International Journal of Advanced Culture Technology Vol.7 No.4 56-62 (2019)
DOI 10.17703/IJACT.2019.7.4.56

IJACT 19-12-7

Analysis of Open-Source Hyperparameter Optimization Software Trends

Yo-Seob Lee[1], Phil-Joo Moon[2]*

[1]Professor, School of ICT Convergence, Pyeongtaek University
E-mail: yslee@ptu.ac.kr
[2]Professor, Dept. of Information & Communication, Pyeongtaek University
E-mail: pjmoon@ptu.ac.kr

Abstract

Recently, research using artificial neural networks has further expanded the field of neural network optimization and automatic structuring from improving inference accuracy. The performance of the machine learning algorithm depends on how the hyperparameters are configured. Open-source hyperparameter optimization software can be an important step forward in improving the performance of machine learning algorithms. In this paper, we review open-source hyperparameter optimization softwares.

Keywords: *Hyperparameter Optimization, Machine Learning, Deep Learning, AutoML*

1. INTRODUCTION

Recently, researches using artificial neural networks have further expanded the field of neural network optimization and automatic structuring from improving the inference accuracy of image, video and natural language based tasks. In addition, with recent interest in complex and computationally expensive machine learning models with many hyperparameters, such as the Automatic Machine Learning(AutoML) framework and deep neural networks, research on hyperparameter optimization(HPO) has begun again. The performance of machine learning algorithms significantly depends on how a configuration of hyperparameters is identified. Every machine learning system has hyperparameters, and the most basic task of AutoML is to set these hyperparameters automatically to optimize performance. In particular, modern deep neural networks critically depend on a broad selection of hyperparameters for the architecture, normalization and optimization of neural networks.[1][2]

HPO is faced with some issues that are actually a difficult problem:[2]

• Function evaluations can be extremely expensive for large models, complex machine learning pipelines, or large datesets.

• The configuration space is often and high-dimensional.

• We usually don't have access to a gradient of the loss function with respect to the hyperparameters.

• One cannot directly optimize for generalization performance as training datasets are of limited size.

In this paper, we review open-source hyperparameter optimization softwares, and compare various characteristics of hyperparameter optimization softwares.

Manuscript received: September 26, 2019 / revised: October 09, 2019 / Accepted: October 28, 2019
Corresponding Author: pjmoon@ptu.ac.kr
Tel:+82-31-659-8281, Fax: +82-31-659-8011
Professor, Dept. of Information and Communication, Pyeongtaek University, Korea

2. HYPERPARAMETER OPTIMIZATION

In machine learning, hyperparameter optimization is the problem of choosing a set of optimal hyperparameters for a learning algorithm.[3] It finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given test data.[4]

Hyperparameters are parameters whose values are set before the learning process begins. The values of other parameters are derived from training.[5]

Let $\mathcal{A}$ denote a machine learning algorithm with N hyperparameters. We denote $\mathcal{A}$ with its hyperparameters instantiated to λ is denoted by $\mathcal{A}_\lambda$. The domain of a hyperparameter can be real-valued (e.g., learning rate), integervalued (e.g., number of layers), binary (e.g., whether to use early stopping or not), or categorical (e.g., choice of optimizer).[2]

Given a data set $\mathcal{D}$, our goal is to find

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\operatorname{argmin}} \, \mathbb{E}_{(D_{train}, D_{valid}) \sim \mathcal{D}} \mathbf{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, D_{train}, D_{valid}),$$

where $\mathbf{V}(\mathcal{L}, \mathcal{A}_\lambda, D_{\text{train}}, D_{\text{valid}})$ measures the loss of a model generated by algorithm $\mathcal{A}$ with hyperparameters λ on training data D_{train} and evaluated on validation data D_{valid}.[2]

Methods of performing hyperparameter optimization include grid search, random search, Bayesian optimization, gradient-based optimization, and evolutionary optimization.

Grid search is an exhaustive search through a manually specified subset of the hyperparameter space of the learning algorithm. A grid search algorithm should be guided by some performance indicators measured by cross-validation or evaluation on a held-out validation set for the training set.

Random Search replaces the exhaustive enumeration of all combinations by selecting them randomly. In particular, when only a small number of hyperparameters affects the final performance of the machine learning algorithm, it outperforms grid search.

Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set. By iteratively evaluating a promising hyperparameter configuration based on the current model, and then updating it, Bayesian optimization, aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum.

Gradient-based optimization is possible to compute the gradient with respect to hyperparameters and then optimize the hyperparameters using gradient descent.

Evolutionary optimization uses evolutionary algorithms to search the space of hyperparameters for a given algorithm. It follows a process inspired by the biological concept of evolution.

3. OPEN-SOURCE HYPERPARAMETER OPTIMIZATION SOFTWARES

3.1 Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.[6]

Table 1 shows Applications, Algorithms, and Modules according to the functions of Scikit-learn. Classification identifies the category to which the object belongs. Regression is to predict the continuous value properties associated with an object. Clustering is the automatic grouping of similar objects into sets.

