebla, México. P.1. Benemérita Universidad Autónoma de Puebla.

Patiño, H.D. Apuntes Curso Introducción a las redes Neuronales Artificiales en Ingeniería (Rama Estudiantil IEEE de la UNSJ), San Juan, Argentina, 2000.

Powell, M.J.D Restart procedures for the conjugate gradient method. Mathematical Programming, Vol. 12, 1977. p. 241-254.

Riedmiller, M. Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. Proceedings of the IEEE International Conference on Neural Networks, 1993.

Rosenblatt, F., Principles of Neurodynamics, Washington, D.C.: Spartan Press, 1961.

Tommisca, M.T. Efficient digital implementation of the sigmoid function for reprogrammable logic. En: Computers and Digital Techniques. Finlandia. Vol 150, no. 6 (Nov. 2003); p.403-411

Vivas, R. L. pasantia: estudio de métodos de inteligencia artificial y desarrollo de códigos y aplicativos en Matlab para el modelamiento de sistemas basados en árboles de decisión, lógica difusa y redes neuronales. Universidad Francisco de Paula Santander. Cúcuta, Colombia, 2005. P.25

Zhang Y., Mackworth A.K. "Constraint Nets: A Semantic Model for Hybrid Dynamic Systems". Theoretical Computer Science 130. Pág. 211-239, 1995.