Tool Comparison: Optuna vs Hyperopt

Audrey Bertin, Nasko Apostolov, Nelson Evbarunegbe, & Raymond Li

Introduction

For our project, we will be conducting a comparison of two tools designed to help with the process of **hyperparameter tuning** in a machine learning pipeline.

Hyperparameters are user-selected parameters applied to learning algorithms that affect how they are implemented–for example, constraints, weights, learning rates, number of algorithm layers/complexity, etc. Depending on which set of hyperparameters is selected, the results and performance of a particular machine learning algorithm can vary dramatically. The process of hyperparameter tuning is that of finding a good set of parameters for a specific model and dataset combination.

The possible search space is infinite, making this task challenging. Grid search, in which users pre-define a list of possible parameter combinations and test them one by one, has historically been a common way to do this. However, it's very subjective (completely up to the user to select which parameters they want to try), and can be very time consuming for testing many combinations.

Hyperparameter tools have been developed to assist with this, using advanced algorithms and parallel computing to allow users to test out parameters with a more mathematical and faster approach.

In this project, we are interested in comparing how two separate hyperparameter tuning tools work and perform–both in comparison to each other and to the standard grid search.

These tools are:

- **Hyperopt:** a popular Python library for hyperparameter optimization that has been around–and been considered industry standard–for more than a decade.
- **Optuna**: a much newer tool that has some advanced features and is gaining popularity.

Both tools have a UI-based component (either visualizations or a full dashboard) and both have their code publicly available on GitHub:

- https://github.com/hyperopt/hyperopt

- https://github.com/optuna/optuna

They also both have publicly available documentation sites:

- http://hyperopt.github.io/hyperopt/
- https://optuna.org

We will evaluate these tools in two separate ways:

1. A **code review**, where we look at the repositories and note things that they are each doing well or not so well.
2. An **experimental evaluation**, where we test an actual hyperparameter tuning process using sample data to compare ease of use, UI features, and which tool creates the best final model.

Code Review

For the code review, we will look at the GitHub pages and documentation sites for each of the tools. We will consider both features involving general repository health, as well as more code-specific items such as non-functional requirements.

For general repository health we will look at the following features of each GitHub repository:

- **Popularity/Use** – how many stars and active users does it have?
- **Community** – how active is the community (how many contributors are there, pull requests, etc).
- **Maintenance** - how often are changes made to the main branch? When was the last release? Are issues/pull requests getting handled or left behind?
- **Dependencies** – how many software dependencies does each project have?
- **Security** – are there any potential security vulnerabilities?

For non-functional requirements, we will look at the following subset:

- **Understandability** – how easy is it for someone unfamiliar with the code to understand what is going on?
- **Testability/Debuggability** - is the code formatted in such a way that it is easy to test features or debug when a problem occurs (modular, throws exceptions, etc)? What does the current test coverage look like?

- **Extensibility/Scalability** - is it easy to add on new features or contribute to the project (since it's open source)? Are the existing features written/provided in such a way that it is easy to implement the code at scale, such as in some sort of production environment at a company, or are external libraries or new features needed?

For each of these categories we will rate the repository's performance on a scale from 1-10. This score is subjective and will be backed up by arguments for both what the repository is doing well, and where it could be improved.

At the end, we will share a table summarizing the ratings in each category to compare each tool's strengths and weaknesses (code-wise). We will also sum up the scores to determine an overall best tool based on code quality.

Experimental Evaluation

Related Work/References