AutoML - Predicting Optimal Hyperparameter

Shriya jayant Salian
Pallavi Ramsingh Aike
ASM Institute of Management & Computer Studies

Email:
Divyasalian0123@gmail.com/Pallaviaike61@gmail.com

Abstract

Algorithm selection and hyperparameter tuning remain two of the most challenging tasks in machine learning. The number of machine learning applications is growing much faster than thenumber of machine learning experts, hence we see an increasing demand for efficient automation of learning processes.

Keywords: Machine Learning, Model Selection, Hyperparameter Optimization, AutomatedMachine Learning.

1 Introduction

Machine learning and data science experts find it difficult to select algorithms and hyperparameter settings suitable for a given dataset; for novices, the challenge is even greater. The large number of algorithms, and the sensitivity of these methods to hyperparameter values, makes it practically infeasible to enumerate all possible configurations. To surmount these challenges, the field of *Automated Machine Learning* (AutoML) seeks to efficiently automate the selection of model configurations, and has attracted increasing attention in recent years. Especially recent deep neural networks crucially depend on a wide range of hyperparameter choices about the neural network's architecture,regularization, and optimization. Automated hyperparameter optimization (HPO) has several important use cases; it can

- Reduce the human effort necessary for applying machine learning. This is particularly important in the context of AutoML.
- improve the performance of machine learning algorithms (by tailoring them to the problem at hand); this has led tonew state-of-the-art performances forimportant machine learningbenchmarks in several studies.
- improve the reproducibility and fairness of scientific studies. Automated HPO is clearly more reproducible than manual search. It facilitates fair comparisons since different methods can only be compared fairly if they all receive the same level of tuning for the problem at hand

The problem of HPO has a long history, dating back to the 1990s and it was also established early that different hyperparameter configurations tend to work best for different dataset. In contrast, it is a rather new insight that HPO can be used to adapt general-purpose pipelines to specific application domains. Nowadays, it is also widely acknowledged that tuned hyperparameters improve over the default setting provided by common machine learning libraries.

Because of the increased usage of machine learning in companies, HPO is also of substantial commercial interest and plays an ever larger role there, be it in company-internal tools, as part of machine learning cloud service, or as a service by itself.

This paper is organized as follows. Section 2

2 Understanding Hyperparameters

2.1 What is a parameter in a ML learningmodel?

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from the given data.

- They are required by the model when making predictions.
- Their values define the skill of the model on your problem.
- They are estimated or learned from data.
- They are often not set manually by the practitioner.
- They are often saved as part of the learned model.

So, parameters are crucial to machine learning algorithms. Also, they are the part of the model that is learned from historical training data.

Some examples of model parameters include:

- The weights in an artificial neural network.
- The support vectors in a support vector machine.
- The coefficients in a linear regression or logistic regression.

2.2 What is a hyperparameter in a ML learning model?

A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.

- They are often used in processes to help estimate model parameters.
- They are often specified by the practitioner.
- They can often be set using heuristics.
- They are often tuned for a given predictive modeling problem.

You cannot know the best value for a model hyperparameter on a given problem. You may use rules of thumb, copy values used on other issues, or search for the best value by trial and error. When a machine learning algorithm is tuned for a specific problem then essentially you are tuning the hyperparameters of the model to discover the parameters of the model that result in the most skillful predictions.

Some examples of model hyperparameters include:

- The learning rate for training a neural network.
- The C and sigma hyperparameters for support vector machines.
- The k in k-nearest neighbors.

2.3 Importance of the right set of hyperparameter values in a machine learning model

The best way to think about hyperparameters is like the settings of an algorithm that can be adjusted to optimize performance, just as you might turn the knobs of an AM radio to get a clear signal. When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often, you don't immediately know what the optimal model architecture should be for a given model, and thus you'd like to be able to explore a range of possibilities. In a true machine learning fashion, you'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically.

2.4 Two simple strategies to optimize/tune the hyperparameters

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem.

Although there are many hyperparameter optimization/tuning algorithms now, this post discusses two simple strategies: 1. grid search and 2. Random Search.

2.4.1 Grid searching of hyperparameters

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

Let’s consider the following example:

Suppose, a machine learning model X takes hyperparameters a1, a2 and a3. In grid searching, you first define the range of values for each of the hyperparameters a1, a2 and a3. You can think of this as an array of values for each of the hyperparameters. Now the grid search technique will construct many versions of X with all the possible combinations of hyperparameter (a1, a2 and a3) values that you defined in the first place. This range of hyperparameter values is referred to as the grid.

Suppose, you defined the grid as:

a1 = [0,1,2,3,4,5]

a2 = [10,20,30,40,5,60]

a3 = [105,105,110,115,120,125]

Note that, the array of values of that you are defining for the hyperparameters has to be legitimate in a sense that you cannot supply Floating type values to the array if the hyperparameter only takes Integer values.

Now, grid search will begin its process of constructing several versions of X with the grid that you just defined.

It will start with the combination of [0,10,105], and it will end with [5,60,125]. It will go through all the intermediate combinations between these two which makes grid search computationally very expensive.

2.4.2 Random searching ofhyperparameters

The idea of random searching of hyperparameters was proposed by James Bergstra & Yoshua Bengio.

