Monte Carlo Tree Search for Optimizing Hyperparameters of Neural Network Training

Karolina Polanska, Wiktoria Dywan, Piotr Labuda, Leszek Koszalka, Iwona Pozniak-Koszalka, and Andrzej Kasprzak
Dept. of Systems and Computer Networks
Wroclaw University of Science and Technology
Wroclaw, Poland
Email: {209108, 218457, 218740}@student.pwr.edu.pl, {leszek.koszalka, iwona.pozniak-koszalka, andrzej.kasprzak}@pwr.edu.pl

Abstract—In tasks related to machine learning, the right selection of hyper-parameters can significantly impact training time and quality of the obtained results. Often, iterative search algorithms are used. In this paper, we propose an approach, based on our own modification of Monte Carlo Tree Search. The new algorithm is designed to work on discrete hyper-parameter spaces, and uses feedback from training process to learn and adjust its subsequent outputs. In the paper, the properties of the algorithm are studied, in particular for training Multilayer Perceptron. Moreover, three search algorithms are compared: Grid Search, Random Search and the proposed Monte Carlo Tree Search. As it is shown, the Monte Carlo Tree Search can give promising results and can be treated as fair competition to the off-shelf solutions.

Keywords-algorithm; Monte Carlo approach; Tree search; hyperparameter; neural network.

I. INTRODUCTION

In the recent decades, a lot of improvements were made in the area of known computing technologies, which had an essential impact on popularizing machine learning, leading to new, and more computationally complex algorithms being created [1]. Despite many advantages of machine learning as we know it nowadays, the high complexity of these methods translates to the time needed by a given model to learn what is desired; hence, a lot of attention given to developing the best way of automatically tuning hyper-parameters can be observed [2].

Hyper-parameters of a neural network are parameters of the learning process itself, such as learning rate, activation function, loss function or number of layers [3]. Their selection can significantly impact training time and results and therefore choosing hyper-parameters for neural network is an optimization problem [4][5]. There is a variety of available methods, for instance based on Bayesian approaches [6], or Sequential Model-based Algorithm Configurations (SMAC) [7]. Their performance varies with the type of network and chosen data. Monte Carlo Tree Search (MCTS) proposed in [8] is a heuristic search algorithm for decision processes; this method is often used in game play [9]. Notable example of usage is AlphaGo, an artificial intelligence application to play Go [10]. It is believed that using Monte Carlo Tree Search could bring satisfying results in hyper-parameter optimization process [8]. The main objective of this work is to improve Monte Carlo Tree Search algorithm so that it finds the best set of neural network hyper-parameters by executing the minimal amount of iterations and to compare the proposed method with two known algorithms, namely Grid Search and Random Search.

Grid Search searches the multidimensional grid of hyper-parameters by giving a trial to every node of the grid. This algorithm requires to manually specify the set of possible values for each parameter. The algorithm moves through the grid in iterative manner. This approach makes Grid Search suffer from the curse of dimensionality as the amount of nodes grows exponentially with the number of hyper-parameters [11].

Random Search is more effective in optimization for high dimensional spaces as it draws subsequent sets of parameters. For discrete parameter collection, Random Search moves over grid nodes, but, unlike Grid Search, in random order [12].

We introduce our method that involves Monte Carlo Tree Search to optimize hyper-parameters. Also, the proposed MCTS algorithm itself can be described along with the applied optimizations and method limitations.

The proposed experimentation system allows the comparison of MCTS with Grid Search and Random Search with regards to the obtained accuracy in subsequent trials. It was decided to focus on classification problems, particularly on Convolutional Neural Network (CNN), Multilayer Perceptron (MLP) and Support Vector Machine (SVM) [13], to confirm that MCTS algorithm can be applied to various machine learning techniques, not only neural networks.

All tests presented in this paper were conducted on Modified National Institute of Standards and Technology dataset (MNIST) [14], which is the biggest available collection of handwritten digits. It consists of about 60 000 samples in shape of matrices 28x28 pixels.

The rest of the paper is organized as follows. Section II contains a short review of important scientific papers in the area. In Section III, the problem is formulated. The core of the paper is Section IV with the presentation of the proposed

 ISBN: 978-1-61208-696-5

algorithm. The designed and implemented experimentation system is described in Section V. This section contains experiment design, the obtained results and comments. The conclusion and plans for further research appear in the last Section VI.

II. RELATED WORK

Traditionally, a manual search (meaning an approach based on empirical research) has been used for finding the most satisfying hyper-parameters [15]. While this approach can be enough for some researchers while training simple models, it still requires constant conscious management of chosen hyper-parameters values as even the slightest change in data used for learning can make them insufficient for achieving satisfactory results.

