Neural Random Access Machines Optimized by Differential Evolution

M.Baioletti, V.Belli, G. Di Bari, V.Poggioni

Universit di Perugia, Dip. Matematica e Informatica, Italy
{marco.baioletti,valentina.poggioni}@unipg.it,
{belli.valerio,dbgabri}@gmail,

Abstract. Keywords: NRAM, differential evolution, neural network, TROVARE ALTRO

1 Introduction

2 Neural Random Access Machines

3 Differential Evolution Neural Networks

We already present an algorithm that optimizes artificial neural networks using Differential Evolution in [REF TO MODE]. The evolutionary algorithm is applied according the conventional neuroevolution approach, i.e. to evolve the network weights instead of backpropagation or other optimization methods based on backpropagation. A batch system, similar to that one used in stochastic gradient descent, is adopted to reduce the computation time.

3.1 Differential Evolution

Differential evolution (DE) is a metaheuristics that solves an optimization of a given fitness function f by iteratively improving a population of NP candidate numerical solutions with dimension D. The population evolution proceeds for a certain number of generations or terminates after a given criterion is met.

The initial population can be generated with some strategies, the most used approach is to randomly generate each vector. In each generation, for every population element, a new vector is generated by means of a mutation and a crossover operators. Then, a selection operator is used to choose the vectors in the population for the next generation.

The fist operator used in DE is the *differential mutation*. For each vector x_i in the current generation, called *target vector*, a vector $\bar{y}_i$, called *donor vector*, is obtained as linear combination of some vectors in the population selected according to a given strategy. There exist many variants of the mutation operator (see for instance [**?**,**?**]). The common mutation (called DE/rand/1) is defined as follows:

$$\bar{y}_i = x_a + F(x_b - x_c)$$

where a, b, c are mutually exclusive indexes. The crossover operator creates a new vector y_i, called *trial vector*, by recombining the donor with the corresponding target vector by means of a given procedure. The crossover operator used in this paper is the binomial crossover regulated by a real parameter CR.

Finally, the usual selection operator compares each trial vector y_i with the corresponding target vector x_i and keeps the better of them in the population of the next generation.

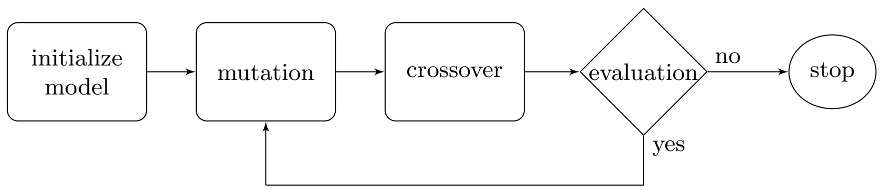

Fig. 1. The evolution of a individual.

3.2 DENN

Since the DE works with continuous values, we can use a straightforward representation based on a one-to-one mapping between the weights of the neural network and individuals in DE population.

In details, suppose we have a feed-forward neural network with k levels, numbered from 0 to $k-1$. Each network level l is defined by a real valued matrix $\mathbf{W}^{(l)}$ representing the connection weights and by the bias vector $\mathbf{b}^{(l)}$.

Then, each population element x_i is described by a sequence

$$\langle(\hat{\mathbf{W}}^{(i,0)}, \mathbf{b}^{(i,0)}), \ldots, (\hat{\mathbf{W}}^{(i,k-1)}, \mathbf{b}^{(i,k-1)})\rangle,$$

where $\hat{\mathbf{W}}^{(i,l)}$ is the vector obtained by linearization of the matrix $\mathbf{W}^{(i,l)}$, for $l = 0, \ldots, k-1$. For a given population element x_i, we denote by $x_i^{(h)}$ its h–th component, for $h = 0, \ldots, 2k-1$, i.e. $x_i^{(h)} = \hat{\mathbf{W}}^{(i,h/2)}$, if h is even, while $x_i^{(h)} = \mathbf{b}^{(i,(h-1)/2)}$ if h is odd. Note that each component $x_i^{(h)}$ of a solution x_i is a vector whose size depends on the number of neurons of the associated levels.

********* TODO *********

4 Experimental Results

5 Conclusions and Future Works

References