EPFL Machine Learning Higgs 2019 – Spot the Boson
CS-433 Machine Learning – Project 1

Louis Amaudruz, Andrej Janchevski, Timoté Vaucher
IC, EPFL, Lausanne, Switzerland

***Abstract*—In this paper we take a look at the discovery of the Higgs boson particle at the LHC at CERN from a machine learning point of view. To this end, we first use standard data processing pipelines which consists of standardizing and cleaning the data before augmenting it with polynomial basis vectors. Several models are then trained to classify them between either coming from a Higgs boson particle or not. The best classifier we found is a Logistic regression model which yields a 0.836 accuracy on the test set.**

I. Introduction

At CERN, the experiments claimed the discovery of the Higgs boson particle in 2012. The Higgs boson has many different processes through which it can decay and when it decays, it produces other particles.
The experiments which led to the discovery consisted of smashing two protons into one another at high speed which produced fast decaying particles with some probability of it being a Higgs boson particle. The speed of the decay made it nearly impossible to directly observe the particle resulting from the collision and thus the goal was to find a function that maps decay signatures as positive signals or background noise [1].
The final goal is to use these features calculated from each collision, to develop a classifier. This is also the main goal of the Higgs Boson Challenge, the basis for this project, where a sample dataset of 250,000 training samples with 30 of these features was given.

II. Data processing

In our case as also commonly found, we noticed that the performance of the models depends mostly on the choice of preprocessing steps and as such we focused on greatly refining this process.

- **Splitting based on `PRI_jet_num`**: After inspecting the data and documentation we observed that this feature is categorical, with only 4 different values (0, 1, 2, 3). In addition, some of the other features are undefined for certain values. As such, we decided to split the dataset into 4 subsets based on this feature and train 4 different models;
- **Dealing with undefined values**: The value -999.0 for any feature vector represents an undefined instance. First, we remove all of the columns in the data where each value is undefined or null. Then, any remaining NaNs in a few feature vectors are set to 0;
- **Standardization**: All of the features are normalized using Z-normalization. This type of normalization assumes that the data has a Gaussian distribution, hence we also attempted using min-max normalization as an alternative, however we observed better results with the first method;
- **Removal of correlated features**: Using Pearson's correlation coefficients we detect pairs of highly linearly correlated features and the feature vector with a lower variance of the two is removed. By inspection, we observed that the best threshold value to use for the coefficients is 0.85 (in absolute value);
- **Augmentation with polynomial basis vectors**: As a feature augmentation step we add additional feature vectors to the data calculated by raising each feature to a degree power. The maximum degree to use is designated as a hyperparameter to be validated during training. Cross terms of the second degree (i.e. $x_i \cdot x_j$) and a bias term are also included;

III. Models and methods

A. Training algorithms

The mandatory six training algorithms were implemented based on the instructions in the lecture notes and code from the exercises. We used only two of them for this task.
Ridge regression was used with mean-square-error L2 regularization. The optimal weights are computed directly using the normal equations.
Regularized logistic regression was used wit the sigmoid function. To approximate the optimal weights Newton's iterative optimization method is used.
SVM - We simulate the training of a SVM classifier [2] by using the gradient descent iterative method to minimize the Hinge loss function, with included L2 regularization. The Hinge loss and its gradient can be computed using the following equations:

$$\begin{aligned} \mathcal{L}(w) &= \max\{0, 1 - y \cdot (Xw)\} + \frac{\lambda}{2}||w||^2 \\ \nabla\mathcal{L}(w) &= \lambda w + \begin{cases} -X^T y, & y \cdot (Xw) < 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

For all of these algorithms the regularization parameter λ was designated as a parameter to be validated during training. The initial value of the learning rate parameter γ was set to a fixed value determined to be optimal by inspection, but we also perform weight decay for SVM and Regularized Logistic Regression.

