Higgs boson: Machine Learning Challenge

Sarah Antille, Lilia Ellouz, Zeineb Sahnoun
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—This paper reports our attempt to address the Higgs boson machine learning challenge as part of the Machine Learning course taught at EPFL: based on the observations we have, we try to predict whether they are caused by the decay of the Higgs boson, which we call ”signal”, or by the decay of other know particles, which we refer to as ”background”.**

I. Introduction

The data set we worked with was simulated by the ATLAS experiment from CERN. Protons were smashed together in hope to produce a Higgs boson particle. However, scientists cannot really observe this particle but only its decay signature through different measurements [1]. Our goal is to use these measurements (features) to infer whether it was a Higgs boson.

To do so, we applied different machine learning algorithms seen in class for classification problems. We mainly used different kind of linear regressions before moving on to logistic regressions. Before selecting the model that maximized the accuracy on the test set, we cleaned the raw data and proceeded to do feature engineering.

The following sections explain our methodology in more details.

II. Exploratory Data analysis and Feature Processing

A. Exploratory Data analysis

Our training data set consists of 250’000 data points with 30 features and their corresponding labels (-1 for ”background” and 1 for ”signal”).
We started by plotting histograms for the different features:

- We noticed that all variables are floating point, except PRI_jet_num which is integer taking values in $\{0,1,2,3\}$. From the challenge documentation [1], we understood that some features only made sense for a specific number of JETs. Hence we decided to split our data set into three distinct subsets having PRI_jet_num respectively equal to 0, 1, 2 & 3. We could see afterwards which features were undefined for certain subsets (had only values corresponding to -999) and we dropped those from the corresponding subset.
- After that, we noticed that most of our features still had significant portions of missing values. We decided to replace those values with the median as it is more robust to outliers.
- Also, a large number of the distributions of our features were positively skewed which gave us the idea to apply a log transformation to those later on.

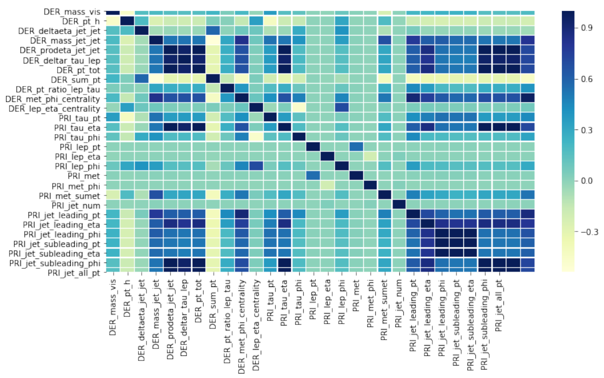

Fig. 1. Correlation matrix of our features.

Afterwards, we tried to plot a pie chart for the number of data points from each label. We noticed that our data set was not perfectly balanced (34.3% labeled signal against 65.7% labeled background). Hence we tried downsampling the background labeled data points for each of our subsets. This approach did not not improve the accuracy of our predictions, therefore we decided not to use it.

Furthermore, we plotted a correlation matrix which showed that some of our features were highly correlated. Hence we tried to drop some of them, but this approach again did not improve the accuracy of our predictions so we decided not to use it as well.

B. Feature processing

1) Log transformation
Some of our features have positively skewed distributions, hence we initially applied the log function only to those features but noticed that stacking the log of every feature using the formula below was the approach that gave us the best results. We came up with the formula to be able to apply the logarithm transformation to features which had negative values. And we perform the addition (+1) because small compact values in the interval [0,1] will make our log transform left skewed.[2]

$$X_n = log(X_n - min_{k=0}^{N}(X_n^k) + 1)$$

where X_n is the feature indexed by n and X_n^k is the value of the feature for data point k.

2) Polynomial expansion
Adding the log transformed features considerably improved our predictions but we still weren’t satisfied with the results we got, so we tried performing feature augmentation using polynomial expansion. We tested different values and found that degree = 7 yields the best accuracy overall.

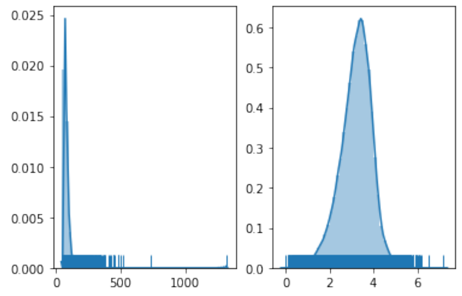

Fig. 2. Log transformation applied to DER_sum_pt feature.

3) Data Standardization
We used standardization on all of our features to remove multicollinearity and interactions between features.

