Momentum Enhanced Neuroevolution

Gene Sher
University of Central Florida
gsher@knights.ucf.edu

ABSTRACT

The momentum parameter is common within numerous optimization and local search algorithms, particularly in the popular back propagation neural network learning algorithm. Computationally cheap and prevalent in gradient descent approaches, it is not currently utilized within neuroevolution. In this paper we present some of the results produced by a momentum enhanced neuroevolutionary algorithm. We demonstrate how this computationally inexpensive parameter in most of the cases results in enhancing the system's performance.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Evolutionary Computation; Momentum Parameter; Neuroevolution; Neural Network; Genetic Algorithm; Memetic Algorithm; DXNN

1. INTRODUCTION

The utilization of momentum in evolutionary computation (EC) has been explored in the past [6,7], but not when applied to the evolution of Neural Networks (NNs) [1]. In gradient descent based algorithms, momentum plays two main roles: 1. Promote larger weight changes in areas of shallow gradients. and 2. Prevent the algorithm from getting stuck in local optima prematurely. When it comes to EC, the purpose of momentum is to cancel out, over the long term, incorrect parameter perturbations. In this paper we add the momentum parameter to a Memetic Algorithm (MA) and Genetic Algorithm (GA) based neuroevolutionary system, and then explore, and demonstrate, its superior resulting performance. Due to the length restrictions of this paper, it will not be possible to discuss the algorithm in detail, and only two benchmarks, instead of six which span multiple domains, will be presented in this paper.

GECCO'15, July 11–15, 2015, Madrid, Spain.

ACM ISBN 978-1-4503-3472-3/15/07..

DOI: http://dx.doi.org/10.1145/2739482.2764709

2. METHOD

To demonstrate the advantages of incorporating the momentum parameter into neuroevolution, we will explore its addition into a generic MA and genetic GA algorithm based neuroevolutionary based on algorithm [8]. The GA variation of the MA algorithm is derived by removing the local search stochastic hill climbing phase, and thus resulting in a single phase based neuroevolutionary algorithm similar to NEAT [2] in performance.

2.1 Momentum Parameter

We create a version of the neuroevolutionary algorithms using the momentum parameter by modifying the weight perturbation phase (if MA), and weight perturbation mutation operator (if GA) as follows:
$\lambda W_t = (uniform() - 0.5) * Spread + \lambda W_{t-1} * M$
$W_t = W_{t-1} + \lambda W_t$
Where λW_t is Weight change, λW_{t-1} is previous weight change, $uniform()$ generates a random uniform value between 0 and 1, $Spread$ is the perturbation spread range that is based on the age of the neuron (it anneals over time) and is discussed in [1], M is the momentum parameter specified by the researcher which determines the percentage of the previous weight change, set to a value between: 0 (no momentum) and: 1 (previous weight change is subtracted completely), that gets added to the new weight change, and W_t and W_{t-1} which are the new and old weights, respectively. The momentum parameter M in the performed benchmarks was set to *0.5*, after determining that it is generally optimal value best suited in most problem domains explored.

2.2 Benchmarks

To determine how momentum effects neuroevolution, we perform benchmarks from different problem domains:
1. Double Pole Balancing [3] (DPB). (Presented in this paper)
2. Multi-Double Pole Balancing (Multi-DPB): Here the NN has to balance two double-pole carts at the same time. (Presented in this paper)
3. Visual Discrimination [4]. (Presented in full paper)
4. T-Maze navigation [9,10], with neurons that utilize Hebbian learning. (Presented in full paper)
5 & 6. Two versions of ALife where simulated robots using range sensors, color sensors, and a differential drive, navi-

gate through 2d space, avoid obstacles, and gather energy particles [5]. (Presented in full paper)

3. RESULTS

Table-1 presents the average number of evaluations needed in 100 evolutionary runs to solve the DPB problem using the specified parameters.

WM%	Evo-Type	Mom.	Avg/StDiv/Suc.%
0%	memetic	on	941/433/100%
0%	memetic	off	1223/979/100%
85%	genetic	on	1528/3123/100%
85%	genetic	off	2687/10644/100%

Table 1: DPB benchmark results for genetic & memetic neuroevolution. *WM%* stands for Weight Mutation percentage, where GA algorithm had 85% of its mutation operators being weight perturbations, and MA algorithm has 0% of its mutation operators being weight perturbation, due to using weight perturbation in the local search phase only. *Evo-Type* stands evolutionary type, genetic or memetic in this case. *Mom.* flag notes whether momentum was turned on or off, and *Suc.* is the percentage of the evolutionary runs that were successful in generating a solution in under 20000 evaluations.

