Auto-tuning of Hyperparameters of Machine Learning Models

Zhen Wang
Tohoku University
Sendai, Miyagi, Japan
wangzhen@dc.tohoku.ac.jp

Ryusuke Egawa
Tohoku University
Sendai, Miyagi, Japan
egawa@tohoku.ac.jp

Reiji Suda
The University of Tokyo
Tokyo, Japan
reiji@is.s.u-tokyo.ac.jp

Hiroyuki Takizawa
Tohoku University
Sendai, Miyagi, Japan
takizawa@tohoku.ac.jp

ABSTRACT

Most machine learning models use hyperparameters empirically defined in advance of their training processes. Even a classic machine learning model, so-called multilayer perceptrons, has a lot of hyperparameters. In the case of using such a model for a classification problem, one difficulty is that the achieved classification accuracy could drastically change every time even if the same hyperparameter values are used. Hence, it is challenging to determine an appropriate hyperparameter configuration at a low cost. The same problem has so far been discussed in the research field of software performance automatic tuning, or auto-tuning. Therefore, in this work, we employ one of such auto-tuning mechanisms, ATMathCoreLib, for auto-tuning hyperparameters of machine learning models, and discuss the feasibility of using such technologies in the field of machine learning.

KEYWORDS

Machine Learning, Hyperparameter, Auto-tuning

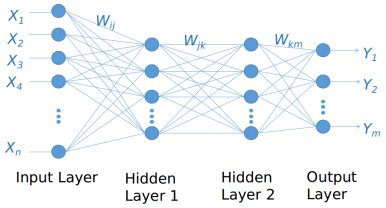

Figure 1: A Multilayer Perceptron

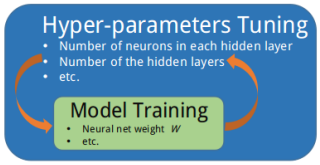

Figure 2: Hyper-parameters Tuning VS Model Training in the MLP

1 INTRODUCTION

Recently, machine learning has been widely used for various real-world problems, such as image recognition problems[1], whose solutions cannot easily be described as clear algorithms. One of popular machine learning algorithms is called a multilayer perceptron (MLP), which is a multi-layered network of neurons, as shown in Figure 1. Once a network configuration is given, the MLP can adjust parameters of the network, such as weight in each neuron, and the parameter adjustment is called a model training process. During the model training process, a lot of pairs of input and output data, called training data, are presented to the MLP. Then, all parameters of the MLP are adjusted so that the input-output relation of the MLP approaches that of training data. Therefore, without defining an explicit algorithm, the MLP can build any input-output relation from a lot of training data and solve a given problem.

One practical problem in using the MLP is that all hyperparameter values must be defined in advance of the model training process, as shown in Figure 2, and there is no established way of properly determining those hyperparameters. Representative examples of hyperparameters are related to the network configuration, such as the number of hidden layers and the number of neurons in each hidden layer. Currently, in practical use, hyperparameters are manually determined in a try-and-error fashion. However, hyperparameters' tuning of learning algorithms is a manual and time-consuming process[3]. Therefore, this work focuses on automation of finding appropriate values of hyperparameters.

There are three common approaches to find good values of hyperparameters. The first approach is called full search. In full search, a hyperparameter space is discretized, and an MLP model is trained and assessed for all combinations of the values over the space. Although this method will lead to the best hyperparameter configuration, it is too costly because the cost grows exponentially with the number of hyperparameters. The second approach is to train and assess many candidate models, each of which is built with a random combination of hyperparameter values. The efficiency of this approach strongly depends on how to decide random combinations of hyperparameter values. The third approach, Latin hypercube sampling, is similar to the random search method but more structured. In Latin hypercube sampling, candidate values are sampled exactly uniform across each hyperparameter but randomly combined. Even though this method can ensure that sampling candidate values are approximately equidistant from one another, we have to decide heuristically the range of sampling values.

HPC Asia, January 2018, Tokyo, Japan

Even if a supercomputer or a high-performance computing (HPC) system is available, the first approach, full search of a hyperparameter space, is too time-consuming, and does not end within a practical time, while the second and the third ones not always can find the best values for hyperparameters and need some heuristic decision.

Furthermore, some optimization methods, such as Bayesian optimization[4], have been proved that they can find better hyperparameters significantly faster than a human expert. However, this process is not automated and the users still need some knowledge about Bayesian optimization before they utilize this method. Hence, this work focuses on the Bayesian optimization, and automates the hyperparameter setting. Specifically, an auto-tuning mechanism called ATMathCoreLib is employed for the automation, and the future direction of this work are discussed.

2 PROBLEM DEFINITION

All hyperparameters such as the number of neurons in each hidden layer need to be determined in advance of training an MLP model. However, there is no systematic way of finding appropriate values of hyperparameters to train an MLP model for a given problem. Thus, those hyperparameters are empirically determined, e.g., by human experts. A problem tackled in this work is that it is time-consuming to assess the performance of a model with a given configuration of hyperparameters. Although a large-scale HPC system can assess a lot of hyperparameter configurations in parallel, the assessment itself is too time-consuming for assessing all possible combinations of hyperparameter values. Therefore, this work explores an effective way to intelligently search a hyperparameter space in addition to parallel parameter search assuming a large-scale HPC system. The goal of this work is to find a better hyperparameter configuration automatically within a shorter time.

3 PROPOSED APPROACH

ATMathCoreLib[5] is an auto-tuning mechanism originally developed for improving the performance of a program by adjusting its parameters affecting the execution time. To be more specific, it can tell which parameter to choose for next experiment using the previous experiment information. ATMathCoreLib can adjust a lot of parameters based on the Bayesian optimization under the situation where the performance of a model defined by the parameters has an uncertainty, i.e., the observed performance may have some perturbations. This difficulty is exactly the same as that in the hyperparameter search in machine learning. However, in this work, ATMathCoreLib is employed for the auto-tuning of hyperparameters so as to maximize the performance of a machine learning model.

