**Hyperparameter Tuning in Python: a Complete Guide**

Choosing the correct hyperparameters for machine learning or deep learning models is one of the best ways to extract the last juice out of your models.

**What is the difference between parameter and hyperparameter?**

First, let’s understand the differences between a hyperparameter and a parameter in machine learning.

* **Model parameters**: These are the parameters that are estimated by the model from the given data. For example the weights of a deep neural network.
* **Model hyperparameters**: These are the parameters that cannot be estimated by the model from the given data. These parameters are used to estimate the model parameters. For example, the learning rate in deep neural networks.

Graphical user interface, table

Description automatically generated

**What is hyperparameter tuning and why it is important?**

Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within the limits you specify. This process once finished will give you the set of hyperparameter values that are best suited for the model to give optimal results.

It is an important step in any Machine Learning project since it leads to optimal results for a model. If you wish to see it in action, I’ve included a research paper on Moodle “Importance of Tuning Hyperparameters of Machine Learning Algorithms” that talks about the importance of hyperparameter optimization by experimenting on datasets.

**How to do hyperparameter tuning? How to find the best hyperparameters?**

Choosing the right combination of hyperparameters requires an understanding of the hyperparameters and the business use-case. However, technically, there are two ways to set them.

**Manual hyperparameter tuning**

Manual hyperparameter tuning involves experimenting with different sets of hyperparameters manually i.e. each trial with a set of hyperparameters will be performed by you. This technique will require a robust experiment tracker which could track a variety of variables from images, logs to system metrics.

There are a few experiment trackers that tick all the boxes. Neptune is one of them. It offers an intuitive UI and an open-source package Neptune-client to facilitate logging in your code. You can easily log hyperparameters and see all types of data results like images, metrics, etc. Here’s a link to the Neptune documentation: https://docs.neptune.ai/logging/what\_you\_can\_log/

Alternative solutions include W&B, Comet, or MLflow.

**Advantages of manual hyperparameter optimization**:

* Tuning hyperparameters manually means more control over the process.
* If you’re researching or studying tuning and how it affects the network weights, then doing it manually would make sense.

**Disadvantages of manual hyperparameter optimization**:

* Manual tuning is a tedious process since there can be many trials and keeping track can prove costly and time-consuming.
* This isn’t a very practical approach when there are a lot of hyperparameters to consider.

**Automated hyperparameter tuning**

Automated hyperparameter tuning utilizes already existing algorithms to automate the process. The steps you follow are:

* First, specify a set of hyperparameters and limits to those hyperparameters’ values (note: every algorithm requires this set to be a specific data structure, e.g. dictionaries are common while working with algorithms).
* Then the algorithm does the heavy lifting for you. It runs those trials and fetches you the best set of hyperparameters that will give optimal results.

**Hyperparameter tuning methods**

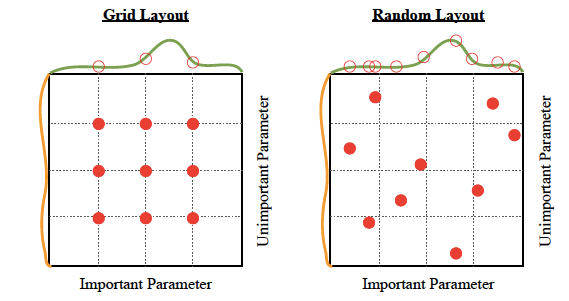
In this section, I will introduce some of the hyperparameter optimization methods that are popular today.

**Random Search**

In the random search method, we create a grid of possible values for hyperparameters. Each iteration tries a random combination of hyperparameters from this grid, records the performance, and lastly returns the combination of hyperparameters that provided the best performance.

**Grid Search**

In the grid search method, we create a grid of possible values for hyperparameters. Each iteration tries a combination of hyperparameters in a specific order. It fits the model on each and every combination of hyperparameters possible and records the model performance. Finally, it returns the best model with the best hyperparameters.



**Bayesian Optimization**

Tuning and finding the right hyperparameters for your model is an optimization problem. We want to minimize the loss function of our model by changing model parameters. Bayesian optimization helps us find the minimal point in the minimum number of steps. Bayesian optimization also uses an acquisition function that directs sampling to areas where an improvement over the current best observation is likely.