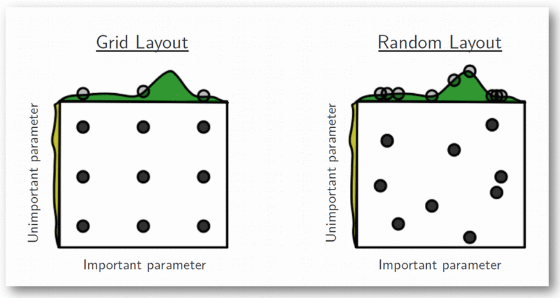
Random search differs from a grid search. In that you longer provide a discrete set of values to explore for each hyperparameter; rather, you provide a statistical distribution for each hyperparameter from which values may be randomly sampled.

You’ll define a sampling distribution for each hyperparameter. You can also define how many iterations you’d like to build when searching for the optimal model. For each iteration, the hyperparameter values of the model will be set by sampling the defined distributions. One of the primary theoretical backings to motivate the use of a random search in place of grid search is the fact that for most cases, hyperparameters are not equally important. According to the original paper:

“….for most datasets only a few of the hyper-parameters really matter, but that different hyper-parameters are important on different datasets. This phenomenon makes grid search a poor choice for configuring algorithms for new datasets.”

In the following figure, we're searching over a hyperparameter space where the one hyperparameter has significantly more influence on optimizing the model score - the distributions shown on each axis represent the model's score. In each case, we're evaluating nine different models. The grid search strategy blatantly misses the optimal model and spends redundant time exploring the unimportant parameter.

During this grid search, we isolated each hyperparameter and searched for the best possible value while holding all other hyperparameters constant. For cases where the hyperparameter being studied has little effect on the resulting model score, this results in wasted effort. Conversely, the random search has much improved exploratory power and can focus on finding the optimal value for the critical hyperparameter.



3 Related Work

The above two strategies described are Model-Free Blackbox Optimization Methods. Another approach is using Bayesian Optimization. Bayesian optimization is a state-of-the-art optimization framework for the global optimization of expensive Blackbox functions, which recently gained traction in HPO by obtaining new state-of-the-art results in tuning deep neural networks for image classification, speech recognition and neural language modeling, and by demonstrating wide applicability to different problem settings.

4 How Hyperparameter OptimizationHPO Works?

Bayesian optimization is an iterative algorithm with two key ingredients: a probabilistic surrogate model and an acquisition function to decide which point to evaluate next.

Bayesian optimization finds the value that minimizes an objective function by building a surrogate function (probability model) based on past evaluation results of the objective. The surrogate is cheaper to optimize than the objective, so the next input values to evaluate are selected by applying a criterion to the surrogate (often Expected Improvement). Bayesian methods differ from random or grid search in that they use past evaluation results to choose the next values to evaluate. The concept is: limit expensive evaluations of the objective function by choosing the next input values based on those that have done well in the past.

In the case of hyperparameter optimization, the objective function is the validation error of a machine learning model using a set of hyperparameters. The aim is to find the hyperparameters that yield the lowest error on the validation set in the hope that these results generalize to the testing set. Evaluating the objective function is expensive because it requires training the machine learning model with a specific set of hyperparameters. Ideally, we want a method that can explore the search space while also limiting evaluations of poor hyperparameterchoices.

Bayesianhyperparameter tuning uses a continually updated probability model to “concentrate” on promising hyperparameters by reasoning from past results.

4.1 Four Parts of OptimizationProblem

There are four parts to a Bayesian Optimization problem:

- **Objective Function:** what we want to minimize, in this case the validation error of a machine learning model with respect to the hyperparameters

- **Domain Space:** hyperparameter values to search over
- **Optimization algorithm:** method for constructing the surrogate model and choosing the next hyperparameter values to evaluate
- **Result history:** stored outcomes from evaluations of the objective function consisting of the hyperparameters and validation loss

With those four pieces, we can optimize (find the minimum) of any function that returns areal value. This is a powerful abstraction that lets us solve many problems in addition to tuning machine learning hyperparameters.

5 Applications of AutoML

We provide a historical overview of the most important hyperparameter optimization systems and applications to automated machinelearning.

Grid search has been used for hyperparameter optimization since the 1990s and was already supported by early machine learning tools in 2002. The first adaptive optimization methods applied to HPO were greedy depth-first search and pattern search, both improving over default hyperparameter configurations, and pattern search improving over grid search, too.

Genetic algorithms were first applied to tuning the two hyperparameters C and γ of an RBF-SVM in 2004 [03] and resulted in improved classification performance in less time than grid search. In the same year, an evolutionary algorithm was used to learn a composition of three different kernels for an SVM, the kernel hyperparameters and to jointly select a feature subset; the learned combination of kernels was able to outperform every single optimized kernel.

6 Conclusion

Based on the research done for this paper it concludes that machine learning engineers and data scientist need HPOs and automated model selection tools. Since 1990s we are using some sort of optimization techniques to optimize models hyperparameters. Future work - Given some input about what the problem is, it would be good if machine learning model itself give predictions (suggestions) about which modelsor models might be best suited for the given problem statement.

References

www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

https://www.cs.ubc.ca/~hoos/Publ/EggEtAl13.pdf

https://link.springer.com/chapter/10.1007/978-3-030-05318-5_1#CR105

https://www.topbots.com/automl-model-selection-optiml/

https://www.arxiv-vanity.com/papers/1808.03233/