HyperparameterDB

Ashish Kumar, Kailash Nadkar, Tripti Santani
{kumar.ashi, nadkar.k, santani.t}@husky.neu.edu
INFO 6105, Spring 2019, Northeastern University

Abstract- The performance of many machine learning methods depends critically on hyperparameter settings. Sophisticated Bayesian optimization methods have recently achieved considerable successes in optimizing these hyperparameters, in several cases surpassing the performance of human experts. However, blind reliance on such methods can leave end users without insight into the relative importance of different hyperparameters and their interactions. Our aim is to create a hyperparameter database by running numerous hyperparameter values over three of public datasets and calculating the individual conditional expectation of every hyperparameter on the quality of a model. Objective is to choose a set of optimal hyperparameters for a learning algorithm. In this paper, the dataset is for two classification and one regression problem.

1. Introduction

1.1 Background

Hyperparameters are variables that we need to set before applying a learning algorithm to a dataset. In machine learning scenarios, a significant part of model performance depends on the hyperparameter values selected. The goal of hyperparameter exploration is to search across various hyperparameter configurations to find the one that results in the optimal performance. The challenge with hyperparameters is that there are no magic number that works everywhere. The best numbers depend on each task and each dataset. The hyperparameter database to be developed as a part of this project is an open resource with algorithms, tools, and data that allows users to visualize and understand how to choose hyperparameters that maximize the predictive power of their models. Phase I of the project involves selecting a unique dataset containing predicted target variables, hyperparameters, meta-data etc. by running different models (with varying hyperparameters) on it using H2O AutoML.

Hyperparameters can be divided into 2 categories:

1. Optimizer hyperparameters

They are related more to the optimization and training process - If our learning rate is too small than optimal value then it would take a much longer time (hundreds or thousands) of epochs to reach the ideal state

If our learning rate is too large than optimal value, then it would overshoot the ideal state and our algorithm might not converge.

2. Model specific Hyperparameters

They are more involved in the structure of the model

Currently, the hyperparameter database analyzes the effect of hyperparameters on the following algorithms: Distributed Random Forest (DRF), Generalized Linear Model (GLM), Gradient Boosting Machine (GBM). Naïve Bayes Classifier, Stacked Ensembles, XGBoost and Deep Learning Models (Neural Networks).

1.2 Dataset

THREE DATASETS IS BEING USED FOR ANALYSIS.

OLS REGRESSION CHALLENGE

1- This is a regression dataset. The project is to determine predict the cancer mortality rates for US counties. The dataset consists of 34 columns and 3047 observations.

TELCO CUSTOMER CHURN

2-Telco Customer Churn- To Predict behavior to retain customers. Here we analyze all relevant customer data and develop focused customer retention programs. The raw data contains 7043 rows (customers) and 21 columns. This is a classification dataset. The project is to determine predict whether the customer will leave the service or not.

3- Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. The data set shared here contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. This is a classification dataset. The project is to determine predict whether it is pulsar star or not.

2.Algorithm and code source

H2O is an open source predictive analytics platform for data scientists and business analysts who need scalable and fast machine learning. Unlike traditional analytics tools, H2O provides a combination of extraordinary math and high-performance parallel processing with unrivaled ease of use. H2O intelligently combines the following features to solve today's most challenging business problems:

- Best of breed Open Source Technology
- Easy-to-use WebUI and Familiar Interfaces
- Data Agnostic Support for all Common Database and File Types
- Massively scalable Big Data Munging and Analysis
- Real-time Data Scoring

The H2O python module is not intended as a replacement for other popular machine learning frameworks such as scikit-learn, pylearn2, and their ilk, but is intended to bring H2O to a wider audience of data and machine learning devotees who work exclusively with Python.

H2O from Python is a tool for rapidly turning over models, doing data munging, and building applications in a fast, scalable environment without any of the mental anguish about parallelism and distribution of work.

In this project, the dataset is cleaned and then passed to H2O to get all the possible models. The models are generated for various runtime. In this project the dataset has been run for 333, 777,999,1333 and 1555 seconds.

H2O connects with its server to generate various models. The log file is generated to save all the logs created in the process. Log files will help to track the actions and the results in the future.

As mentioned earlier, each model does not generate separate meta-data. For every run-time a single metadata is generated which is stored in a separate file.