Dimension reduction is to reduce the number of random variables to consider. Model selection is the comparison, verification and selection of parameters and models. Preprocessing is feature extraction and normalization.[7]

Table 1. Functions, Applications, Algorithms, Modules of Scikit-learn

Function	Applications	Algorithms / Modules
Classification	Spam detection, Image recognition	SVM, nearest neighbors, random forest
Regression	Drug response, Stock prices	SVR, ridge regression, Lasso
Clustering	Customer segmentation, Grouping experiment outcomes	k-Means, spectral clustering, mean-shift
Dimensionality reduction	Visualization, Increased efficiency	PCA, feature selection, non-negative matrix factorization
Model selection	Improved accuracy via parameter tuning	grid search, cross validation, metrics
Preprocessing	Transforming input data such as text for use with machine learning algorithms	preprocessing, feature extraction

3.2 Tune

Tune is a Python library for hyperparameter tuning at any scale. Tune takes a user-defined Python function or class and evaluates it on a set of hyperparameter configurations. Tune includes the following features:[8]

- Launch a multi-node distributed hyperparameter sweep in less than 10 lines of code.
- Supports any machine learning framework, including PyTorch, XGBoost, MXNet, and Keras.
- Visualize results with TensorBoard.
- Choose among scalable SOTA algorithms such as Population Based Training (PBT), Vizier's Median Stopping Rule, HyperBand/ASHA.
- Tune integrates with many optimization libraries such as Facebook Ax, HyperOpt, and Bayesian Optimization and enables you to scale them transparently.

3.3 Hyperopt

Hyperopt is a Python library for serial and parallel optimization over awkward search spaces, which may include real-valued, discrete, and conditional dimensions. Currently Random Search, Tree of Parzen Estimators (TPE), Adaptive TPE algorithms are implemented in Hyperopt.[9]

3.4 Auto-sklearn

Auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator. Preprocessing in Auto-sklearn is divided into data preprocessing and feature preprocessing. Data preprocessing includes One-Hot encoding of categorical features, imputation of missing values and the normalization of features or samples. Feature preprocessing is a single transformer which implements for example feature selection or transformation of features into a different space.

Auto-sklearn supports parallel execution by data sharing on a shared file system. In this mode, the SMAC algorithm shares the training data for it's model by writing it to disk after every iteration. Because Auto-sklearn is mostly a wrapper around Scikit-learn, it is possible to follow the persistence example from Scikit-learn.[10]

3.5 BOCS

The Bayesian Optimization of Combinatorial Structures (BOCS) algorithm is a method for finding a global minimizer of an expensive-to-evaluate black-box function that is defined over discrete inputs given a finite budget of function evaluations. The algorithm combines an adaptive generative model and semidefinite programming techniques for scalability of the acquisition function to large combinatorial domains.

The BOCS algorithm is implemented in MATLAB and only requires the CVX package to be available in the local path for performing convex optimization. The scripts for comparing to other discrete optimization methods require an installation of SMAC, the python wrapper pySMAC and the bayesopt function in MATLAB for Bayesian Optimization with the Expected Improvement acquisition function.[11]

3.6 mlrMBO

mlrMBO is an R package for model-based/Bayesian optimization of black-box functions. mlrMBO includes the following features:[12]

- EGO-type algorithms (Kriging with expected improvement) on purely numerical search spacesMixed search spaces with numerical, integer, categorical and subordinate parameters
- Arbitrary parameter transformation allowing to optimize on, e.g., logscale
- Optimization of noisy objective functions
- Multi-Criteria optimization with approximated Pareto fronts
- Parallelization through multi-point batch proposals
- Parallelization on many parallel back-ends and clusters through batchtools and parallelMap

3.7 Scikit-optimize

Scikit-Optimize, or skopt, is a simple and efficient library to minimize (very) expensive and noisy black-box functions. It implements several methods for sequential model-based optimization. skopt aims to be accessible and easy to use in many contexts. The library is built on top of NumPy, SciPy and Scikit-Learn.[13]

3.8 FAR-HO

FAR-HO is a Python package containing Tensorflow implementations and wrappers for gradient-based hyperparamteter optimization with forward and reverse mode algorithmic differentiation. Aim of this package is to implement and develop gradient-based hyperparameter optimization(HO) techniques in TensorFlow, thus making them readily applicable to deep learning systems. This optimization techniques find also natural applications in the field of meta-learning and learning-to-learn.[14]

3.9 XGBoost

XGBoost is an open-source software library which provides a gradient boosting framework for C++, Java, Python, R, and Julia. It is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Kubernetes, Hadoop, SGE, MPI, Dask) and can solve problems beyond billions of examples. It is a novel evolutionary computation framework for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and

data structures transparent. It works in perfect harmony with parallelisation mechanisms such as multiprocessing and SCOOP.[15]

3.10 DEAP

DEAP is a Python framework for general evolutionary computation which is flexible and integrates with parallelization packages like scoop and pyspark, and other Python frameworks like sklearn via sklearn-deap. DEAP includes the following features:[16]