There are two auto-tuning modes in ATMathCoreLib: offline auto-tuning mode and online auto-tuning mode. Offline auto-tuning mode is a two-phase method. In the first phase, many trials are executed with the sample data in order to search for the parameters with best model performance. In the second phase, the chosen parameters are used for practical executions. On the other hand, in the online auto-tuning method, no trial execution is performed and the candidate values of parameters are evaluated directly in the practical executions.

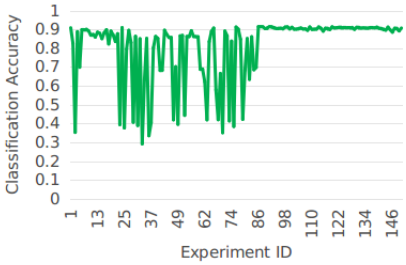

Figure 3: The accuracy of the trained model

In machine learning, the tuning process of hyperparameters is different from the model training process; the former one should be finished before the later one. Moreover, since the performance of a machine learning model has some perturbations, we need to assess the performance of a certain parameter configuration N times (N trials) to obtain the average performance. Thus, if there are M parameter configurations, one simple way is to repeat training and assessing a model $N \times M$ times, i.e., $N \times M$ trials, to decide the best parameter configuration in terms of the average performance. On the other hand, ATMathCoreLib uses a linear statistical model to decide an appropriate parameter configuration in a statistically-reliable way with fewer trials.

Thus, in this work, we employ ATMathCoreLib with offline auto-tuning mode for auto-tuning hyperparameters of machine learning models, and discuss the feasibility of using such technologies in the field of machine learning.

4 EVALUATIONS

In this work, auto-tuning of hyperparameters using ATMathCoreLib is preliminarily evaluated to show the importance of intelligent hyperparameter search and feasibility of the proposed method. A well-known data set of handwritten digits, called the MNIST2 data set, is used as the training data set given for an MLP model. The MNIST data set consists of 60,000 examples for training, and 10,000 examples for testing, and Tensorflow[6] is used for training an MLP model. The execution is evaluated using a PC of Intel Core i7-3770 CPU of eight cores running at 3.4 GHz.

In the preliminary evaluation, an MLP model has two hidden layers, and the number of neurons in each hidden layer is adjusted using ATMathCoreLib. That is, the number of hyperparameters adjusted in the evaluation is two. ATMathCoreLib is used with the offline mode with no user model, where ATMathCoreLib automatically constructs a simple model with mean and variance for the performance of the hyperparameters. All parameters, not hyperparameters, are iteratively adjusted during the training process. The number of neurons in each hidden layer is an important hyperparameter that affects the performance of the trained MLP model in terms of classification accuracy.

A pair of training a model with a certain hyperparameter configuration and assessing its classification accuracy is called an experiment, and a unique ID is given to each experiment and hyperparameter configuration. Figure 3 shows that the classification accuracy changes by repeating the experiment. It changes significantly until the 100-th experiment. However, after that, it just fluctuates slightly

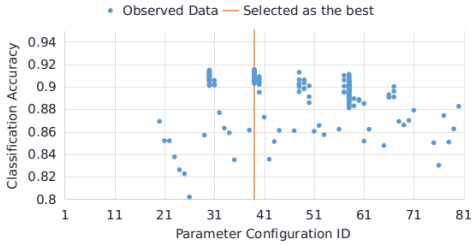

Figure 4: The hyperparameter configuration distribution

near the best classification accuracy. These results show that, after a small number of running trials, only the hyperparameter configurations with higher classification accuracies will be run until the best one can be found with the proposed method.

Figure 4 shows the relationship between the configuration ID and the observed classification accuracy. The classification accuracy obviously changes whenever an MLP model is trained even with exactly the same hyperparameter configuration. Therefore, the MLP model should be trained multiple times to determine appropriate values of hyperparameters in terms of the average classification accuracy. Thus, hyperparameter configurations with high classification accuracies are tested more times than the others. As a result, in a statistically reliable way, the proposed approach can select an appropriate hyperparameter configuration indicated by the red line in the figure. Moreover, the results also indicate that the number of appropriate hyperparameter configurations is limited, and hence conventional full search and random search approaches to finding an appropriate hyperparameter configuration are inefficient and time-consuming.

Figures 3 and 4 indicate that only promising parameter configurations are repeatedly tested to make sure which is the best one. That is, ATMathCoreLib can efficiently find an appropriate parameter configuration. Accordingly, it is concluded that a smart hyperparameter search method such as the method implemented in ATMathCoreLib is necessary to achieve efficient hyperparameter search.

5 CONCLUSION AND FUTURE WORK

In this work, we have used one of auto-tuning mechanism, ATMathCoreLib, for auto-tuning hyperparameters of a simple MLP model. The preliminary evaluation results show that auto-tuning technologies developed in the HPC field are also effective to intelligently adjust the hyperparameters of machine learning models.

In our future work, we will use more advanced machine learning models, such as convolution neural network, with larger hyperparameter spaces, and discuss the effectiveness of auto-tuning in more practical situations.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
[2] Cortes Corinna LeCun Yann. 2017. The MNIST Database of Handwritten Digits. (2017). http://yann.lecun.com/exdb/mnist/
[3] Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*. 2113–2122.
[4] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.
[5] Reiji Suda. 2011. A Bayesian method of online automatic tuning. In *Software Automatic Tuning*. Springer, 275–293.
[6] Tensorflow. 2017. An open-source software library for Machine Intelligence. (2017). https://www.tensorflow.org/