CS 769 - Optimization in Machine Learning
Course Proect
Hyperparameter Optimization of Machine Learning Algorithms

A Preprint

Manas Jain (170040068)
Department of Civil Engineering
Indian Institute of Technology, Bombay
manasjain@iitb.ac.in

May 10, 2021

Abstract

To fit a machine learning model into different problems, its hyper-parameters must be tuned. Selecting the best hyper-parameter configuration for machine learning models has a direct impact on the model's performance. In this project, optimizing the hyper-parameters of common machine learning models is explored. We present several state-of-the-art optimization techniques and discuss how to apply them to machine learning algorithms. Many available libraries and frameworks developed for hyper-parameter optimization problems are provided, and some open challenges of hyper-parameter optimization research are also studied in this project. Moreover, experiments are conducted on benchmark datasets to compare the performance of different optimization methods and provide practical examples of hyper-parameter optimization. This can be very helpful for industrial users, data analysts, and researchers to get insight of various hyperparameter optimization techniques and better develop machine learning models by identifying the proper hyper-parameter configurations effectively.

***K*eywords** Hyperparameter Tuning · Optimization · Machine Learning

1 Introduction

Different ML algorithms are suitable for different types of problems or datasets. In general, building an effective machine learning model is a complex and time-consuming process that involves determining the appropriate algorithm and obtaining an optimal model architecture by tuning its hyper-parameters. To build an optimal ML model, a range of possibilities must be explored. The process of designing the ideal model architecture with an optimal hyper-parameter configuration is named hyper-parameter tuning. Tuning hyper-parameters is considered a key component of building an effective ML model, especially for tree-based ML models and deep neural networks, which have many hyper-parameters. Apart from performance improvement of ML models, one important reason for applying HPO techniques to ML models is It makes the models and research more reproducible. Only when the same level of hyper-parameter tuning process is implemented can different ML algorithms be compared fairly; hence, using a same HPO method on different ML algorithms also helps to determine the most suitable ML model for a specific problem.
Manual testing is a traditional way to tune hyper-parameters and is still prevalent in graduate student research, although it requires a deep understanding of the used ML algorithms and their hyper-parameter value settings. However, manual tuning is ineffective for many problems due to certain factors, including a large number of hyper-parameters, complex models, time consuming model evaluations, and non-linear hyper-parameter interactions. These factors have inspired increased research in techniques for automatic optimization of hyper-parameters; so-called hyper-parameter optimization (HPO). The main aim of HPO is to automate hyper-parameter tuning process and make it possible for users to apply machine learning models to practical problems effectively. The optimal model architecture of a ML

model is expected to be obtained after a HPO process. Some important reasons for applying HPO techniques to ML models are as follows:

1. It reduces the human effort required, since many ML developers spend considerable time tuning the hyper-parameters, especially for large datasets or complex ML algorithms with a large number of hyperparameters.
2. It improves the performance of ML models. Many ML hyper-parameters have different optimums to achieve best performance in different datasets or problems.
3. It makes the models and research more reproducible. Only when the same level of hyper-parameter tuning process is implemented can different ML algorithms be compared fairly; hence, using a same HPO method on different ML algorithms also helps to determine the most suitable ML model for a specific problem.

2 Hyper-parameter Optimization Problem Statement

The hyper-parameter optimization process consists of four main components: an estimator (a regressor or a classifier) with its objective function, a search space (configuration space), a search or optimization method used to find hyper-parameter combinations, and an evaluation function to compare the performance of different hyper-parameter configurations. The domain of a hyper-parameter can be continuous (e.g., learning rate), discrete (e.g., number of clusters), binary (e.g., whether to use early stopping or not), or categorical (e.g., type of optimizer) In general, for a hyper-parameter optimization problem, the aim is to obtain :

$$x^* = \underset{x \epsilon X}{\operatorname{argmin}} f(x) \tag{1}$$

where $f(x)$ is the objective function to be minimized, such as the error rate or the root mean squared error (RMSE); x_* is the hyper-parameter configuration that produces the optimum value of $f(x)$; and a hyper-parameter x can take any value in the search space X.

