Novelty Producing Synaptic Plasticity

Anil Yaman∗
KAIST
Daejeon, Republic of Korea
a.yaman@tue.nl

Giovanni Iacca
University of Trento
Trento, Italy
giovanni.iacca@unitn.it

Decebal Constantin Mocanu†
Eindhoven University of Technology
Eindhoven, the Netherlands
d.c.mocanu@tue.nl

George Fletcher
Eindhoven University of Technology
Eindhoven, the Netherlands
g.h.l.fletcher@tue.nl

Mykola Pechenizkiy
Eindhoven University of Technology
Eindhoven, the Netherlands
m.pechenizkiy@tue.nl

ABSTRACT

A learning process with the plasticity property often requires reinforcement signals to guide the process. However, in some tasks (e.g., maze-navigation), it is very difficult to measure the performance of an agent to provide reinforcements, since the position of the goal is not known. This requires finding the correct behavior among a vast number of possible behaviors without having any feedback. In these cases, an exhaustive search may be needed. However, this might not be feasible especially when optimizing artificial neural networks in continuous domains. In this work, we introduce novelty producing synaptic plasticity (NPSP), where we evolve synaptic plasticity rules to produce as many novel behaviors as possible to find the behavior that can solve the problem. We evaluate the NPSP on deceptive maze environments that require the achievement of subgoals. Our results show that the proposed NPSP produces more novel behaviors compared to Random Search and Random Walk.

CCS CONCEPTS

• Theory of computation → Evolutionary algorithms;

KEYWORDS

Unsupervised learning, novelty, synaptic plasticity, neuro-evolution.

ACM Reference Format:
Anil Yaman, Giovanni Iacca, Decebal Constantin Mocanu, George Fletcher, and Mykola Pechenizkiy. 2020. Novelty Producing Synaptic Plasticity. In Genetic and Evolutionary Computation Conference Companion (GECCO ’20 Companion). July 8–12, 2020, Cancún, Mexico. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389976

1 INTRODUCTION

During a learning process, the fitness value of each behavior can be measured and used as reinforcement signal. For instance, in a maze-navigation task, the distance of the agent to the goal can be measured and used as reinforcement signal. For instance, in a maze-navigation task, the distance of the agent to the goal can be measured and used as reinforcement signal. For instance, in a maze-navigation task, the distance of the agent to the goal can be measured and used as reinforcement signal. For instance, in a maze-navigation task, the distance of the agent to the goal can be measured and used as reinforcement signal. For instance, in a maze-navigation task, the distance of the agent to the goal can be measured and used as reinforcement signal.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO ’20 Companion, July 8–12, 2020, Cancún, Mexico © 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7127-8/20/07... $15.00
https://doi.org/10.1145/3377929.3389976

We consider two environments consisting of $23 \times 23$ cells, that we refer to as deceptive maze (DM), shown in Figure 1. Each cell has five possibilities: empty, wall, goal, button, agent, color-coded in
white, black, blue, green, red respectively. Figures 1a-1b show two versions of the first environment, while Figures 1c-1d show two versions of the second environment. The difference between two versions of the same environment is the state (closed/opened) of the door in the middle wall. The behavior that solves the task requires the agent to go to the button area and perform a "press" action (which opens the door), then pass through the door and reach the goal.

To control the agents, we use recurrent neural networks (RNNs) without hidden layer (40 parameters in total) and with hidden layer (with 15, 30 and 50 hidden neurons, thus in total 420, 1290 and 3150 parameters), with 3 inputs (right, front, and left cells), and 5 outputs corresponding to one of the actions: stop, left, right, straight, press. Each input can sense if there is a wall or not.

We use a GA to evolve NPSP rules, which consist of 16 discrete genes, initialized randomly from \{-1, 0, 1\}, and up to 4 continuous genes, initialized randomly in \([0, 1]\). For each of the two environments, we consider two starting positions for three trials each. At the end of each trial, we compute a Novelty measure as the average number of novel behaviors (scaled in \([0, 1]\)). To calculate that, we record the behavior of an agent during each episode and find the average number of novel (unique) behaviors per trial. We abstract the behavior of an agent by recording its trajectory as a sequence of visited cells. Furthermore, we compute a second measure, Distance, that is the average of the smallest distances to the goal that an agent achieved during the episodes (scaled in \([0, 2]\)).