III. Models and choices of implementation

We were asked to implement and use the methods seen in class, so we applied them on the already cleaned and processed data set. To test our models, we did a random $80-20$ train-validation split on each subset of the train data set.
Table I summarizes their testing accuracy on the validation set.

Methods	**Accuracy (%)**		
	subset 1	**subset** 2	**subset** 3
Gradient Descent	84	76.8	79.6
Stochastic Gradient Descent (batch size = 100)	72	49.7	52
Least Squares (normal equations)	85	78.9	83
Ridge Regression (cross-validated)	84	78	82
Logistic Regression	74	72	74.6
Penalized Logistic Regression	79.9	72.9	73.5

TABLE I
Methods used and Accuracy on validation sets

For all of these methods, we classified data by testing different thresholds for each of the three subsets and choosing the one with best accuracy. If the predicted value Xw was superior to the threshold, the decided label was 1 , otherwise it was -1. The optimal threshold was determined by scanning 1000 values between -0.5 and 0.5 for a given set of weights and comparing the accuracy for each of these values.

The selected learning rates were: $0.1, 0.05, 0.05$ for GD (for each subset respectively), 0.01 for SGD, and 0.1 for logistic regression and its regularized version.

For polynomial augmentation, we tested degrees ranging from 2 to 10 and found out that degree=7 improved considerably the accuracy of our predictions. Higher degrees didn't improve on the accuracy that our model already reached.

In order to test how well our model would generalize to unknown data, we used a k-fold cross-validation method. We split the data into k subsets of the same size, we trained the model on $k-1$ subsets and tested on the k^{th} one. This operation was done k times, and the resulting test root mean-square errors were averaged. The value $k = 4$ seemed to balance the computational speed and the test accuracy.

We cross-validated different regularization terms for ridge regression as illustrated in Figure 3, which helps minimize

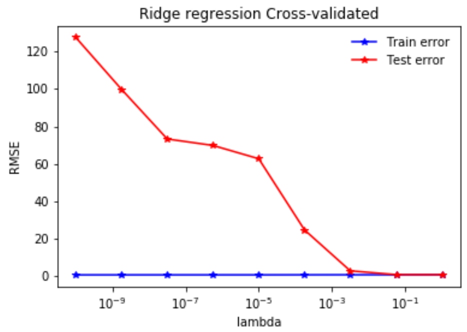

Fig. 3. Cross-validating lambda for subset 1 ridge regression

overfitting[3]. This led us to select $\lambda = 10^{-2}$ for subset 1 and $\lambda = 10^{-5}$ for subsets 2 and 3.

We then determined the weights again by re-running the method with the selected lambda. We used the same method to decide on the optimal lambda for regularized logistic regression, and selected $\lambda = 1$ for the three subsets.

IV. Results & Discussion

Combining feature processing and ridge regression, the best accuracy we have obtained on aicrowd was **0.81%**.

The resulting accuracy on validation sets of the implementations are reported in table I.

Gradient descent, least squares, and ridge regression were the most successful methods, with least squares getting a 85% accuracy on subset 1. However, gradient descent and least squares were not regularized, which means that this is probably a result of overfitting. In contrast, ridge regression performed better, since it is equivalent to a regularized version of least squares. This hypothesis is supported by the results of one of our submissions on aicrowd that used a very small value for $\lambda = 10^{-9}$: the accuracy was 0.6% due to overfitting.

Linear regression performing better than logistic regression seemed odd, considering the fact that logistic regression is mainly implemented for classifiable outcomes, and that the Higgs boson problem precisely requires a binary outcome.

In hindsight, we could go through more feature engineering to explore the limitations of regularized logistic regression. Since the logistic model is fit by computing the loss gradient and updating weights in an iterative way, the computational speed difference between logistic and linear regression was significantly different, which made cross-validation for the logistic model difficult.

V. Conclusion

Throughout the course of this project, we realized that simply using machine learning methods or raw data is not enough to get an optimal accuracy. Steps like cleaning and processing data, deriving new features and selecting the parameters for each model are as important as selecting the best method to guarantee optimal results. We were also confronted to the trade-off between computational speed and parameter optimization when we tried to optimize the somewhat slow penalized logistic regression.

As a future work, exploring how we can obtain a better classification with logistic regression could be a good lead.

REFERENCES

[1] "Learning to discover: the higgs boson machine learning challenge." [Online]. Available: https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf

[2] "Log-transformation and its implications for data analysis." [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/

[3] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*. Washington, D.C.: John Wiley & Sons, New York: V. H. Winston & Sons, 1977.