In the DPB experiment, we can see from Table-1 that the inclusion of momentum parameter improves the performance of the system drastically. On average, the number of needed evaluations to solve the problem decreases anywhere from 40% in the GA based system, to 25% in the MA versions. The GA based neuroevolutionary algorithm (such as NEAT for example) approach takes on average 2687 evaluations to solve the problem without momentum, and is significantly improved when momentum term is added, needing only 1528. The MA based system (such as DXNN [1] for example) solves the problem on average in roughly 1223 evaluations, and is further improved through the use of momentum to only needing 941 evaluations.

For the multi (2) DPB experiment, none of the runs were able to solve it completely, with the fitness score graph of the results presented in Fig-1. This benchmark tests how well the system can handle solving problems where we have to set multiple parameters to the right value ***simultaneously***. If only one of the sub-problems is solved (only 1 cart), the fitness does not improve since it's based on when the first (on either cart) pole falls below the 35 degree mark. Here too we see that momentum improves performance.

Though not shown in this short version of the paper, we have also performed benchmarks within the domain of visual discrimination, T-Maze navigation, and Evolutionary Robotics & ALife. We found the performance to be improved in all but the ALife benchmarks/problem domains.

4. DISCUSSION & CONCLUSION

In this paper we demonstrated the improvements in performance that a neuroevolutionary systems can achieve by incorporating the computationally inexpensive momentum parameter. We found that the momentum parameter improves performance to a significant degree in problems such

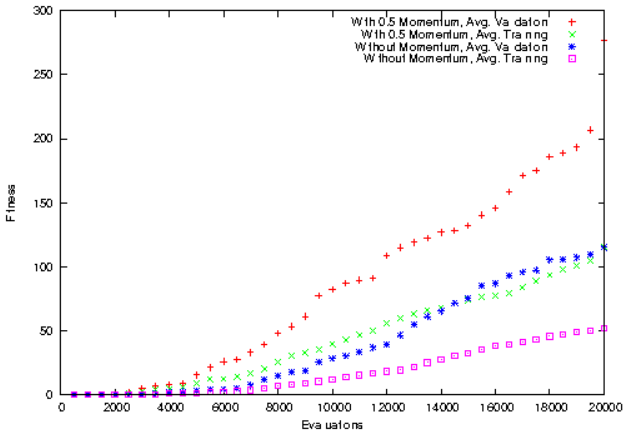

Figure 1: Multi-DPB: Avg. Max Fitness Vs. Evals.

as DPB, Multi-DPB, Visual Discrimination, and T-Maze navigation with plastic neurons. We also found that the momentum parameter enhanced neuroevolution did not produce statistically significant improvements in the ALife problem domains. But, *in no benchmark did we find the momentum parameter inclusion to decrease the systems' performance!* Based on these results, due to the momentum parameter's extremely low computational cost, we strongly believe that it should be become a standard part in all future neuroevolutionary approaches.

5. REFERENCES

[1] Sher, G. I. (2013). Handbook of Neuroevolution Through Erlang. Springer.

[2] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.

[3] Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, Machine Intelligence. Oliver and Boyd, Edinburgh, UK, 1968

[4] D. Whitley, S. Dominic, R. Das, and Charles W. Anderson. Genetic reinforcement learning for neurocontrol problems. Machine Learning, 13:259-284, 1993.

[5] Stanley, K., et al. "CPPNs Effectively Encode Fracture: A Response to Critical Factors in the Performance of HyperNEAT."

[6] L. Temby, P. Vamplew, A. Berry: Accelerating Real-Valued Genetic Algorithms Using Mutation-with-Momentum, 2005

[7] Berry, Adam, and P. Vamplew. PoD Can Mutate: A simple dynamic directed mutation approach for genetic algorithms. Diss. University of Tasmania, 2004.

[8] https://github.com/CorticalComputer

[9] Soltoggio, A., Bullinaria, J.A., Mattiussi, C., Durr, P., Floreano, D.: Evolutionary Advantages of Neuromodulated Plasticity in Dynamic, Reward-based Scenarios. In: Artificial Life XI, Cambridge, MA, MIT Press (2008) 569-576

[10] Blynel, J., Floreano, D.: Exploring the T-Maze: Evolving Learning-Like Robot Behaviors using CTRNNs. In: 2nd European Workshop on Evolutionary Robotics (EvoRob'2003). Lecture Notes in Computer Science (2003)