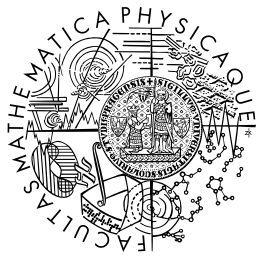

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jiří Krejčí

Efficient hyper-parameter optimization

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis:	Mgr. Martin Pilát, Ph.D.
Study programme:	Computer Science - Artificial Intelligence
Study branch:	IUIP

Prague 2024

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication. It is nice to say thanks to supervisors, friends, family, book authors and food providers.

Title: Efficient hyper-parameter optimization

Author: Jiří Krejčí

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Abstracts are an abstract form of art. Use the most precise, shortest sentences that state what problem the thesis addresses, how it is approached, pinpoint the exact result achieved, and describe the applications and significance of the results. Highlight anything novel that was discovered or improved by the thesis. Maximum length is 200 words, but try to fit into 120. Abstracts are often used for deciding if a reviewer will be suitable for the thesis; a well-written abstract thus increases the probability of getting a reviewer who will like the thesis.

Keywords: key words

Contents

Introduction

In this thesis, we are addressing the problem of hyperparameter tuning of deep neural networks. The most common way to tune hyperparameters still to this day is by hand, by trial and error, relying on previous experience and various rules of thumb. It is very time-consuming for people and requires a great deal of expertise to do efficiently and well. With the steady increase in computing power, it is preferable to offload this task to computers. The right algorithm can do the task more thoroughly and without much human effort.

Even though this is an extensively studied problem, there still is no consensus on which method performs best. In this thesis, we want to expand on the knowledge of these algorithms and conduct experiments in the healthcare domain.

The main issue is that the function we optimize is expensive to evaluate and the range of possible input values is vast. One function evaluation, or training the network, can take hours or days. That is why the classical methods for hyperparameter optimization in machine learning such as grid search are not applicable to more complex deep learning architectures. Therefore, more sophisticated methods and algorithms were developed and are still being researched. The most important metric in this field is the efficiency of the search.

Hyperparameter optimization has roots in black-box optimization techniques. The machine learning models act as a black-box system to an extent. In deep learning, there is some potential to exploit the knowledge of the system, but only to a small degree. Many tools and optimization frameworks exist for hyperparameter tuning. Most of them are even open source, such as Optuna [1] or SMAC3 [2]. These tuning tools usually use Bayesian optimization techniques internally. Bayesian optimization is well suited for global optimization of black-box, expensive-to-evaluate functions.

TODO: Decide on the focus of the thesis. Below are some options that came to my mind. In this thesis:

- We survey the literature (has been done recently [3][4]) and we compare the HPO methods on some machine learning tasks in the domain of health care *if the data are reasonable.

- We focus on the problem of distribution shift when new training data arrive in the process of developing the network. More specifically, we study how the optimal hyperparameters change when a new batch of data arrives. Notes:

    1. How do we study this, when datasets are complete? Metadata? Would it be a valid approximation of the real case?
    2. Do we think the results are going to be interesting? I would suppose that the hyperparameters stay the same or change very slightly.

- Focus on low-budget search. Research and experiment with how to best spend a limited budget. May be some search strategy random search + Bayesian optimization, or a multi-fidelity approach. There is a lot of literature on early stopping.

- We focus on transfer learning.

  Note: I'm not sure if it is a viable option anymore. The papers seem to be quite advanced to me and build on deeper knowledge of the HPO methods.

  We are interested in making the best use of previous experiments so that we can efficiently iteratively optimize the model. Also applicable for the training data arriving in batches as we develop the model. Transfer learning in HPO is also an established method. So again, we could compare some methods?

Chapter 1

Background

Let A be a machine learning algorithm with hyperparameters $\lambda_1, \ldots, \lambda_n$ with domains $\Lambda_1, \ldots, \Lambda_n$. Let $\boldsymbol{\Lambda} = \Lambda_1 \times \cdot \times \Lambda_n$ denote its hyperparameter space. Hyperparameters can be continuous, integer-valued, or categorical. Also, we can have conditional hyperparameters. We say that hyperparameter λ_i is *conditional* on another hyperparameter λ_j, if λ_i is only active if hyperparameter λ_j takes values from a given set $V_i(j) \subset \Lambda_j$

For each hyperparameter configuration or setting $\lambda \in \boldsymbol{\Lambda}$, we denote A_λ the learning algorithm A using this hyperparameter setting. Let $\mathscr{L}(A_\Lambda, \mathscr{D}_{train}, \mathscr{D}_{valid})$ denote the validation loss of algorithm A_λ on data $\mathscr{D}_{valid}$ when trained on $\mathscr{D}_{train}$.