- Genetic algorithm using any imaginable representation
- List, Array, Set, Dictionary, Tree, Numpy Array, etc.
- Genetic programing using prefix trees
- Loosely typed, Strongly typed
- Automatically defined functions
- Evolution strategies (including CMA-ES)
- Multi-objective optimisation (NSGA-II, NSGA-III, SPEA2, MO-CMA-ES)
- Co-evolution (cooperative and competitive) of multiple populations
- Parallelization of the evaluations (and more)
- Hall of Fame of the best individuals that lived in the population
- Checkpoints that take snapshots of a system regularly
- Benchmarks module containing most common test functions
- Genealogy of an evolution (that is compatible with NetworkX)
- Examples of alternative algorithms : Particle Swarm Optimization, Differential Evolution, Estimation of Distribution Algorithm

3.11 DEvol

DEvol(DeepEvolution) is a Python package that performs Deep Neural Network architecture search using genetic programming. It is a basic proof of concept for genetic architecture search in Keras. The current setup is designed for classification problems, though this could be extended to include any other output type as well.[17]

4. ANALYSIS OF OPEN-SOURCE HYPERPARAMETER OPTIMIZATION SOFTWARES

In this section we compare the features of the open-source hyperparameter optimization softwares.

Table 2 shows the approache, interface, and feature of the open-source hyperparameter optimization softwares. Most software uses a variety of hyperparameter optimization approaches. Most of the interface uses python, but some use R or Matlab. Tune and hyperopt show distributed hyperparameter tuning, and Auto-sklearn and scikit-optimize are implemented using scikit-learn.

Some software, such as Auto-sklearn, Scikit-optimize, and FAR-HO, are built on top of other machine learning libraries. Tune and hyperopt provide distributed hyper parameter tuning, and DEAP supports parallelization packages.

Table 2. Features of Open-source Hyperparameter Optimization Softwares

Name	Approache	Interface	Feature
scikit-learn	Grid, Random	Python	built on top of SciPy
Tune	Grid, Random	Python	distributed hyper parameter tuning
hyperopt	Random	Python	distributed hyper parameter tuning
Auto-sklearn	Bayesian	Python	built around the scikit-learn
BOCS	Bayesian	Matlab	semidefinite programming
mlrMBO	Bayesian	R	model-based/Bayesian optimization
scikit-optimize	Bayesian	Python	built on top of NumPy, SciPy and Scikit-Learn
FAR-HO	Gradient-based	Python	meta-learning package based on TensorFlow
XGBoost	Gradient-based	C++, Java, Python, R, and Julia	gradient boosting framework
DEAP	Evolutionary	Python	integrates with parallelization packages
DEvol	Evolutionary	Python	Deep Neural Network architecture search using genetic programming

5. CONCLUSION

In this paper, we review open-source hyperparameter optimization softwares, and compare various characteristics of hyperparameter optimization softwares.

Because the performance of machine learning algorithms depends on how the hyperparameters are constructed, the use of appropriate open source hyperparameter optimization software can be instrumental in improving the performance of machine learning algorithms.

Choosing the right hyperparameter optimization algorithm depends on your ML problem, computing infrastructure, and associated computing budget.

Through review of open-source hyperparameter optimization software, we expect that using efficient hyperparameter optimization method to optimize a given machine learning method will greatly improve the efficiency of machine learning.

REFERENCES

[1] Yonghyuk Moon, Ikhee Shin, Yongju Shin, Okgi Min, "Recent Research & Development Trends in Automated Machine Learning," *Electronics and Telecommunications Trends*, Vol. 34 No. 4, pp.32-42, Aug 1, 2019. *https://dx.doi.org/10.22648/ETRI.2019.J.340404*

[2] F. Hutter et al. (eds.), "Automated Machine Learning*", The Springer Series on Challenges in Machine Learning,* 2019, *https://doi.org/10.1007/978-3-030-05318-5_1*

[3] Hyperparameter Optimization*, https://en.wikipedia.org/wiki/Hyperparameter_optimization.*

[4] Claesen, Marc, and Bart De Moor. "Hyperparameter Search in Machine Learning", *https://arxiv.org/abs/1502.02127*

[5] Hyperparameter, *https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning).*

[6] Scikit-learn, *https://en.wikipedia.org/wiki/Scikit-learn.*

[7] Scikit-learn, *https://scikit-learn.org/.*
[8] Tune, *https://ray.readthedocs.io/en/latest/tune.html.*
[9] Hyperopt, *https://github.com/hyperopt/hyperopt.*
[10] Auto-sklearn, *https://github.com/automl/auto-sklearn.*
[11] BOCS, *https://github.com/baptistar/BOCS.*
[12] mlrMBO, *https://github.com/mlr-org/mlrMBO.*
[13] scikit-optimize, *https://github.com/scikit-optimize/scikit-optimize.*
[14] FAR-HO, *https://github.com/lucfra/FAR-HO.*
[15] XGBoost, *https://github.com/dmlc/xgboost.*
[16] DEAP, *https://github.com/DEAP/deap.*
[17] DEvol, *https://github.com/joeddav/devol.*