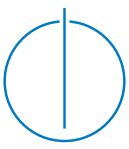
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Neural Network Hyperparameter Optimization with Sparse Grids

Maximilian Michallik



DEPARTMENT OF INFORMATICS

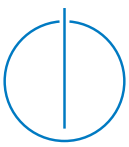
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Neural Network Hyperparameter Optimization with Sparse Grids

Parameteroptimierung von neuronalen Netzen mit dünnen Gittern

Author:	Maximilian Michallik
Supervisor:	Dr. Felix Dietrich
Advisor:	Dr. Michael Obersteiner, Dr. Felix Dietrich
Submission Date:	



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, Maximilian Michallik

Acknowledgments

Abstract

In recent years, machine learning has gained much importance due to the increasing amount of available data. The models that are performing very different tasks have a thing in common. They have parameters that are fixed before being trained on the data. The right choice of those hyperparameters can have a huge impact on the performance which is why they have to be optimized. Different techniques like grid search, random search, and bayesian optimization tackle this problem.
In this thesis, a new approach called adaptive sparse grid search for hyperparameter optimization is introduced. This new technique allows to adapt to the hyperparameter space and the model which leads to less training and evaluation runs compared to normal grid search while still finding the optimal model configuration for the best model results.
We compare the new approach to the other three techniques mentioned regarding execution time and resulting model performance using different machine learning tasks. The results show that adaptive sparse grid search is very efficient with a model performance similar to bayesian optimization and grid search.

Zusammenfassung

Contents

1 Introduction

2 Related Work

2.1 Hyperparameter Optimization

Most machine learning models have parameters that have to be defined before the learning phase. They are called hyperparameters and strongly influence the behavior of the model. One example is the number of epochs of the learning phase of a neural network. There are different techniques for the optimization of hyperparameters and they all define the machine learning model as a black box function f with the hyperparameters as input and the resulting performance as output. The overall goal is to find a configuration λ_{min} from $\Lambda = \Lambda_1 \times \Lambda_2 \times ... \times \Lambda_N$ that minimizes the function f with N hyperparameters with

$$\lambda_{min} = \arg\min_{\lambda \in \Lambda} f(\lambda). \tag{2.1}$$

In our case, the function f is a machine learning algorithm that is trained on a training set and evaluated on a testing set. With this, the minimization of e.g. the loss of the model optimizes the decisions it is making which leads to better prediction results. Note that one function evaluation of f is usually very expensive as the training of a machine learning model with many parameters and weights takes much time. The data set consists of $\{(x_i, y_i) | x_i \in X, y_i \in Y, 0 \leq i \leq m\}$ with m being the number of data samples. The x_i is the input data to the model and the goal is that

$$\forall i : M(x_i) = y_i. \tag{2.2}$$

where M is the model. In the context of supervised learning, the whole data set is split into a training set which is used to optimize the model and a testing set to evaluate the performance on new, unseen data [1].

All in all, the goal is get evaluation scores on the testing data set which can be achieved with Equation 2.1. [2], [3]

In the following, different techniques for the optimization are presented and discussed with their advantages and disadvantages.

2.1.1 Grid Search

The idea of the first approach for the optimization is to discretize the domains of each hyperparameter and evaluate each combination. This suffers from the curse of the dimensionality as it scales exponentially with the number of hyperparameters. For d parameters and n values per hyperparameter, n^d different configurations are possible which all have to be evaluated.

One advantage of this method is that it is easy to implement and very simple. Also, the whole search space is explored evenly.

On the other hand, the curse of the dimensionality makes it very slow if the function evaluations are very expensive. This is the case for most machine learning algorithms because the training phase of the model takes much time. Another drawback is that each hyperparameter only takes n different values.

2.1.2 Random Search

The next technique [4] is similar to the grid search because the idea is also to evaluate different hyperparameter configurations. In contrast to the previous one, random search generates for each run and for each parameter exactly one random value from an interval which has to be specified. For this approach, a budget b has to be given. This parameter determines the number of different combinations that are evaluated. A direct comparison of grid search and random search can be seen in figure 2.1.

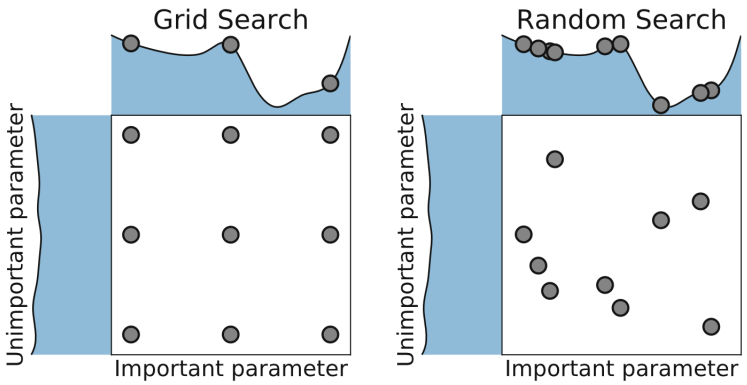


Figure 2.1: Comparison of grid search (left) and random search (right) in the two dimensional case. For both techniques, 9 different combinations are evaluated. In the left case, only 3 distinct values for each hyperparameter are set whereas there are 9 different values for each parameter in the random search. Taken from [2].

In this figure, a two dimensional setting is depicted. For both techniques, 9 different combinations are evaluated. In the case of grid search, only 3 distinct values are taken for each hyperparameter while there are 9 different ones in the random search. In this example, the better result is found in random search as more distinct values are taken for the important parameter. Note that this is not always the case.

Compared to the normal grid search, this is one advantage. For each hyperparameter, b (budget) different values are taken into consideration which is much more compared to the grid search with the same overall number of combinations. Additionally, this technique is also easy to implement and relatively simple.

One disadvantage is that it is also very expensive if the budget is high because of the long training times of machine learning models.

2.1.3 Bayesian Optimization

[3]

Algorithm 1 Bayesian Optimization

Generate initial $\lambda^{(1)}, ..., \lambda^{(k)}$
Initialize archive $A^{[0]} \leftarrow ((\lambda^{(1)}, f(\lambda^{(1)})), ..., (\lambda^{(k)}, f(\lambda^{(k)})))$
$t \leftarrow 1$
while Stopping criterion not met **do**
    Fit surrogate model $(f(\lambda), \sigma(\lambda))$ on $A^{[t-1]}$
    Build acquisition function $u(\lambda)$ from $(f(\lambda), \sigma(\lambda))$
    Obtain proposal λ^+ by optimizing $u : \lambda^+ \in arg\max \lambda \in \Lambda u(\lambda)$
    Evaluate $f(\lambda^+)$
    Obtain $A[t]$ by augmenting $A^{[t-1]}$ with $(\lambda^{(+)}, f(\lambda^{(+)}))$
    $t \leftarrow t + 1$
end while
return λ_{best}: Best-performing λ from archive or according to surrogates prediction

2.1.4 Other Techniques

2.2 Sparse Grids

2.2.1 Numerical Approximation of Functions

2.2.2 Adaptive Sparse Grids

3 Hyperparameter optimization with sparse grids

3.1 Methodology

3.1.1 Adaptive Grid Search with Sparse Grids

3.1.2 Implementation

3.2 Results

4 Conclusion and Outlook

List of Figures

List of Tables

Bibliography

[1] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," *Machine learning techniques for multimedia: case studies on organization and retrieval*, pp. 21–49, 2008.

[2] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated machine learning: Methods, systems, challenges*, pp. 3–33, 2019.

[3] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, e1484, 2021.

[4] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.