3 HPO Techniques

Compared with traditional optimization methods like gradient descent, many other optimization techniques are more suitable for HPO problems, including decision-theoretic approaches, Bayesian optimization models, multifidelity optimization techniques, and metaheuristics algorithmsYang and Shami (2020).

1. Grid search (GS) is a decision-theoretic approach that involves exhaustively searching for a fixed domain of hyperparameter values
2. Random search (RS) is another decision-theoretic method that randomly selects hyper-parameter combinations in the search space, given limited execution time and resources
3. Bayesian optimization (BO) models determine the next hyper-parameter value based on the previous results of tested hyperparameter values, which avoids many unnecessary evaluations. BO can model the distribution of the objective function using different models as the surrogate function, including Gaussian process (GP), random forest (RF), and tree-structured Parzen estimators (TPE) models
4. Multifidelity optimization algorithms are developed to tackle problems with limited resources, and the most common ones being bandit-based algorithms. Hyperband is a popular bandit-based optimization technique that can be considered an improved version of RS
5. Metaheuristic algorithms are a set of techniques used to solve complex, large search space and non-convex optimization problems to which HPO problems belong. Among all metaheuristic methods, genetic algorithm (GA) and particle swarm optimization (PSO) are the two most prevalent metaheuristic algorithms used for HPO problems. Genetic algorithms detect well-performing hyper-parameter combinations in each generation, and pass them to the next generation until the best-performing combination is identified. In PSO algorithms, each particle communicates with other particles to detect and update the current global optimum in each iteration until the final optimum is detected

3.1 Bayesian Optimization : HPO technique

To determine the next hyper-parameter configuration, BO uses two key components: a surrogate model and an acquisition function . The surrogate model aims to fit all the currently-observed points into the objective function.

After obtaining the predictive distribution of the probabilistic surrogate model, the acquisition function determines the usage of different points by balancing the trade-off between exploration and exploitation. Exploration is to sample the instances in the areas that have not been sampled, while exploitation is to sample in the currently promising regions where the global optimum is most likely to occur, based on the posterior distribution. The basic procedures of BO are as follows :

1. Build a probabilistic surrogate model of the objective function.
2. Detect the optimal hyper-parameter values on the surrogate model.
3. Apply these hyper-parameter values to the real objective function to evaluate them.
4. Update the surrogate model with new results.
5. Repeat steps 2 - 4 until the maximum number of iterations is reached.

Common surrogate models for BO include Gaussian process (GP), random forest (RF), and the tree Parzen estimator (TPE). Therefore, there are three main types of BO algorithms based on their surrogate models: BO-GP, BO-RF, BO-TPE. An alternative name for BO-RF is sequential model-based algorithm configuration (SMAC).

3.1.1 BO-GP

Gaussian process (GP) is a standard surrogate model for objective function modeling in BO. Assuming that the function f with a mean μ and a covariance σ is a realization of a GP, the predictions follow a normal distribution :

$$P(y|x, D) = \mathcal{N}(y|\mu, \sigma^2) \tag{2}$$

where D is the configuration space of hyper-parameters, and $y = f(x)$ is the evaluation result of each hyper-parameter value x. After obtaining a set of predictions, the points to be evaluated next are then selected from the confidence intervals generated by the BO-GP model. Each newly-tested data point is added to the sample records, and the BO-GP model is re-built with the new information. This procedure is repeated until termination.
Applying a BO-GP to a size n dataset has a time complexity of $O(n^3)$ and space complexity of $O(n^2)$. One main limitation of BO-GP is that the cubic complexity to the number of instances limits the capacity for parallelization. Additionally, it is mainly used to optimize continuous variables

3.1.2 BO-TPE

Tree-structured Parzen estimator (TPE) is another common surrogate model for BO. To apply TPE, the observation results are divided into good results and poor results by a pre-defined percentile y_*, and the two sets of results are modeled by simple Parzen windows.
After that, the expected improvement in the acquisition function is reflected by the ratio between the two density functions, which is used to determine the new configurations for evaluation. The Parzen estimators are organized in a tree structure, so the specified conditional dependencies are retained. Therefore, TPE naturally supports specified conditional hyperparameters. The time complexity of BO-TPE is $O(nlogn)$, which is lower than the complexity of BO-GP.