We set a population size of 14 and employ roulette wheel selection with 4 elites. We use 1-point crossover with a probability of 0.5 and a custom mutation operator which re-samples each discrete gene with a probability of 0.15 and performs a Gaussian perturbation \(N(0, 0.1)\) for the continuous genes. We run the evolutionary process for 100 generations. In each generation of the evolutionary process, we store the NPSP rules that produced the largest number of novel behaviors and achieved the minimum distance to the goal.

We compare the NPSP rules with Random Search (RS) and Random Walk (RW), which use a single solution to perform synaptic changes after every episode. All algorithms start with randomly initialized RNNs. At the end of an episode, we obtain the episodic performance as \(E_P = 1\) or \(E_P = 0\), which indicates that either the task is solved or not. If the task is not solved, we perform synaptic changes and test again the agent on the task. This process continues for a certain number of episodes or until the task is solved. In the case of RS, after each episode the network is re-initialized. In the case of RW, Gaussian perturbation is applied as: \(\sigma_j = \sigma_j + N(0, \sigma)\).

4 EXPERIMENTAL RESULTS

Table 1 shows the median novelty and distance of the agents trained by RS, RW and evolved NPSP rules. The columns "Goal" and "Second Room" report the number of times the agent reached the goal and entered into the second room. For all algorithms we run 12 trials (3 trials for 2 starting positions, for 2 environments), each consisting of 500 episodes of 250 action steps. The rows labelled as RS0H, RW0H and NPSP0H show the results obtained without hidden layer. We observe that the NPSP0H rules outperform both RS0H and RW0H in terms of distance and novelty. The rest of the rows shows the comparison results of RNNs with hidden layers. We observe that NPSP rules always outperform RS and RW, and that novelty increases with the number of hidden neurons.

5 CONCLUSIONS

In this work, we proposed novelty producing synaptic plasticity (NPSP), whose goal is to produce as many novel behaviors as possible and find the behavior that can solve a given problem. The NPSP performs synaptic changes based on the neuron activation traces (NATs), that store pairwise activations of neurons during an episode. Our results on maze-navigation tasks show that the use of NATs in NPSP leads to more novel behaviors w.r.t. RS and RW and helps solving the tasks. We also noted that the number of hidden neurons correlates positively with novelty, but not with the chance of reaching the goal, i.e. larger search spaces may produce more behaviors, but finding the ones leading to the goal becomes harder.

REFERENCES

[1] Wolfram Gerstner, Marco Lehmann, Vasiliki Liatoni, Dane Corneil, and Johanni Brea. 2018. Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of Hebbian Three-Factor Learning Rules. Frontiers in Neural Circuits 12 (2018), 53.
[2] Eduard Kuriscak, Petr Marsalek, Julius Strofek, and Peter G Toth. 2015. Biological context of Hebb learning in artificial neural networks, a review. Neurocomputing 152 (2015), 27–35.
[3] Jeff Orchard and Lin Wang. 2016. The evolution of a generalized neural learning rule. In Proceedings of the International Joint Conference on Neural Networks. IEEE, New York, NY, USA, 4688–4694.
[4] Sebastian Risi and Kenneth O Stanley. 2010. Indirectly encoding neural plasticity as a pattern of local rules. In Proceedings of the International Conference on Simulation of Adaptive Behavior. Springer, Berlin, Heidelberg, 533–543.
[5] Anil Yaman, Giovanni Iacca, Decebal Constantin Mocanu, George Fletcher, and Mykola Pechenizkiy. 2019. Learning with Delayed Synaptic Plasticity. In Proceedings of the Genetic and Evolutionary Computation Conference. Association for Computing Machinery, New York, NY, USA, 152–160.