Definition 1 (Hyperparameter optimization problem)**.** *The hyperparameter optimization problem is to find hyperparameters λ^* that minimize the loss function*

$$\lambda^* = \arg\min_{\lambda} f(\lambda) = \arg\min_{\lambda} \mathscr{L}(A_\lambda, \mathscr{D}_{train}, \mathscr{D}_{valid}).$$

Several properties make the problem hard to solve.

- It is hard to obtain derivatives of the loss function with respect to hyperparameters and we will not use them to find the optimal solution. Such an optimization problem is called a black-box optimization in literature.
- Each function evaluation is expensive. Fully training a single deep neural network can take days.
- Each function evaluation may require a variable amount of time. For example, training larger models (e.g. more artificial neurons) takes more time to train. Therefore, the hyperparameter optimization algorithm should take training time into account.

- Observations are noisy. Repeated training may result in models that vary in performance since it is common to use random initialization of weights. The training process itself may not be deterministic as well. For example, if we use mini-batch shuffling.

On the other hand, we can leverage parallel computation to run multiple trials at the same time. One additional benefit of solving the optimization problem limited to deep neural networks is that we have access to intermediate results.

In this thesis, we assume that the general architecture of the neural network is already given and hyperparameters can change only smaller aspects, such as the number of neurons in a layer, or kernel size in a convolutional layer. For literature dealing with the more general problem, please refer to Neural Architecture Search (NAS).

For anyone interested in the process of hyperparameter tuning and how it might be done in practice, we recommend the Deep learning tuning playbook [5]. The authors give valuable insights into practical aspects of hyperparameter tuning that they have collected over more than ten years of working in deep learning. These insights are rarely documented. As the authors state in the text, they could not find any comprehensive attempt to explain how to get good results with deep learning. More importantly for this thesis, it gives us insight into how experts might do hyperparameter tuning. The text reveals that even today, advanced hyperparameter tuning tools are not the ultimate solution to the problem. Instead, they recommend how to use them smartly. They propose that there is still a human expert guiding the search, at least in the first, exploratory, phase.

1.1 Basic approaches

Before we get to algorithmic approaches, let us consider manual hyperparameter tuning. We cannot be surprised that people still tune hyperparameters manually. There is no technical overhead or barrier. Also, in the process of hyperparameter tuning, we gain insight into the problem, which might allow us to improve our solution in ways that are not achievable just by hyperparameter tuning. Nevertheless, there are clear limits to manual tuning so let us dive into the automated approaches. The traditional algorithms for hyperparameter optimization are grid search and random search. These algorithms are simple and still widely used.

Grid Search Grid search performs an exhaustive search through a manually specified subset of the hyperparameter space. Grid search is best used when the number of hyperparameters is small, or the function evaluation is not that expensive. Its biggest drawback is that the number of configurations to evaluate

grows exponentially with the number of hyperparameters. Therefore, it is best to determine which hyperparameters are the most important and limit the search only to this subset. If we did not do this, we would waste a lot of computational power on hyperparameter combinations, where only the unimportant hyperparameter changes, but the important ones stay the same. But how do we determine which hyperparameters are important and we need to tune them together? On the other hand, grid search is easily parallelizable. That is an enormous advantage since in real-world scenarios, it is not uncommon to have access to a computing cluster.

Random Search Random search is often used in the HPO literature as the baseline method for more advanced algorithms. In real-world optimization problems, random search often works better than the grid search. Bergstra et al. [6] compared random search to grid search and found that randomly chosen trials are more efficient for hyperparameter optimization than trials on a grid. It is possible to encounter a random search with 2X-budget as a baseline in some research papers. It is just a random search with two times the budget of other methods in comparison. As Li et al. [7] show, 2X-budget random search provides a strong baseline.

Quasi-random search If our budget is low then quasi-random search might be the better option. It works by generating a low-discrepancy sequence. Intuitively, a low-discrepancy sequence covers the whole domain evenly. Therefore, the search space is better covered even with a small number of samples. In the Deep learning tuning playbook, the authors recommend using quasi-random search over grid search and random search for the initial exploration of the hyperparameter space.

1.2 Bayesian optimization

So far, we have seen model-less approaches, where each trial is independent. This approach offers some advantages, like parallelization and simplicity of implementation, but it is quite inefficient. Information obtained from previous trials is not used in any way to guide the search. Bayesian optimization methods build and use an internal model of the learning algorithm's generalization performance. Bayesian optimization is widely covered in literature, we will use the work of Brochu et al. [8] and Frazier [9] for the definitions and description of the method.

