Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Neural identification of dynamic systems on FPGA with improved PSO learning

Mehmet Ali Cavuslu[a], Cihan Karakuzu[b], Fuat Karakaya[c,*]

[a] Koc Information and Defense Technologies Inc., METU Technopolis, ODTÜ-Teknokent, 06800 Ankara, Turkey
[b] Bilecik Seyh Edebali University, Engineering faculty, Department of Computer Engineering, Gülümbe Campus, 11210, Bilecik, Turkey
[c] Nigde University, Engineering Faculty, Department of Electrical and Electronics Engineering, Nigde, Turkey

A R T I C L E   I N F O

Article history:
Received 9 June 2011
Received in revised form 31 January 2012
Accepted 4 March 2012
Available online 17 March 2012

Keywords:
Artificial neural networks (ANN)
Particle swarm optimization (PSO)
FPGA
System identification

A B S T R A C T

This work introduces hardware implementation of artificial neural networks (ANNs) with learning ability on field programmable gate array (FPGA) for dynamic system identification. The learning phase is accomplished by using the improved particle swarm optimization (PSO). The improved PSO is obtained by modifying the velocity update function. Adding an extra term to the velocity update function reduced the possibility of stucking in a local minimum. The results indicates that ANN, trained using improved PSO algorithm, converges faster and produces more accurate results with a little extra hardware utilization cost.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Artificial neural networks (ANNs) have been widely and successfully used in many fields to solve complex problems. ANNs are capable of successfully modeling non-linear complex correlation between input and output of a system. The most commonly used ANN is the multi layer perceptron (MLP) [1,2]. ANN is trained using a data set which is compatible with input–output relation of system to be modeled. Training is usually performed using back propagation (BP) algorithm. In recent years evolutionary algorithms are also used for ANN training as an alternative to BP algorithm.

For example population-based particle swarm optimization (PSO) [3], capable of stochastic search, has been successfully utilized in neural network training in addition to its usage in function optimization and fuzzy system control [4–7].

In recent years FPGAs, capable of parallel processing, are emerging as a predominant embedded system platform for many ANN applications [8–13]. In the literature numerous of hardware based ANN implementations on FPGAs with offline training [1,14–17] and online training [18–20] are reported. In these implementations different number formats such as fixed-point numbers with different bit width [16,18,20], floating-point numbers [1,14,17–19] and integer numbers [15] are utilized. Nonlinear logarithmic sigmoid [14–19] and nonlinear hyperbolic tangent [1,16,20] are employed as activation function. To implement nonlinear activation functions on hardware several approaches have been proposed: look-up table [15,20], piecewise-linear [1,16,18], parabolic [14] and functional [17,19].

In this work we present FPGA implementation of a system identification module based on ANN with PSO based online training. ANN and its PSO based online training module implemented in single precision floating-point number format and neural cell activation functions are implemented using functional approaches. In functional approach a divider is used in addition to adder and multiplier when compared to piecewise-linear and parabolic approach. Divider usage could be considered as the disadvantage of functional approach, but this approach has several advantages: (i) contrary to look-up table approach, no memory needed (ii) contrary to piecewise-linear approach, no control statements needed. The most important contribution of this work is the modified velocity update function of the PSO. In this paper authors proposed to add an extra term to the function which reduces the possibility of stucking in a local minimum during training. The experimental results indicate that the PSO algorithm utilizing the modified velocity update function converges faster and produces more accurate results.

The rest of the paper is organized as follows. Section 2 provides brief introduction to standard PSO and proposed improved PSO. Section 3 describes hardware implementation of ANN and its PSO based training module. Experimental results and performance of proposed approach in hardware are given in Section 4. The comparison results of the proposed improved PSO with other methods are given in Section 5 and the conclusions are given in Section 6.

* Corresponding author.
E-mail addresses: ali.cavuslu@kocsavunma.com.tr (M.A. Cavuslu), cihan.karakuzu@bilecik.edu.tr (C. Karakuzu), fkarakaya@nigde.edu.tr, karakuzu@bilecik.edu.tr (F. Karakaya).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.03.022