3.2 Multi-fidelity Optimization Algorithms

Multifidelity optimization techniques are common approaches to solve the constraint of limited time and resources. To save time, people can use a subset of the original dataset or a subset of the features. Multi-fidelity involves low-fidelity and high-fidelity evaluations and combines them for practical applications. In low-fidelity evaluations, a relatively small subset is evaluated at a low cost but with poor generalization performance. In high-fidelity evaluations, a relatively large subset is evaluated with better generalization performance but at a higher cost than low-fidelity evaluations. In multifidelity optimization algorithms, poorly-performing configurations are discarded after each round of hyper-parameter evaluation on generated subsets, and only well-performing hyper-parameter configurations will be evaluated on the entire training set.
Bandit-based algorithms categorized to multi-fidelity optimization algorithms have shown success dealing with deep learning optimization problems.

3.2.1 Hyperband

Hyperband Li et al. (2018) is proposed to solve the dilemma of successive halving algorithms by dynamically choosing a reasonable number of configurations. It aims to achieve a trade-off between the number of hyper-parameter configurations (n) and their allocated budgets by dividing the total budgets (B) into n pieces and allocating these pieces to each

configuration ($b = B/n$). Successive halving serves as a subroutine on each set of random configurations to eliminate the poorly-performing hyper-parameter configurations and improve efficiency.
The successive halving algorithm discards the identified poorly-performing configurations and passes the well-performing configurations on to the next iteration. This process is repeated until the final optimal hyper-parameter configuration is identified. By involving the successive halving searching method, Hyperband has a computational complexity of $O(nlogn)$

3.2.2 Bayesian Optimization HyperBand

BOHB Wang et al. (2018) is a state-of-the-art HPO technique that combines Bayesian optimization and Hyperband to incorporate the advantages of both while avoiding their drawbacks. The original Hyperband uses a random search to search the hyper-parameter configuration space, which has a low efficiency. BOHB replaces the RS method by BO to achieve both high performance as well as low execution time by effectively using parallel resources to optimize all types of hyper-parameters. In BOHB, TPE is the standard surrogate model for BO, but it uses multidimensional kernel density estimators. Therefore, the complexity of BOHB is also $O(nlogn)$.
It has been shown that BOHB outperforms many other optimization techniques when tuning SVM and DL models. The only limitation of BOHB is that it requires the evaluations on subsets with small budgets to be representative of evaluations on the entire training set; otherwise, BOHB may have a slower convergence speed than standard BO models. Falkner et al. (2018)

3.3 Metaheuristic Algorithms

Metaheuristic algorithms are a set of algorithms mainly inspired by biological theories and widely used for optimization problems. Unlike many traditional optimization methods, metaheuristics have the capacity to solve non-convex, non-continuous, and non-smooth optimization problems.
Population-based optimization algorithms (POAs) are a major type of metaheuristic algorithm, including genetic algorithms (GAs), evolutionary algorithms, evolutionary strategies, and particle swarm optimization (PSO). POAs start by creating and updating a population as each generation; each individual in every generation is then evaluated until the global optimum is identified. The main differences between different POAs are the methods used to generate and select populations. POAs can be easily parallelized since a population of N individuals can be evaluated on at most N threads or machines in parallel. Genetic algorithms and particle swarm optimization are the two main POAs that are popularly-used for HPO problems.

