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

Recently a research trend of learning algorithms [4–9] by means deep learning techniques has started. The basic idea is to learn regularities in sequences of symbols generated by simple algorithms which can only be learned by models having the capacity to count and memorize. Most of these are different implementations of the controller-interface abstraction: they use a neural controller as a "processor" and provide different interfaces for input, output and memory. In order to make these models trainable with gradient descent, sometimes the authors made them "artificiously" fully differentiable modifying the discrete nature of their interfaces [4, 7].**verificare**

In this trend, we consider of particular interest the Neural Random-Access Machines, called NRAM and proposed in [7], because this model is able to solve problems with pointers: it can learn to solve problems that require explicit manipulation and dereferencing of pointers and can learn to solve a number of algorithmic problems. Moreover the authors showed that the solutions can generalize well to inputs longer than ones seen during the training. In particular, for some problems they generalize to inputs of arbitrary length.

However, as the authors themselves said, the optimization problem resulting from the backpropagating through the execution trace of the program is very challenging for standard optimization techniques.

Moreover, aspects adding complexity to the optimization problem are the continuity condition of the optimization function and the fuzzyfication process. For example, since the output are represented as probability distribution, the output of modules is a computationally costly operation. **verificare**

Since this additional cost is due to the use of gradient descent we think that different optimization methods not requiring differentiability can be of help.

This work is part of a wider project aiming to rewrite a NRAM model totally based on evolutionary optimization. In particular, in this paper we present a version of the Neural Random-Access Machines, where the core neural controller is

trained with Differential Evolution meta-heuristic instead of the usual backpropagation algorithm. In particular we propose to use the DENN method already proposed in [1] to evolve neural networks by means of Differential Evolution.

The experiments are conducted to evaluate the system from two different points of view: the ability of DENN technique to optimize the network and compete with backpropagation, and the generalization capability of our solutions. In the experiments several algorithmic problems whose solutions required pointer manipulation and chasing are chosen.

This work puts the Differential Evolution meta-heuristic to another test, where the neural networks are deep and with a lot of parameters. **mettere qualche dettaglio sui numeri delle reti** Even if DENN method has been applied in its simplest form flattening the network deep and a lot of work has to be done, these first results are very encouraging and prove that other efforts should be undertaken.

The paper is organized as follows: Section 2 and 3 will recall the two basic components combined in this work, the NRAM model and DENN technique; then the experimental part is described in Section 4 where the main results are also shown; the paper ends with Section 5 where some ideas for future works are depicted and discussed.

2 Neural Random Access Machines

In this section we briefly recall the NRAM model presented in [7]. This model can be included in the recent research trend of learning algorithms [4–6, 8, 9]. Like others, NRAM model is an implementation of the controller-interface abstraction: it uses a neural controller as a "processor" and provide different interfaces for input, output and memory.

In particular the NRAM model can learn to solve problems that require explicit manipulation and dereferencing of pointers. The controller, the core part of the NRAM model can be a feedforward neural network or an LSTM, and it is the only trainable part of the model.

3 Differential Evolution Neural Networks

We have already presented 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 [3, 2]). 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.

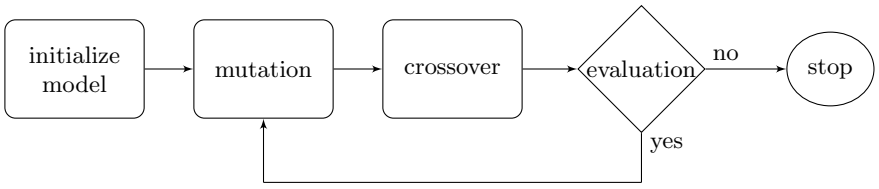

Fig. 1. The evolution of a individual.

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

In this paper we have presented a Differential Evolution algorithm for the problem of optimizing neural networks.

References

1. Marco Baioletti, Gabriele Di Bari, Valentina Poggioni, and Mirco Tracolli. Can differential evolution be an efficient engine to optimize neural networks? In *Machine Learning, Optimization, and Big Data*, pages 401–413, Cham, 2018. Springer International Publishing.
2. Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. Recent advances in differential evolution an updated survey. *Swarm and Evolutionary Computation*, 27:1 – 30, 2016.
3. Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.
4. Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.
5. Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi. Evolving neural turing machines for reward-based learning. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 117–124. ACM, 2016.
6. Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *NIPS*, 2015.
7. K. Kurach, M. Andrychowicz, and I. Sutskever. Neural random-access machines. *CoRR*, abs/1511.06392, 2015.
8. Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus. Learning simple algorithms from examples. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 421–429. JMLR.org, 2016.
9. Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. *CoRR*, abs/1505.00521, 2015.