Several methods of automated choosing values of hyper-parameters were proposed over the years. One of the most common approaches is known as Grid Search, which looks for the best combination of parameters within whole space of previously defined fixed values, thus it can become time-consuming for a large space of potential solutions [16].

One of the most popular approaches, Random Search, is also one of the simplest. As suggested by its name, combinations of hyper-parameters values are chosen randomly until a satisfactory result of learning process is received. As presented in [12], Random Search can achieve the same results as Grid Search, but without the need to check every possible combination, i.e., it is relatively faster.

The Monte Carlo approach is applied to support solving many problems in artificial intelligence area [17], in particular in optimization of reinforcement learning process [18]. Very new and interesting review of applications of Monte Carlo Tree search can be found in [19].

III. PROBLEM STATEMENT

Given an artificial neural network N, with variable vector of hyper-parameters V, let a(V) be the accuracy of the vector V, defined as the highest accuracy reached by network N, trained with hyper-parameters V, among all the accuracies reached in a 10-fold cross-validation.

Let S be an algorithm searching through the possible space of hyper-parameter vectors V. During its operation, algorithm S produces the number of m hyper-parameter vectors. Accuracy of the algorithm A(S,-m) is defined as the highest a(Vi), where i = 1, 2,…, m.

As training a neural network can be a computationally expensive operation, the optimization task lies in finding an algorithm S such that A(S, m) is maximized, while m is minimized at the same time.

IV. PROPOSED METHOD AND ALGORITHM

The proposed method of exploring the hyper-parameter space is based on a modified Monte Carlo Tree Search approach. We introduced several changes that allowed the approach to be used for exploring a discrete hyper-parameter space.

A. Building the tree

For each neural network, the hyper-parameter space to be explored is defined as a discrete set of possible values for each of hyper-parameters taken into consideration. When transforming the space into a tree data structure, the following approach was used:

First, hyper-parameters are ordered according to their number of possible values, from lowest to highest. The ordered list of hyper-parameters is marked as HS.

The root node of the tree represents the beginning of the decision process. For each possible value of first hyper-parameter in the list HS, a child node is added to the root node, representing the choice of that value for a given hyper-parameter. Then, for each possible value of the second hyper-parameter in the list HS, a child node is added to all of the level 2 nodes. The process repeats itself until there are no more hyper-parameters on the list HS to further expand the tree.

The resulting tree has every possible combination of chosen values represented as a leaf node, and represents the whole space of hyper-parameters as a multi-staged decision process.

B. The algorithm

The modified version of the MCTS algorithm follows a standard model:
Selection - Expansion - Simulation - Backpropagation.

Each node (except the root) in the tree has a value representing expected accuracy of a neural network trained using hyper-parameters represented by leaf descendants of a given node. This value is assigned and updated by the MCTS algorithm during its operation.

a) Selection: As long as the node the algorithm is in has children nodes of known value, the algorithm chooses a node of highest value and moves to it.

b) Expansion: If the node has no children of known value, a node is chosen at random for the Simulation phase.

c) Simulation: To complete the set of hyper-parameter values the algorithm chooses remaining values at random. A neural network of choice is constructed and trained using this set of hyper-parameters, and its accuracy, measured as a result of 10 k-fold cross-validation, is assigned as a value of the node the algorithm started from.

d) Backpropagation: After all child nodes created during expansion phase are assigned a value, the value of their parent node is updated to the mean of their values. The process propagates recursively, updating the parent nodes value until the root is reached.

C. Optimisation

As the search algorithms are compared on what was their best proposed solution after N trials (one trial being equal to one 10 k-fold validation of a given set of hyper-parameters), three optimizations were introduced to maximize the algorithms performance within the three allowed trials described below.

 ISBN: 978-1-61208-696-5

a) Hyper-parameter sorting: As described in Section IV-A, hyper-parameters are ordered from the one with least possible values to the one with the most. The algorithm will, therefore, always have to start by expanding the root node of the tree. If the initial expansion requires a small amount of simulations, the algorithm can start making more informed decisions sooner, relying less on random choice and more on past values of the nodes.

b) Node exclusion: The algorithm can find itself in a local maximum of accuracy when the entire set of hyper-parameters is chosen based on node values and not random chance. In that case the leaf node of this solution is marked as excluded and cannot be included in any following set of hyper-parameters. This forces the algorithm to consider other options, and due to backpropagation will gradually push the algorithm to the global maximum as the number of generated solutions tends to infinity.

c) Result caching: Due to random choice, it might occur that an algorithm generates the same solution multiple times. For that reason, all previously generated solutions and their respective accuracy are stored, and can be used to substitute the training process when such case occurs. Since this does not count as a generated solution, the MCTS can generate overall better results within a given solutions limit.