3.3.1 Genetic Algorithm

Genetic algorithm (GA) is one of the common metaheuristic algorithms based on the evolutionary theory that individuals with the best survival capability and adaptability to the environment are more likely to survive and pass on their capabilities to future generations. The next generation will also inherit their parents' characteristics and may involve better and worse individuals. Better individuals will be more likely to survive and have more capable offspring, while the worse individuals will gradually disappear. After several generations, the individual with the best adaptability will be identified as the global optimum
To apply GA to HPO problems, each chromosome or individual represents a hyper-parameter, and its decimal value is the actual input value of the hyper-parameter in each evaluation. Every chromosome has several genes, which are binary digits; and then crossover and mutation operations are performed on the genes of this chromosome. The population involves all possible values within the initialized chromosome/parameter ranges, while the fitness function characterizes the evaluation metrics of the parameters.

3.3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is another set of evolutionary algorithms that are commonly used for optimization problems. PSO algorithms are inspired by biological populations that exhibit both individual and social behaviors. PSO works by enabling a group of particles (swarm) to traverse the search space in a semi-random manner. PSO algorithms identify the optimal solution through cooperation and information sharing among individual particles in a group.
The main limitation of PSO is that it requires proper population initialization; otherwise, it might only reach a local instead of a global optimum, especially for discrete hyper-parameters. Proper population initialization requires developers prior experience or using population initialization techniques. Many population initialization techniques have been proposed to improve the performance of evolutionary algorithms, like the opposition-based optimization algorithm and the space transformation search method. Involving additional population initialization techniques will require more execution time and resources.

Table 3: Configuration space for the hyper-parameters of tested ML models

ML Model	Hyper-parameter	Type	Search Space
RF Classifier	n_estimators	Discrete	[10,100]
	max_depth	Discrete	[5,50]
	min_samples_split	Discrete	[2,11]
	min_samples_leaf	Discrete	[1,11]
	criterion	Categorical	['gini', 'entropy']
	max_features	Discrete	[1,64]
SVM Classifier	C	Continuous	[0.1,50]
	kernel	Categorical	['linear', 'poly', 'rbf', 'sigmoid']
KNN Classifier	n_neighbors	Discrete	[1,20]
RF Regressor	n_estimators	Discrete	[10,100]
	max_depth	Discrete	[5,50]
	min_samples_split	Discrete	[2,11]
	min_samples_leaf	Discrete	[1,11]
	criterion	Categorical	['mse', 'mae']
	max_features	Discrete	[1,13]
SVM Regressor	C	Continuous	[0.1,50]
	kernel	Categorical	['linear', 'poly', 'rbf', 'sigmoid']
	epsilon	Continuous	[0.001,1]
KNN Regressor	n_neighbors	Discrete	[1,20]

Figure 1: Experimental Setup HP search space

4 Experiments

Firstly, two standard benchmarking datasets provided by the sklearn library, namely, the Modified National Institute of Standards and Technology dataset (MNIST) and the Boston housing dataset, are selected as the benchmark datasets for HPO method evaluation on data analytics problems. MNIST is a hand-written digit recognition dataset used as a multiclassification problem, while the Boston housing dataset contains information about the price of houses in various places in the city of Boston and can be used as a regression dataset to predict the housing prices.
At the next stage, the ML models with their objective function need to be configured. Thus, three ML algorithms, KNN, SVM, and RF, are selected as the target models to be optimized, since their hyper-parameter types represent the three most common HPO cases: KNN has one important hyper-parameter, the number of considered nearest neighbors for each sample; SVM has a few conditional hyper-parameters, like the kernel type and the penalty parameter C; RF has multiple hyperparameters of different types. Moreover, KNN, SVM, and RF can all be applied to solve both classification and regression problems.
In the next step, the performance metrics and evaluation methods are configured. For each experiment on the selected two datasets, 3-fold cross validation is implemented to evaluate the involved HPO methods. The two most commonly-used performance metrics are used in our experiments. For classification models, accuracy is used as the classifier performance metric, which is the proportion of correctly classified data; while for regression models, the mean squared error (MSE) is used as the regressor performance metric, which measures the average squared difference between the predicted values and the actual values. Additionally, the computational time (CT) , the total time needed to complete a HPO process with 3-fold cross-validation, is also used as the model efficiency metric.
Grid search, random search, hyperband, Bayesian Optimization with Gaussian Processes (BO-GP), Bayesian Optimization with Tree-structured Parzen Estimator (BO-TPE), particle swarm optimization (PSO), genetic algorithm (GA) HPO tecniques results were compared.
The involved ML and HPO algorithms are evaluated using multiple open-source Python libraries and frameworks including sklearn, Skopt, Hyperopt, Optunity, Hyperband, BOHB, and TPOT