The dataset is passed to H2O AutoML algorithm and the leaderboard is generated which displays all the algorithms used to generate the models.

Once the leaderboard is obtained, we have stored the leaderboard in json format. Every model is stored in the database for future reference. The hyperparameters of the best model are found using further analysis techniques.

The models generated in the leaderboard for the algorithm along with its hyperparameters is extracted for a specific algorithm and is as follows:

```
df_gbm3
```

	Model_name	Leaderboard_rank	learn_rate	learn_rate_annealing	max_abs_leafnode_pred	pred_noise_bandwidth
0	GBM_grid_1_AutoML_20190426_065209_model_5	1	0.050	1.0	1.797693e+308	0.0
1	GBM_grid_1_AutoML_20190426_065209_model_15	4	0.050	1.0	1.797693e+308	0.0
2	GBM_grid_1_AutoML_20190426_065209_model_4	5	0.100	1.0	1.797693e+308	0.0
3	GBM_grid_1_AutoML_20190426_065209_model_13	6	0.080	1.0	1.797693e+308	0.0

Next, we predicted the variable importance of the predictors for all runtime across all datasets to get insight of the important predictors as described below:

	Variable	relative_importance	scaled_importance	percentage
0	Contract_Month-to-month	760.058289	1.000000	0.295374
1	tenure	421.066925	0.553993	0.163635
2	InternetService_Fiber optic	259.032623	0.340806	0.100665
3	MonthlyCharges	254.001038	0.334186	0.098710
4	tenure_group_Tenure_0-12	215.121246	0.283033	0.083601
5	TotalCharges	140.344391	0.184650	0.054541

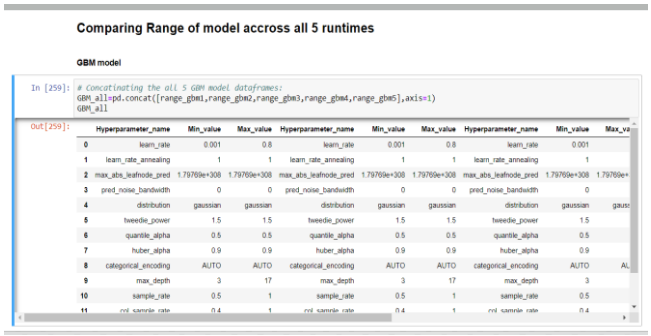
Variable Importance for each runtime is also imported in json format.

We, then, aimed to do analysis of the minimum and maximum values to get the range of hyperparameters for a specific runtime as below:

```
# Range of values for hyperparameters of GBM along with
get_rang(df_gbm11)
```

	Hyperparameter_name	Min_value	Max_value
0	learn_rate	0.001	0.8
1	learn_rate_annealing	1	1
2	max_abs_leafnode_pred	1.79769e+308	1.79769e+308
3	pred_noise_bandwidth	0	0
4	tweedie_power	1.5	1.5
5	quantile_alpha	0.5	0.5

Once range of values of hyperparameters for a particular runtime is obtained, we used these values and compared the range of values across the models for all hyperparameters as below:



Algorithm designers wanting to manually assess hyperparameter importance often investigate the local neighbourhood of a given hyperparameter configuration: vary one hyperparameter at a time and measure how performance changes. Note that the only information obtained with this analysis is how different hyperparameter values perform in thecontextofasingleinstantiationoftheotherhyperpa rameters. Optimally, algorithm designers would like to know how their hyperparameters affect performance in general, not just in the context of a single fixed instantiation of the remaining hyperparameters, but across all their instantiations.

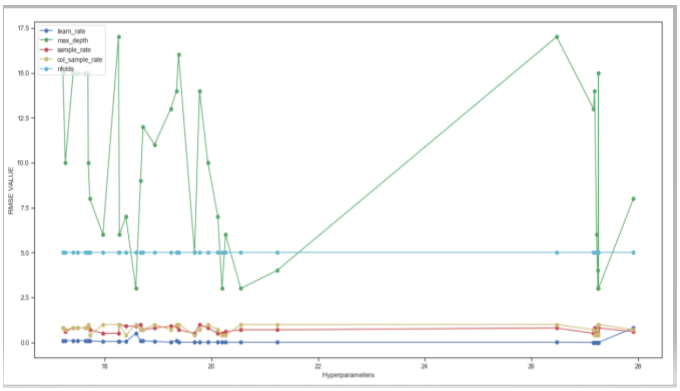
For this purpose, we implemented below mentioned methods:

1) GridSearch CV: Exhaustive search over specified parameter values for an estimator.