D. Limitation

Due to its nature, MCTS is significantly harder to execute in parallel than Random Search or Grid Search. Due to the fact that next set of hyper-parameters is known only after training the network on previous sets, the execution has to be done sequentially. Only during the expansion phase several sets of hyper-parameters to test are known in advance and the order of their testing does not matter. Alternatively, the algorithm could be modified to test several promising nodes of the tree at once - however, this hypothetical method is beyond the scope of this paper and poses several questions about potentially redundant work.

V. EXPERIMENTATION SYSTEM

The main goal of the research was checking whether the proposed method of exploring hyper-parameter space is useful while solving the classification task. In addition, the results of comparative research aimed in comparison of the accuracy of known algorithms and the accuracy of a new algorithm based on Monte Carlo tree search are presented.

A. Experiment Design

In order to conduct research, the Python application has been created. Its components have been written with usage of Keras, TensorFlow and Scikit libraries [13][20].

The classification of the MNIST dataset [14] has been chosen as the task to train the network on. It is a popular set of handwritten letters and digits represented as bitmap. It was chosen with regards to its size of around 60 000 samples. A Multilayer Perceptron classifier was trained, and several parameters of the training process and network's structure were chosen as the hyper-parameter space to compare the search algorithms. Table I presents the hyper-parameters and their values used in experiments.

TABLE I. HYPERPARAMETERS

Hyper-parameter	Possible Values
Learning rate	0.1, 0.01, 0.001
Activation function	tanh, relu, sigmoid
Hidden layer units	1, 30, 100, 800
First dropout	0.1, 0.25, 0.7, 0.9
Second dropout	0.1, 0.3, 0.5, 0.9

Three search algorithms, Random Search, Grid Search, and our own, were tested at finding the best set of hyper-parameters in as few guesses as possible.

B. Results

The obtained results of the number of sixteen experiments, conducted on Multilayer Perceptron, are shown: in Figure 1, produced by Grid Search, in Figure 2, produced by Random Search, and in Figure 3, produced by our Monte Carlo Tree Search.

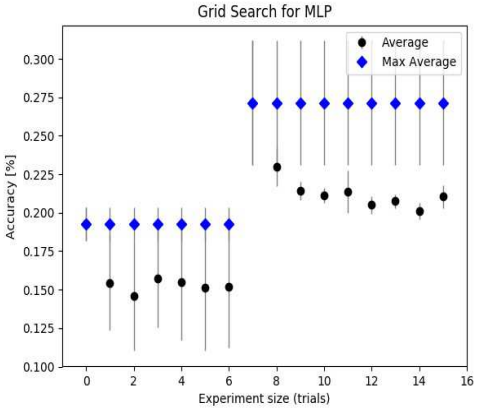

Figure 1. Results of the experiment with Grid Search.

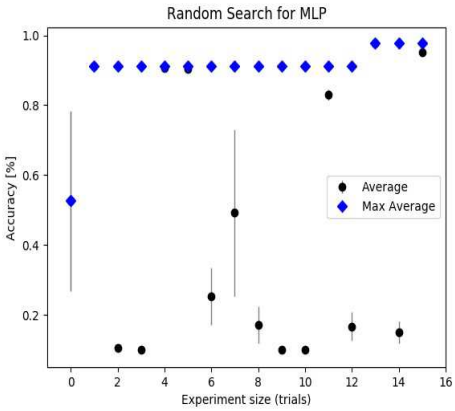

Figure 2. Results of the experiment with Random Search.

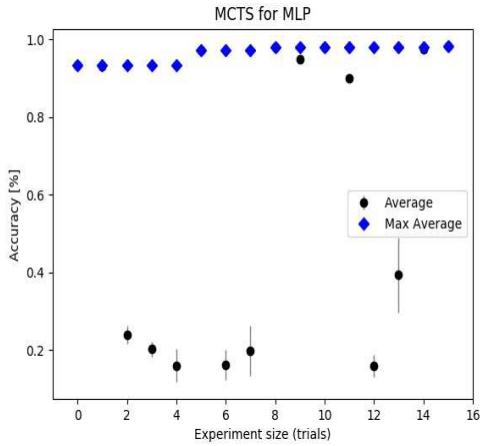

Figure 3. Results of the experiment with Monte Carlo Tree Search.

C. Comments

As shown in Figure 1 and Figure 2, the known search algorithms are susceptible to a sudden spike of precision due to accidental finding of a good solution. Contrary, as it can be observed in Figure 3, our Monte Carlo Tree Search algorithm is able to make small incremental changes from session to session.

