Predicting Higgs Boson with Machine Learning

Alec Flowers 321786, Alexander Glavackij 322968, Janet van der Graaf 327759
Machine Learning Course CS-433, EPFL, Switzerland

***Abstract*—The goal of the Higgs boson challenge is to classify observed events as signals or background noise. We develop a pipeline which pre-processes the provided data set and implements six machine learning regression methods. We evaluate the performance of each method and use these performances as baseline for the optimization of the regularized logistic regression. Through pre-processing, feature selection, and polynomial expansion we were able to improve the classification accuracy from 0.76 to 0.80.**

I. Introduction

The Higgs boson is an elementary particle in the Standard Model of particle physics and was theorized over 50 years ago [1]. It was discovered by the ATLAS project at CERN [2]. The ATLAS project provides simulation data which is used to optimize the discovery of Higgs boson particles. The Higgs boson machine learning challenge makes the simulation data available to the public. The goal of this challenge is to use the given data set to predict whether an observed event was background noise or a signal event, i.e. a Higgs boson particle [2]. We structure our sections by the structure of our pipeline we implemented. First, we present the data set and how we pre-processed it. Next, we detail the process of selecting the best hyper parameters to develop baselines. In Section IV we discuss steps we took to optimize including iterative feature selection and learning curve comparisons to improve upon our baseline results by 4%. Lastly, we draw a conclusion and present future work.

Requirements: The requirements for this project are given in [3]. We are required to implement the following Machine Learning methods: *Least Squares Gradient Descent* (GD), *Least Squares Stochastic Gradient Descent* (SGD), *Least Squares* (LS), *Ridge Regression* (Ridge), *Logistic Regression* (LR), and *Regularized Logistic Regression* (RLR). We can use a competition platform to submit predictions and compute test scores.

II. Pre-Processing

The data set is comprised of 30 features and $350k$ entries. The features describe raw measured values and derived values selected by physicists of the ATLAS project. Missing entries are denoted by a value of -999, correct entries range from -18.066 to 4974.97. 181886 rows have at least one missing entry, therefore, it is not feasible to drop rows with missing entries as this would decrease the size of the data set drastically. To be able to use these rows in training, we impute the missing entries by calculating the median of the corresponding feature and replacing the missing value with the median. Another property of the data set are the different scales of the values for each feature/column. The values of column 12 range from 0 to 1, while for column 5 they range from 13.602 to 4974.979. To equalize the distributions of each column, we standardize each column by subtracting the mean and dividing by the standard deviation. This improves the numerical stability of the gradient descent and makes the columns comparable. Note that standardization and imputation need to be done during the cross validation. Otherwise, the training folds in a cross validation iteration would contain information of the test fold (since the median and standardization were calculated over all values including the testing fold), which results in data leakage. We also only use medians, means and standard deviations from the training data set to keep our test conditions as close as possible to the situation where we only have past information and will be classifying future, unknown information.

Method	Hyperparameters	
GD	$d \in \{1, \mathbf{2}\}$	$\gamma \in \{0.1, 0.01, 0.001, \mathbf{0.005}, 1e-4, 1e-5\}$
SGD	$d \in \{\mathbf{1}, 2\}$	$\gamma \in \{0.1, 0.01, 0.001, \mathbf{0.005}, 1e-4, 1e-5\}$
LS	$d \in \{1, \mathbf{2}\}$	
Ridge	$d \in \{1, \mathbf{2}\}$	$\lambda \in \{\mathbf{0.001}, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5\}$
LR	$d \in \{1, \mathbf{2}\}$	$\gamma \in \{\mathbf{1e\text{-}4}, 1e-5, 1e-6, 1e-7\}$
RLR	$d \in \{1, \mathbf{2}\}$	$\gamma \in \{\mathbf{1e\text{-}4}, 1e-5, 1e-6, 1e-7\}$ $\lambda \in \{0.001, \mathbf{0.1}\, 0.2, 0.3, 0.4, 0.5\}$

Table I: Hyperparameters of every method mentioned in Section I. d denotes the degree of the polynomial expansion, γ denotes the learning rate in gradient descent, λ denotes the regularization parameter. In bold, best performing hyperparameters are selected.

III. Hyperparameters

We find the best performing set of hyperparameters for a specific model (Table I, bold) with a grid search: we generate all possible combinations of values for a hyperparameter set. We then determine the performance of every hyperparameter allocation with 4-fold cross validation. The best performing allocation is the one which yields the highest average accuracy over all folds. For every model, we save the best obtained hyperparameter configuration in a file to be used for training of the respective model. After determining the best performing hyperparameters, we can train the methods given in Section I. During cross validation, we calculated the weights for every fold combination. To achieve a more robust model and reduce the variance, we train the models with the best hyperparameters on the whole data set. We then store the calculated weights in a file to be used for generating the predictions with the test set.

