CS 769 Course Project
Hyperparameter Optimization of Machine Learning Algorithms

Manas Jain - 170040068

10th May 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.

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.

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 :

$$x^{\cdot} = \underset{x \in X}{argmin}\, f(x)$$

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.

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 algorithms.

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
4. 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

HPO Techniques (continue)

1. 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
2. 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

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.

HPO Algorithm 1: Grid Search

Search all the given hyper-parameter configurations. For GS, assuming that there are k parameters, and each of them has n distinct values, its computational complexity increases exponentially at a rate of O(n^k). Thus, only when the hyper-parameter configuration space is small can GS be an effective HPO method.

Advantages:

- Simple implementation and parallelized.

Disadvantages:

- Time-consuming,
- Only efficient with categorical HPs.

HPO Algorithm 2: Random Search

Randomly search hyper-parameter combinations in the search space. Since the number of total evaluations in RS is set to a fixed value n before the optimization process starts, the computational complexity of RS is O(n). In addition, RS can detect the global optimum or the near-global optimum when given enough budgets. RS is more efficient than GS for large search spaces

Advantages:

- More efficient than GS.
- Enable parallelization.

Disadvantages:

- Not consider previous results.
- Not efficient with conditional HPs.

HPO Algorithm 3: Hyperband

Generate small-sized subsets and allocate budgets to each hyper-parameter combination based on its performance. 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. By involving the successive halving searching method, Hyperband has a computational complexity of O(nlogn)

Advantages:

- Enable parallelization.

Disadvantages:

- Not efficient with conditional HPs.
- Require subsets with small budgets to be representative.

HPO Algorithm 4: BO-GP

Bayesian Optimization with Gaussian Process (BO-GP). Applying a BO-GP to a size n dataset has a time complexity of O(n^3) and space complexity of O(n^2)

Advantages:

- Fast convergence speed for continuous HPs.

Disadvantages:

- Poor capacity for parallelization.
- Not efficient with conditional HPs.

HPO Algorithm 5: BO-TPE

Bayesian Optimization with Tree-structured Parzen Estimator (TPE)

Advantages:

- Efficient with all types of HPs.
- Keep conditional dependencies.

Disadvantages:

- Poor capacity for parallelization.

HPO Algorithm 6: PSO

Partical swarm optimization (PSO): Each particle in a swarm communicates with other particles to detect and update the current global optimum in each iteration until the final optimum is detected. The computational complexity of PSO algorithm is O(nlogn). In most cases, the convergence speed of PSO is faster than of GA.

Advantages:

- Efficient with all types of HPs.
- Enable parallelization.

Disadvantages:

- Require proper initialization.

HPO Algorithm 7: Genetic Algorithm

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.

Advantages:

- Efficient with all types of HPs.
- Not require good initialization.

Disadvantages:

- Poor capacity for parallelization.