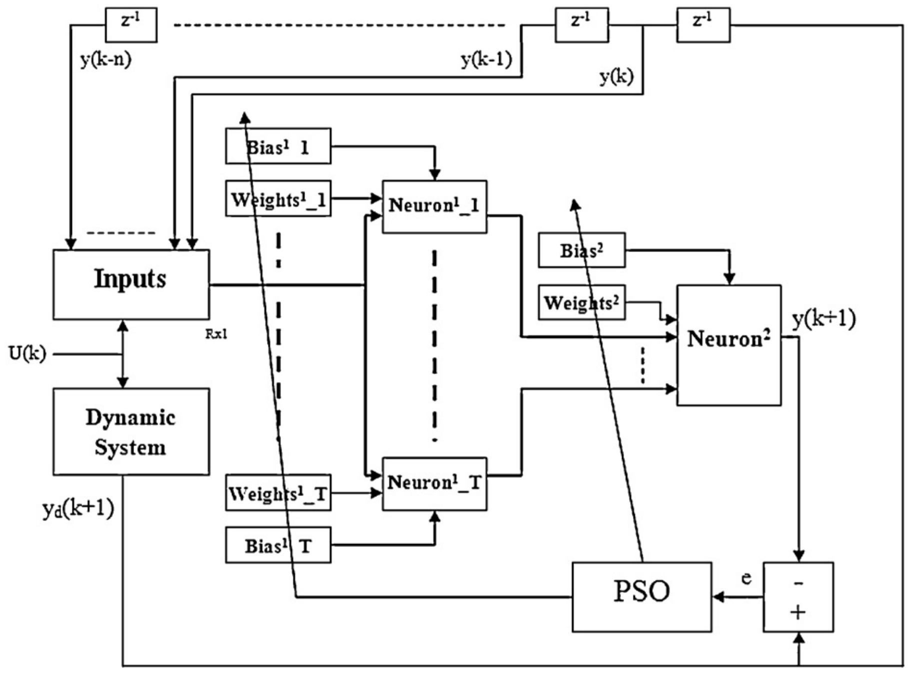

Fig. 1. PSO based ANN training and dynamic system identification architecture.

2. Standard and proposed modified PSO

PSO, inspired from social interactions among animals, is a high performance optimizer. PSO performs search using a population of particles. Each particle is directed to the most significant parts of the search space as a result of interactions between particles. The algorithm starts with randomly assigned particles. In every iteration the velocity and the location of particles are updated. Each particle's position is evaluated as a possible solution candidate. In a swarm with N particles, the position of the ith particle is defined as a vector as given in Eq. (1). D is the dimension of the search space.

$$\vec{p_i} = [p_{i1}, p_{i2}, p_{i3}, \ldots, p_{iD}] \tag{1}$$

Each particle memorizes its own best position that it has discovered, called the local best (p_b, Eq. (2)), and it also knows the best position discovered by any particle, called the global best (g_b, Eq. (3)).

$$\vec{p_b} = [p_{b1}, p_{b2}, p_{b3}, \ldots, p_{bD}] \tag{2}$$

$$\vec{g_b} = [g_{b1}, g_{b2}, g_{b3}, \ldots, g_{bD}] \tag{3}$$

The rate of change in particle position is called particle velocity and for the ith particle it is given as in Eq. (4).

$$\vec{v_i} = [v_{i1}, v_{i2}, v_{i3}, \ldots, v_{iD}] \tag{4}$$

At each iteration particle velocity is computed to determine the location of the particle at the next iteration. Particle velocity computing function, given in Eq. (5), have been proposed in [3] to compute the new velocity of each particle.

$$v_i(n+1) = \xi[v_i(n) + \alpha_1 r_1 (p_{b,i} - p_i(n)) + \alpha_2 r_2 (g_{b,i} - p_i(n))] \tag{5}$$

α_1, α_2 are called learning constants; r_1 and r_2 are uniformly distributed random numbers in [0–1) range; ξ is constriction factor. The most common problem with Eq. (5) is the possibility of stucking in a local minimum and as a result immature termination of the search. The authors of this paper propose an additional parameter to Eq. (5) which reduces the possibility of stucking in a local minimum. The proposed new velocity update function is given in Eq. (6).

$$v_i(n+1) = \xi[v_i(n) + \alpha_1 r_1 (p_{b,i} - p_i(n)) + \alpha_2 r_2 (g_{b,i} - p_i(n))] + [\alpha_3 \lambda(n)] \tag{6}$$

α_3 is also called additive learning constant; λ is a normally distributed random number vector. The value of α_3 is chosen according to the inequality given in Eq. (7). Where X is the matrix of input samples to be used for modeling or identification task.

$$\alpha_3 \ll \frac{1}{\max\{\mathrm{eig}(XX^T)\}} \tag{7}$$

After determining range of $\alpha_3\lambda$ term in modified PSO, $\alpha_3\lambda$ term is represented as power of two $[-2^{-z}\ 2^{-z}]$ (z is an integer number) to expedite the handling in hardware. As stated before, the last term in Eq. (6) prevents particles from stucking in a local minimum and it permits a more detailed search of the space. After the determination of the particle velocity, the new location of the particle is calculated using Eq. (8).

$$p_i(n+1) = p_i(n) + v_i(n+1) \tag{8}$$

3. PSO based ANN training ON FPGA

In this section, the details of PSO based ANN training on FPGA is described. The block diagram of the proposed architecture with the intention of dynamic system identification is shown in Fig. 1. PSO based ANN training on FPGA has four main stages as given in Fig. 2. Short description of these four stages is as follows.

3.1. Stage1: assignment of initial values

In FPGA implementation, the swarm (P), local best particles (P_b), global best particle (g_b), particle velocities (V_m) and fitness values (E_n) are stored in block RAMs. For P, P_b and V_m matrices a block RAM in $N \times D$ dimensions, for g_b vector a block RAM in $1 \times D$ dimension,

Download English Version:

https://daneshyari.com/en/article/496551

Download Persian Version:

https://daneshyari.com/article/496551

Daneshyari.com