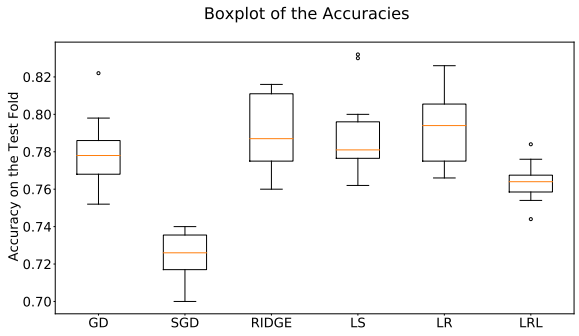
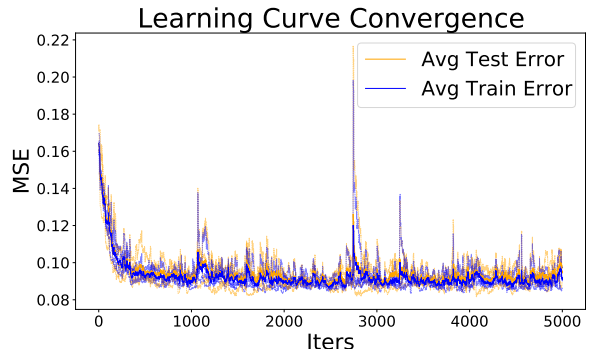
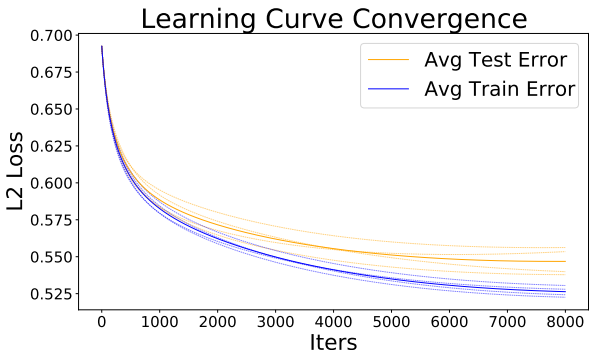


Fig. 1: Boxplots showing the accuracy for each model on the test fold during 10-fold cross validation.



(a) Loss plot showing convergence for SGD.



(b) Loss plot showing overfitting for RLR.

Fig. 2: Plots showing the MSE for SGD and L^2 loss for LRL during cross-validation.

IV. Results

We evaluate the performance of the models with a 10-fold cross validation on the full data set, cf. Figure 1. SGD shows the lowest accuracies, even though it converges, cf. Figure 2a. Logistic regression is built for binary classification to fix the problems of linear regression such as sensitivity to unbalanced data and the prediction value being continuous. Therefore, we chose to optimize LRL to take advantage of logistic regression and regularization to punish large weights which indicate over fitting. The data set contains 30 features, some of which are derived features. As a result, it is likely that some features correlate. Using correlated features in regressions and classifiers yields ill-conditioned regression weight matrices, which induce numerical instabilities. Additionally, a simpler model is preferable to a more complex one. The learning curve for LRL indicates overfitting, cf. Figure 2b. To combat these issues, we conduct a correlation analysis of the features to decide which columns we can drop. In selecting which feature to keep we had a bias towards keeping derived features versus primitive ones [2]. We dropped 8 columns $\{5, 6, 12, 21, 26, 27, 28, 29\}$ which had high correlations with $\{4, 9\}$ and re-ran many of our grid-searches. We noticed almost no losses in accuracy, but did not see any improvements over our baseline attempts. We were also looking to improve the speed of grid search but found dropping only 8 columns gave us marginal speed gains especially when using polynomial expansion.

Our next step was to drop the 18 primitive columns and keep only the 12 derived columns. Considering the 12 derived columns are computed from the primitives we figured we can keep the same amount of information, and therefore keep our accuracy score high, while significantly reducing the feature space. We can now take advantage of polynomial expansion without the weight matrix exploding as we are only expanding 12 instead of 30 columns.

Our best model yet is submission 92765: regularized logistic regression on the 12 columns with $d = 2$, $\gamma = 1e-6$, $\lambda = 0.0$ for 3000 iterations which gave us a submission score of 0.802.

V. Conclusion

The goal of this work was to implement the six machine learning methods given in Section I and use these methods to predict the occurrence of Higgs boson particles. We developed a pipeline which executes pre-processing on the data set, finds the best hyperparameters for each method via a grid search, trains each model and computes predictions. Additionally, we analyzed the convergence of each method by examining the loss curves of the cross validations. We evaluated the accuracy of each method and further optimized the LRL method. Our final test score on the submission platform is 0.802. Future work could include further feature selection, iteratively dropping the least significant coefficient and refitting the model. Instead of manually selecting features, future work could also include a principal component analysis (PCA) to reduce the high-dimensionality of the data set.

References

[1] P. Onyisi. (2012). Higgs boson faq, [Online]. Available: https://wikis.utexas.edu/display/utatlas/Higgs+boson+FAQ (visited on 10/24/2020).

[2] C. Adam-Bourdariosa, G. Cowanb, Ć. Germainc, G. Isabelle, K. Baĺazs, and D. Rousseaua. (2014). Learning to discover: The higgs boson machine learning challenge, [Online]. Available: https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf (visited on 10/24/2020).

[3] N. Flammarion and M. Jaggi. (2020). Class project 1, [Online]. Available: https://raw.githubusercontent.com/epfml/ML_course/master/projects/project1/project1_description.pdf (visited on 10/24/2020).