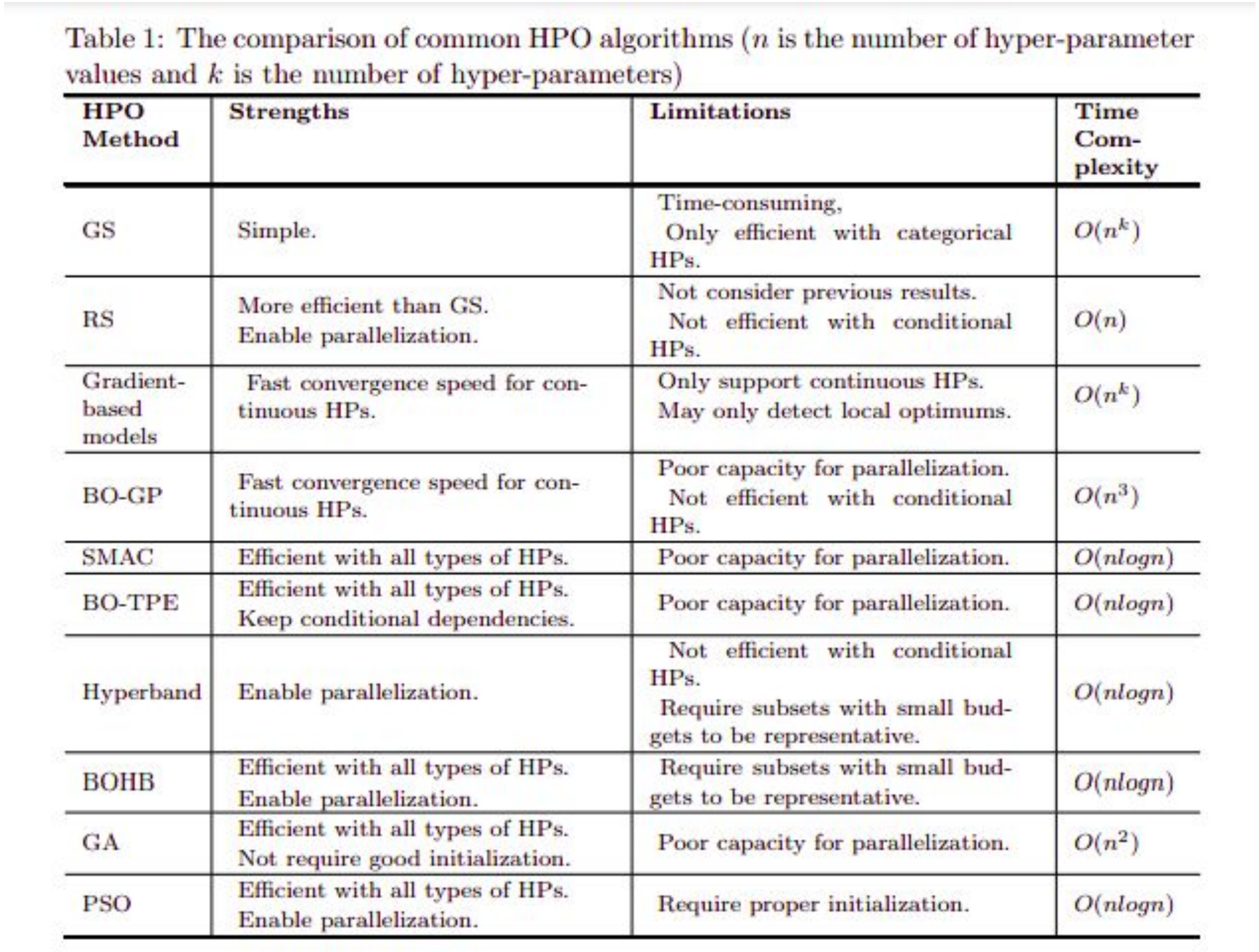

Table 1: The comparison of common HPO algorithms (n is the number of hyper-parameter values and k is the number of hyper-parameters)

HPO Method	**Strengths**	**Limitations**	**Time Complexity**
GS	Simple.	Time-consuming, Only efficient with categorical HPs.	$O(n^k)$
RS	More efficient than GS. Enable parallelization.	Not consider previous results. Not efficient with conditional HPs.	$O(n)$
Gradient-based models	Fast convergence speed for continuous HPs.	Only support continuous HPs. May only detect local optimums.	$O(n^k)$
BO-GP	Fast convergence speed for continuous HPs.	Poor capacity for parallelization. Not efficient with conditional HPs.	$O(n^3)$
SMAC	Efficient with all types of HPs.	Poor capacity for parallelization.	$O(nlogn)$
BO-TPE	Efficient with all types of HPs. Keep conditional dependencies.	Poor capacity for parallelization.	$O(nlogn)$
Hyperband	Enable parallelization.	Not efficient with conditional HPs. Require subsets with small budgets to be representative.	$O(nlogn)$
BOHB	Efficient with all types of HPs. Enable parallelization.	Require subsets with small budgets to be representative.	$O(nlogn)$
GA	Efficient with all types of HPs. Not require good initialization.	Poor capacity for parallelization.	$O(n^2)$
PSO	Efficient with all types of HPs. Enable parallelization.	Require proper initialization.	$O(nlogn)$

Experiments (1) - Regression problem Analysis

Dataset used:

Boson Housing dataset from sklearn

Machine learning algorithms used:

Random forest (RF), support vector machine (SVM), k-nearest neighbor (KNN), artificial neural network (ANN)

HPO algorithms used:

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).

Performance metric:

Mean square error (MSE)

Experiments (2) - Classification problem Analysis

Dataset used:

MNIST from sklearn

Machine learning algorithms used:

Random forest (RF), support vector machine (SVM), k-nearest neighbor (KNN), artificial neural network (ANN)

HPO algorithms used:

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).

Performance metric:

Classification accuracy

Experimental Setup

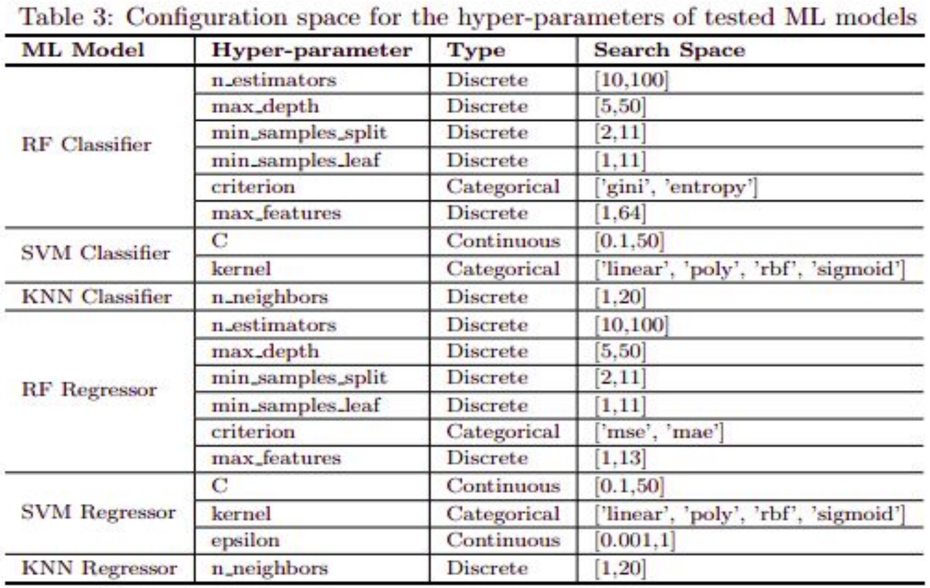

Table 3: Configuration space for the hyper-parameters of tested ML models

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]

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

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

Conclusion

BO models are recommended for small hyper-parameter configuration space, while PSO is usually the best choice for large configuration space

HPO frameworks used for implementation

- Keras
- scikit-learn
- hyperband
- scikit-optimize
- hyperopt
- optunity
- DEAP
- TPOT

References

On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice by Li Yang and Abdallah Shami

https://github.com/LiYangHart/Hyperparameter-Optimization-of-Machine-Learning-Algorithms

Numerous Wikipedia pages, github repos

THANK YOU