2) Stepwise Regression: Performed stepwise forward linear regression with target as evaluation metric and

4. Observation:

Upon plotting graph for variation in hyperparameters against evaluation metrics, which in case of regression is RMSE, is as below:



From GridSearch CV, we inferred that the value of residual deviance, keep on decreasing for parameters 'learn rate', 'sample_rate', 'max_depth' and 'col_sample_rate' as their value gets tuned which implied that the evaluation metric and the above-mentioned hyperparameters are related. Thereby, proving that these hyperparameters are important for the GLM model.

For the below parameters, p value is achieved to be less than 0.05 while implementing Forward Stepwise Regression:

Hyperparameter	p-value
huber_alpha	0.00230766
nbins	0.00230766
quantile_alpha	0.00230766
max_after_balance_size	0.00230766
tweedie_power	0.00230766
nbins_top_level	0.00230766
nbins_cats	0.00230766
histogram_type	0.00230766
categorical_encoding	0.00230766
learn_rate_annealing	0.00230766
col_sample_rate_change_per_level	0.00230766
stopping_tolerance	0.00230766
learn_rate	0.00494899

5. Conclusion:

We conducted a study towards our contribution to Skunkworks Hyperparameter database and tried to find the range of hyperparameters for each runtime and compared it across all the five runtimes. We

also implemented Stepwise Forward Selection to assess important hyperparameters for Gradient Boosting Machine algorithm. We implemented GridSearchCV for Genralized Linear Model (GLM) and Gradient Boosting Machine(GBM) algorithm and observed that the change in lambda and alpha for GLM and col_sample_rate learn_rate and sample_rate for GBM are reflected on residual deviance.

6. Future Scope:

With the advent of automated machine learning, automated hyperparameter optimization methods are by now routinely used. However, this progress is not yet matched by equal progress on automatic analyses that yield information beyond performance-optimizing hyperparameter settings.

We can use variance decomposition techniques to identify a hyperparameter subspace that captures most of the potential for improvements over a good configuration. **Functional ANOVA** approach rates as important hyperparameters that capture the potential for improvement. The functional ANOVA framework for assessing hyperparameter importance of algorithms is based on efficient computations of marginal performance. Various hyperparameter configurations of an algorithm result in various performances. Functional ANOVA determines per hyperparameter how it contributes to the variance in performance

Functional ANOVA takes as input performance data gathered with different hyperparameter settings of the algorithm (for example using the H2O AutoML) and assess how important each of the hyperparameters and each low-order interaction of hyperparameters is to performance.

The methods introduced in this work offer a principled, scientific way for algorithm designers and users to gain deeper insights into the way in which design choices controlled by hyperparameters affect the overall performance of a given algorithm. In future work, we plan to extend our approach to detect dominated hyperparameter values and interactions between instance characteristics and hyperparameter settings. We can also develop configuration procedures that determine important hyperparameters on the fly and exploit the information to speed up the optimization process.

With this approach, we can prove that the performance variability is often largely caused by few hyperparameters that define a subspace to which we can restrict configuration.

Finally, we can aim to use the knowledge obtained from hyperparameter importance analysis to prune the hyperparameter search space

7. Acknowledgment:

We want to thank Prof. Nick Brown for his constant support and guidance during this project. We would also like to thank Prabhu Subramanium, manager of the project, who has helped us on every step and guided us wherever required.

8. References:

[1]https://github.com/nikbearbrown/INFO_6105/blob/master/Projects-Portolios/NBA%20MVP%20Prediction%20with%20Principal%20Component%20Analysis/Project/Protfolio%20Blog.ipynb
[2]https://github.com/prabhuSub/Hyperparamter-Samples
[3] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html
[4]https://www.kaggle.com/dansbecker/how-models-work
[5] https://data.world/nrippner/ols-regression-challenge
[6]https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction
[7] https://www.h2o.ai/blog/hyperparameter-optimization-in-h2o-grid-search-random-search-and-the-future/
[8]https://machinelearningmastery.com/configure-gradient-boosting-algorithm/
[9] http://ceur-ws.org/Vol-1998/paper_09.pdf
[10]http://proceedings.mlr.press/v32/hutter14.pdf