5 Results

The experiments of applying 7 different HPO methods to ML models are summarized in below figures. Figure 2 provide the performance of each optimization algorithm when applied to RF, SVM, and KNN classifiers evaluated on the MNIST dataset after a complete optimization process; while Figure 3 demonstrate the performance of each HPO

RESULTS (classification)

	RF	SVM	KNN	ANN
Default HP	89.65	96.99	96.27	98.99
GS	93.6	97.38	**96.83**	99.61
RS	93.09	97.44	96.44	**100**
Hyperband	93.21	97.44	96.21	99.94
BO-GP	**93.99**	97.44	96.82	**100**
BO-TPE	93.48	**97.49**	**96.83**	**100**
PSO	92.56	96.05	**96.83**	99.05
GA	92.04	97.44	**96.83**	99.94

Figure 2: Classification on MNIST data

RESULTS (Regression)

	RF	SVM	KNN	ANN
Default HP	32.43	77.42	81.49	43.29
GS	29.03	67.07	81.53	52.18
RS	26.38	60.03	80.77	53.52
Hyperband	26.44	70.78	80.87	56.59
BO-GP	26.14	59.52	80.77	63.93
BO-TPE	26.87	63.07	81.26	58.39
PSO	26.07	60.26	**80.74**	***39.02***
GA	**26.05**	**59.41**	80.77	50.73

Figure 3: Regression on Boston data

method when applied to RF, SVM, and KNN regressors evaluated on the Boston-housing dataset. In the first step, each ML model with its default hyper-parameter configuration is trained and evaluated as the baseline model. After that, each HPO algorithm is implemented on the ML models to evaluate and compare their accuracies for classification problems, or MSEs for regression problems.

The performance of BO and multi-fidelity models is much better than GS and RS. The computation time of BO-GP is often higher than other HPO methods due to its cubic time complexity, but it can obtain better performance metrics for ML models with small-size continuous hyper-parameter space, like KNN. Conversely, hyperband is often not able to obtain the highest accuracy or the lowest MSE among the optimization methods, but their computational time is low because it works on the small-sized subsets. The performance of BO-TPE is often better than others, since they can detect the optimal or near-optimal hyper-parameter configurations within a short computational time.

For metaheuristics methods, GA and PSO, their accuracies are often higher than other HPO methods for classification problems, and their MSEs are often lower than other optimization techniques. However, their computational time is often higher than BO-TPE and multi-fidelity models, especially for GA, which does not support parallel executions.

6 Conclusion

It is simple to implement GS and RS, but they often cannot detect the optimal hyper-parameter configurations or cost much computational time. BO-GP and GA also cost more computational time than many other HPO methods, but BO-GP works well on small configuration space, while GA is effective for large configuration space. Hyperband's computational time is low, but it cannot guarantee to detect the global optimums. For ML models with large configuration space, BO-TPE, BOHB, and PSO often work well.
To apply optimization methods to ML models, the hyper-parameter types in a ML model is the main concern for HPO method selection. To summarize, BOHB is the recommended choice for optimizing a ML model, if randomly selected subsets are highly-representative of the given dataset, since it can efficiently optimize all types of hyper-parameters; otherwise, BO models are recommended for small hyper-parameter configuration space, while PSO is usually the best choice for large configuration space.

References

Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295 – 316, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.07.061. URL `http://www.sciencedirect.com/science/article/pii/S0925231220311693`.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

Jiazhuo Wang, Jason Xu, and Xuejun Wang. Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning, 2018.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale, 2018.