Sherpa: Hyperparameter Optimization for Machine Learning Models

Lars Hertel
Department of Statistics
University of California, Irvine
lher1@uci.edu

Julian Collado
Department of Computer Science
University of California, Irvine
colladou@uci.edu

Peter Sadowski
University of Hawaii
peter.sadowski@hawaii.edu

Pierre Baldi
Department of Computer Science
University of California, Irvine
pfbaldi@ics.uci.edu

Abstract

Sherpa is a free open-source hyperparameter optimization library for machine learning models. It is designed for problems with computationally expensive iterative function evaluations, such as the hyperparameter tuning of deep neural networks. With Sherpa, scientists can quickly optimize hyperparameters using a variety of powerful and interchangeable algorithms. Additionally, the framework makes it easy to implement custom algorithms. Sherpa can be run on either a single machine or a cluster via a grid scheduler with minimal configuration. Finally, an interactive dashboard enables users to view the progress of models as they are trained, cancel trials, and explore which hyperparameter combinations are working best. Sherpa empowers machine learning researchers by automating the tedious aspects of model tuning and providing an extensible framework for developing automated hyperparameter-tuning strategies. Its source code and documentation are available at [https://github.com/LarsHH/sherpa](https://github.com/LarsHH/sherpa) and [https://parameter-sherpa.readthedocs.io/](https://parameter-sherpa.readthedocs.io/) respectively. A demo can be found at [https://youtu.be/L95sasMLgP4](https://youtu.be/L95sasMLgP4).

1 Existing Hyperparameter Optimization Libraries

Hyperparameter optimization algorithms for machine learning models have previously been implemented in software packages such as Spearmint [15], HyperOpt [2], Auto-Weka 2.0 [9], and Google Vizier [5] among others.

Spearmint is a Python library based on Bayesian optimization using a Gaussian process. Hyperparameter exploration values are specified using the markup language YAML and run on a grid via SGE and MongoDB. Overall, it combines Bayesian optimization with the ability for distributed training.

HyperOpt is a hyperparameter optimization framework that uses MongoDB to allow parallel computation. The user manually starts workers which receive tasks from the HyperOpt instance. It offers the use of Random Search and Bayesian optimization based on a Tree of Parzen Estimators.

Auto-WEKA 2.0 implements the SMAC [6] algorithm for automatic model selection and hyperparameter optimization within the WEKA machine learning framework. It provides a graphical user interface and supports parallel runs on a single machine. It is meant to be accessible for novice users and specifically targets the problem of choosing a model. Auto-WEKA is related to Auto-Sklearn [4] and Auto-Net [11] which specifically focus on tuning Scikit-Learn models and fully-connected...
Table 1: Comparison to Existing Libraries

<table>
<thead>
<tr>
<th></th>
<th>Spearmint</th>
<th>Auto-WEKA</th>
<th>HyperOpt</th>
<th>Google Vizier</th>
<th>Sherpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Stopping</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dashboard/GUI</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Distributed</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td># of Algorithms</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

neural networks in Lasagne, respectively. Auto-WEKA, Auto-Sklearn, and Auto-Net focus on an end-to-end automatic approach. This makes it easy for novice users, but restricts the user to the respective machine learning library and the models it implements. In contrast our work aims to give the user more flexibility over library, model and hyper-parameter optimization algorithm selection.

Google Vizier is a service provided by Google for its cloud machine learning platform. It incorporates recent innovation in Bayesian optimization such as transfer learning and provides visualizations via a dashboard. Google Vizier provides many key features of a current hyperparameter optimization tool to Google Cloud users and Google engineers, but is not available in an open source version. A similar situation occurs with other cloud based platforms like Microsoft Azure Hyperparameter Tuning and Amazon SageMaker’s Hyperparameter Optimization.

2 Need for a new library

The field of machine learning has experienced massive growth over recent years. Access to open source machine learning libraries such as Scikit-Learn, Keras, Tensorflow, PyTorch, and Caffe allowed research in machine learning to be widely reproduced by the community making it easy for practitioners to apply state of the art methods to real world problems. The field of hyperparameter optimization for machine learning has also seen many innovations recently such as Hyperband, Population Based Training, Neural Architecture Search, and innovation in Bayesian optimization. While the basic implementation of some of these algorithms can be trivial, evaluating trials in a distributed fashion and keeping track of results becomes cumbersome which makes it difficult for users to apply these algorithms to real problems. In short, Sherpa aims to curate implementations of these algorithms while providing infrastructure to run these in a distributed way. The aim is for the platform to be scalable from usage on a laptop to a computation grid.

3 Key Features

3.1 Choice of Algorithms

A key motivation for Sherpa is to provide implementations of recent hyperparameter tuning algorithms to users while making it easy to add new algorithms. The currently implemented algorithms are:

- Random Search
- Grid Search
- Local Search: increasing or decreasing one hyperparameter at a time
- Bayesian Optimization using a Gaussian Process and Expected Improvement Acquisition function
- Population Based Training (PBT)

The documentation provides tutorials for using Bayesian Optimization, PBT, and Local Search and how to implement a new hyperparameter search algorithm.

Sherpa also allows to implement stopping rules, decision rules which allow stopping of underperforming trials. A trial is the training of a model with a specific set of hyperparameters. Sherpa currently implements a Median-Stopping-Rule [5] which automatically stops any trial with lower performance than the median of the finished trials at the same epoch. Additional information can be found in the documentation.

3.2 Scalable Computation via Plug-In Schedulers

Sherpa is meant to be flexible around the user’s computational resources. It can simply be run in a single Python session such as a Jupyter Notebook, or in parallel using two scripts. The first Python script then runs the Sherpa optimization and uses the Sherpa scheduler to submit jobs. A second Python script implements the trial evaluation. A database is automatically run in the background to receive metrics from trials. Sherpa submits the desired number of parallel trials and submits a new one whenever a previous trial finishes. Schedulers can execute jobs locally via subprocesses, or remotely via a grid engine such as SGE [12]. This gives the user an easy way to coordinate and run many parallel trials and it is easy to implement new schedulers.

3.3 Visualization Dashboard

Sherpa provides an interactive web-based dashboard to keep the user informed about the progress of the hyperparameter optimization. It has a table of all completed trials including their best performance and hyperparameter configuration. A line chart shows the sequence of objective values for each trial against its training iterations. Finally, a parallel coordinates plot allows the user to explore completed trials. This is very useful for noticing trends among useful configurations allowing faster and more successful hyperparameter explorations.

Figure 1: The dashboard provides a parallel coordinates plot (top) and a table of finished trials (bottom left). Trials in progress are shown via a progress line-plot (bottom right).

Furthermore, the dashboard provides an interface to interact with the optimization while it is running. A stopping button allows the user to stop trials during their training. If the user notices a trial performing badly via the line chart, she can decide to stop it and free up the resource to train a new trial.
References