**Tree-structured Parzen estimators (TPE)**

The idea of Tree-based Parzen optimization is similar to Bayesian optimization. Instead of finding the values of p(y|x) where y is the function to be minimized (e.g., validation loss) and x is the value of hyperparameter the TPE models P(x|y) and P(y). One of the great drawbacks of tree structured Parzen estimators is that they do not model interactions between the hyper-parameters. That said TPE works extremely well in practice and was battle-tested across most domains. (https://optunity.readthedocs.io/en/latest/user/solvers/TPE.html)

**Hyperparameter tuning algorithms**

These are the algorithms developed specifically for doing hyperparameter tuning.

**Hyperband**

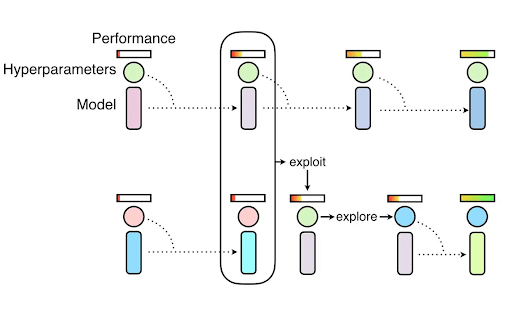
Hyperband is a variation of random search, but with some explore-exploit theory to find the best time allocation for each of the configurations. I’ve included a research paper on Moodle for further references.

**Population-based training (PBT)**

This technique is a hybrid of the two most commonly used search techniques: Random Search and manual tuning applied to Neural Network models.

PBT starts by training many neural networks in parallel with random hyperparameters. But these networks aren’t fully independent of each other.

It uses information from the rest of the population to refine the hyperparameters and determine the value of hyperparameter to try. You can check this link (https://www.deepmind.com/blog/population-based-training-of-neural-networks) for more information on PBT.

[*Source*](https://deepmind.com/blog/article/population-based-training-neural-networks)

**BOHB**

BOHB (Bayesian Optimization and HyperBand) mixes the Hyperband algorithm and Bayesian optimization. You can check this link (https://www.automl.org/blog\_bohb/) for further reference.

**Tools for hyperparameter optimization**

Now that you know what the methods and algorithms are let’s talk about tools, and there are a lot of those out there.

**Some of the best hyperparameter optimization libraries are:**

1. Scikit-learn
2. Scikit-Optimize
3. Hyperopt
4. Ray.tune
5. Talos
6. BayesianOptimization
7. Metric Optimization Engine (MOE)
8. Spearmint
9. SigOpt
10. Fabolas

**1. Scikit-learn**

Scikit-learn has implementations for grid search and random search and is a good place to start if you are building models with sklearn.

For both of those methods, scikit-learn trains and evaluates a model in a k fold cross-validation setting over various parameter choices and returns the best model.

Specifically:

* **Random search:** with randomsearchcv runs the search over some number of random parameter combinations
* **Grid search:**gridsearchcv runs the search over all parameter sets in the grid

Tuning models with scikit-learn is a good start but there are better options out there and they often have random search strategies anyway.

**2. Scikit-optimize**

Scikit-optimize uses a Sequential model-based optimization algorithm to find optimal solutions for hyperparameter search problems in less time.

Scikit-optimize provides many features other than hyperparameter optimization such as:

* store and load optimization results,
* convergence plots,

**3. Hyperopt**

Hyperopt is one of the most popular hyperparameter tuning packages available. Hyperopt allows the user to describe a search space in which the user expects the best results allowing the algorithms in hyperopt to search more efficiently.

Currently, three algorithms are implemented in hyperopt.

* Random Search
* Tree of Parzen Estimators (TPE)
* Adaptive TPE

To use hyperopt, you should first describe:

* the objective function to minimize
* space over which to search
* the database in which to store all the point evaluations of the search
* the search algorithm to use

This link will walk you through how to structure the code and use the hyperopt package to get the best hyperparameters. (https://github.com/hyperopt/hyperopt/wiki/FMin)

**4. Ray Tune**

Ray Tune is a popular choice of experimentation and hyperparameter tuning at any scale. Ray uses the power of distributed computing to speed up hyperparameter optimization and has an implementation for several states of the art optimization algorithms at scale.

Some of the core features provided by ray tune are:

* distributed asynchronous optimization out of the box by leveraging Ray.
* Easily scalable.
* Provided SOTA algorithms such as ASHA, BOHB, and Population-Based Training.
* Supports Tensorboard and MLflow.
* Supports a variety of frameworks such Sklearn, XGBoost, TensorFlow, PyTorch, etc.

You can refer to this tutorial (<https://docs.ray.io/en/latest/tune/tutorials/overview.html>) to learn how to implement ray tune for your problem.