VI. CONCLUSION

The presented approach to exploring hyper-parameters space, in particular the proposed Monte Carlo Tree Search algorithm can be considered as an interesting alternative for the off-shelf solutions. Its computational overhead is significantly higher than in the case of Grid Search or Random Search but negligible compared to the typical task of training artificial neural networks.

In the near future, we plan to conduct more research using the proposed approach on Convolutional Neural Network and Support Vector Machine. Also, we are in the process of some improvement consisting in implementation of the multistage experimentation system along with the rules described in [21].

ACKNOWLEDGMENT

This work was supported by the statutory funds of the Department of Systems and Computer Networks under grant no. 0401/0132/18, Faculty of Electronics, Wroclaw University of Science and Technology, Wroclaw, Poland.

REFERENCES

[1] G. Montavon, G. Orr, and K-R. Muller, “Neural Networks: Tricks of the Trade,” LNCS, vol. 7700, Springer, 2012.

[2] W. Koehrsen, “Automated Machine Learning Hyper-Parameter Tuning in Python,” https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a [retrieved: January 2019].

[3] A. Honchar, “Neural Networks for Algorithmic Trading. Hyper-Parameters Optimization,” https:/medium.com/@alexrachnog/neural-networks-for-algorithmic-trading-hyperparameters-optimization-cb2b [retrieved: January 2019].

[4] R. Bardenet, M. Brendel, B. Kegl, and M. Sebag, “Collaborative Hyper-Parameter Tuning,” Proc. of ICML, 2013, pp. 199-207.

[5] P. Sharma, “Improving Neural Networks – Hyper-Parameter Tuning„” analyticsvidhya.com/blog/2018/11/neural-networks-hyperparameter-tuning-regularization-deeplearning [retrieved: January 2019].

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, vol. 15, 2014, pp. 1929-1958.

[7] F. Huffer, H. H. Hoos, and K. Leyton-Brown, “Sequential model Based Optimization for General Algorithm Configuration,” Learning and Intell. Optim., Springer, 2011, pp. 507-523.

[8] G. M. J-B. Chaslot, “ Monte Carlo Tree Search,” PhD Thesis, Maastricht University, 2010, Dissertation Series No. 2010-41, ISBN 978-90-8559-099-6.

[9] T. Cazenave and N. Jouandeau, “A Parallel Monte Carlo Tree Search Algorithm,” Proc. Comput. And Games, LNCS, vol. 5131, 2008, pp. 72-80.

[10] H. Bajer and M. H. Winands, “Beam Monte Carlo Tree Search” Proc. IEEE Conf. Comput. Intell. Games, 2012, pp. 227-233.

[11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, “Algorithms for Hyper-Parameter Optimization,” Advances in Neural Information Processing Systems 24 (NIPS 2011), pp.2546-2554.

[12] J. Bergstra and Y. Bengio, “Random Search for Hyper-parameter Optimization," Journal of Machine Learning Research, vol. 13(1), 2012, pp. 281-305.

[13] S. Raschka, “Python Machine Learning,” Packt Publishing, 2016.

[14] Yann.lecun.com/exdb/mnist, [retrieved: November 2018].

[15] P. Koch, B. Wujek, O. Golowidov, and S. Gardner, ”Automated Hyper-parameter Tuning for Effective Machine Learning,” 2017, SAS514-2017.

[16] S. Li and M. Tan, “Tuning SVM Parameters by Using-Hybrid CLPSO-BFGS Algorithm,” Journal of Neurocomputing, vol. 73, June 2010, pp. 2089-2096.

[17] H. Akiyama, K. Komiya, and Y. Kotani, “Nested Monte Carlo Search with AMAF Heuristic,” Proc. Int. Conf. Tech. Applica. Artif. Intell, 2010, pp. 172-176.

[18] J. Asmuth and M. L. Littman, “Learning Planning Near Bayes Optimal Reinforcement Learning via Monte Carlo Search,” Proc. Conf. Uncert. Artif. Intell., 2011, pp.19-26.

[19] J. N. van Rijn and F. Hutter, “Hyper-parameter Importance Across Datasets,” Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2367-2376.

[20] Stackshare.io/stackups/keras-vs-scikit-learn-vs-tensorflow [retrieved: January 2019].

[21] M. Hudziak, I. Pozniak-Koszalka, L Koszalka, and A. Kasprzak, “Multiagent Pathfinding in the Crowded Environment with Obstacles,” Journal of Intelligent and Fuzzy Systems, vol. 32(2), 2017, pp. 1561-1573.

 ISBN: 978-1-61208-696-5