The basic loop of a Bayesian optimization algorithm is simple. It uses the internal model to get a suggestion of the next hyperparameter configuration to try. Then it trains the neural network using the suggested configuration and uses

the resulting performance metric to update the model. We repeat this process until we run out of budget or the neural network performs well enough.

In general, Bayesian optimization is a class of optimization methods focused on optimizing a real-valued objective function

$$\max_{x \in A} f(x).$$

Since we have defined the hyperparameter optimization problem as minimization of the loss function, we can assume that f is defined as $f(\lambda) = -\mathscr{L}(\lambda)$. Maximizing this function is equivalent to minimizing the original function. Also, not all hyperparameters are real-valued but we will address that later.

We assume that the objective is Lipschitz-continuous. That is, there exists some constant C such that for all $x_1, x_2 \in A : \|f(x_1) - f(x_2)\| \leq C\|x_1 - x_2\|$. The constant C may be unknown. The objective is commonly a black-box function, which in our case is true. We also assume that the search space is bounded in all dimensions.

The method got its name from the Bayes' theorem.

Practical BO of ML algorithms. They show how different kernels affect performance and describe algorithms that take into account the variable cost (duration) of learning algorithm experiments [10].

1.2.1 Gaussian process

"For continuous functions, Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian process and maintains a posterior distribution for this function as observations are made or, in our case, as the results of running learning algorithm experiments with different hyperparameters are observed (Practical Bayesian optimization 2012)" Good overview of Gaussian processes: [8].

1.2.2 Parzen-Tree Estimator

Algorithms for hyperparameter optimization [11] introduces TPE with Expected improvement.

1.3 Multi-fidelity approaches

Supervising the multi-fidelity race of hyperparameter configurations [12] - Gaussian process kernel that allows for comparison of networks trained at different budgets.

Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves [13].

Freeze-thaw Bayesian optimization [14].

1.4 Transfer learning

One advantage that an experienced practitioner will have over a classical HPO algorithm is that he will be good at generalizing and estimating good hyperparameter configurations across similar learning problems. The algorithm either depends on good bounds given by a user for efficient search, or it has to try a lot of configurations that do not perform well at all to find the bounds itself. The main idea of transfer learning is to use the experience from previous trials and similar problems in a new trial. This target function estimate should guide the search until the model is refined by new trials.

Collaborative hyperparameter tuning [15].

Efficient transfer learning method for automatic hyperparameter tuning [16].

Scalable hyperparameter transfer learning [17].

Pre-trained Gaussian processes for Bayesian optimization [18].

Chapter 2

More complicated chapter

2.1 Datasets

Chapter 3

Results and discussion

Conclusion

Bibliography

[1] Takuya Akiba et al. "Optuna: A Next-Generation Hyperparameter Optimization Framework". In: *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2623–2631.

[2] Marius Lindauer et al. "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization". In: *Journal of Machine Learning Research* 23.54 (2022), pp. 1–9. URL: http://jmlr.org/papers/v23/21-0888.html.

[3] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316.

[4] Bernd Bischl et al. "Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. arXiv 2021". In: *arXiv preprint arXiv:2107.05847* ().

[5] Varun Godbole et al. *Deep Learning Tuning Playbook*. Version 1.0. 2023. URL: http://github.com/google-research/tuning_playbook.

[6] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2 (2012).

[7] Lisha Li et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *Journal of Machine Learning Research* 18.185 (2018), pp. 1–52.

[8] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[9] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[10] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[11] James Bergstra et al. “Algorithms for hyper-parameter optimization”. In: *Advances in neural information processing systems* 24 (2011).

[12] Martin Wistuba, Arlind Kadra, and Josif Grabocka. “Supervising the multi-fidelity race of hyperparameter configurations”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 13470–13484.

[13] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. “Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves”. In: *Twenty-fourth international joint conference on artificial intelligence*. 2015.

[14] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. “Freeze-thaw Bayesian optimization”. In: *arXiv preprint arXiv:1406.3896* (2014).

[15] Rémi Bardenet et al. “Collaborative hyperparameter tuning”. In: *International conference on machine learning*. PMLR. 2013, pp. 199–207.

[16] Dani Yogatama and Gideon Mann. “Efficient transfer learning method for automatic hyperparameter tuning”. In: *Artificial intelligence and statistics*. PMLR. 2014, pp. 1077–1085.

[17] Valerio Perrone et al. “Scalable hyperparameter transfer learning”. In: *Advances in neural information processing systems* 31 (2018).

[18] Zi Wang et al. “Pre-trained Gaussian processes for Bayesian optimization”. In: *arXiv preprint arXiv:2109.08215* (2021).