**5. Keras Tuner**

Keras Tuner is a library that helps you pick the optimal set of hyperparameters for your TensorFlow program. When you build a model for hyperparameter tuning, you also define the hyperparameter search space in addition to the model architecture. The model you set up for hyperparameter tuning is called a *hypermodel*.

You can define a hypermodel through two approaches:

* By using a model builder function
* By subclassing the HyperModel class of the Keras Tuner API

You can also use two pre-defined HyperModel classes – HyperXception and HyperResNet for computer vision applications.

**6. BayesianOptimization**

BayesianOptimization is a package designed to minimize the number of steps required to find a combination of parameters that are close to the optimal combination.

This method uses a proxy optimization problem (finding the maximum of the acquisition function) which, although it’s still a hard problem, it’s cheaper in the computational sense, and common tools can be employed. Therefore, Bayesian Optimization is most adequate for situations where sampling the function to be optimized is a very expensive endeavour.

**7. Metric Optimization Engine**

MOE (Metric Optimization Engine) is an efficient way to optimize a system’s parameters when evaluating parameters is time-consuming or expensive.

It is ideal for problems in which

* the optimization problem’s objective function is a black box, not necessarily convex or concave,
* derivatives are unavailable,
* and we seek a global optimum, rather than just a local one.

This ability to handle black-box objective functions allows us to use MOE to optimize nearly any system, without requiring any internal knowledge or access.

**8. Spearmint**

Spearmint is a software package that also performs Bayesian optimization. The software is designed to automatically run experiments (thus the code name spearmint) in a manner that iteratively adjusts a number of parameters so as to minimize some objectives in as few runs as possible.

**9. SigOpt**

SigOpt fully integrates automated hyperparameter tuning with training runs tracking to give you a sense of the bigger picture and the path to reach your best model.

With features like highly customizable search spaces and multimetric optimization, SigOpt can advance your model with a simple API for sophisticated hyperparameter tuning before taking it into production.

**10. Fabolas**

While traditional Bayesian hyperparameter optimizers model the loss of machine learning algorithms on a given dataset as a black box function to be minimized, FAst Bayesian Optimization on LArge data Sets (FABOLAS) models loss and computational cost across dataset size and uses these models to carry out Bayesian optimization with an extra degree of freedom.

**Hyperparameter tuning resources and examples**

Here are some hyperparameter tuning examples implemented for different ML and DL frameworks.

Random forest hyperparameter tuning

* [Understanding Random forest hyperparameters](https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74)
* [Bayesian hyperparameter tuning for random forest](https://blog.dominodatalab.com/hyperopt-bayesian-hyperparameter-optimization/)
* [Random forest tuning using grid search](https://www.kaggle.com/sociopath00/random-forest-using-gridsearchcv)

XGBoost hyperparameter tuning

* [XGBoost hyperparameters tuning python](https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/)
* [XGBoost hyperparameters tuning in R](https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/)
* [XGBoost hyperparameter using hyperopt](https://www.kaggle.com/alexandrnikitin/xgboost-hyperparameter-optimization)
* [Optuna hyperparameter tuning example](https://www.kaggle.com/corochann/optuna-tutorial-for-hyperparameter-optimization)

LightGBM hyperparameter tuning

* [Understanding LightGBM parameters](https://neptune.ai/blog/lightgbm-parameters-guide)
* [LightGBM hyperparameter tuning example](https://www.kaggle.com/mlisovyi/lightgbm-hyperparameter-optimisation-lb-0-761)
* [Optuna for LIghtGBM hyperparameter tuning](https://tech.preferred.jp/en/blog/lightgbm-tuner-new-optuna-integration-for-hyperparameter-optimization/)

CatBoost hyperparameter tuning

* [Catboost hyperparamters overview](https://catboost.ai/docs/concepts/parameter-tuning.html)
* [Grid search for catboost hyperparameter tuning](https://effectiveml.com/using-grid-search-to-optimise-catboost-parameters.html)

Keras hyperparameter tuning

* [Hyperparameter tuning using Keras- tuner example](https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html)
* [Keras CNN hyperparameter tuning](https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner)
* [How to use Keras models in scikit-learn grid search](https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)
* [Keras Tuner: Lessons Learned From Tuning Hyperparameters of a Real-Life Deep Learning Model](https://neptune.ai/blog/keras-tuner-tuning-hyperparameters-deep-learning-model)

PyTorch hyperparameter tuning

* [Hyperparameter tuning for Pytorch](https://towardsdatascience.com/rocking-hyperparameter-tuning-with-pytorchs-ax-package-1c2dd79f2948)