B. *Cross validation*

In order to tune the two hyperparameters we perform 5-fold cross-validation. For each parameter combination training and validation accuracy was recorded across each fold. The best parameter combination was chosen based on maximal mean validation accuracy, however the standard deviation of the accuracy was also considered and different parameters were chosen if the deviation was too high.
Using this technique and also by inspection it was discovered that the optimal values for λ are:

- Ridge regression : $[10^{-6}, 10^{-4}]$;
- Regularized Logistic Regression : $[10^{-15}, 10^{-12}]$ or 0;
- SVM : $[10^{-4}, 10^{-2}]$

The maximum polynomial degree was set between 5 and 13.

IV. Results and discussion

We first performed 5-fold cross validation on the training set. Each model was trained on the same folds in order to allow for the comparison to be valid. The results of one such iteration of the validation process is shown on Figure 1.

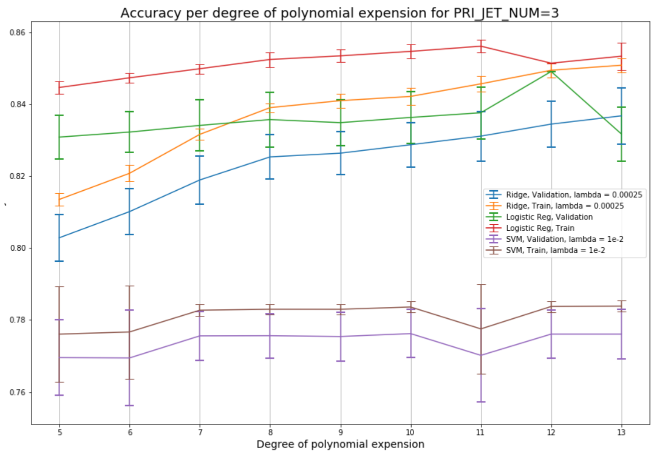

Figure 1. Example results of 5-fold cross validation for `PRI_jet_num`=3

What we found surprising is that SVM was the worst performer, whereas we assumed that it would be working better as the algorithm takes into account the margin as the data is not linearly separable. One of our hypothesis is that the algorithm didn't converge in the set number of steps for the cross validation. Our best performing classifier was across all folds and models the Logistic Regression as can be seen in Figure 2. From the results we can also observe that the choice of splitting the data using `PRI_jet_num` was valid.
We tried two different orders of preprocessing steps yielding the highest difference in submission performance (from 0.804 to 0.836). We noticed that for Ridge regression it was better to set the undefined values to 0 before performing z-normalization and then expanding the features, whereas for the other models such a policy resulted in unstable results. Changing the pipeline to first perform z-normalization, polynomial expansion and then finally setting the undefined values to 0 increased stability and range of usable learning rates.

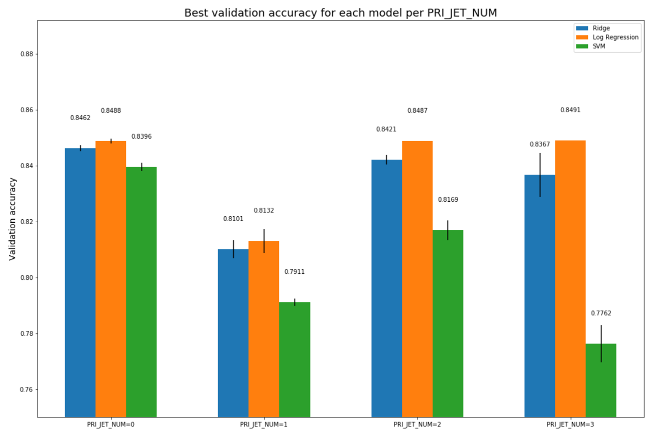

Figure 2. Comparison of best validation accuracy across the different subsets and algorithms

V. Conclusion

In this project, we approached different basic machine learning models to tackle a binary classification task, while paying a lot of attention to the data processing and finding ways to validate the models with cross-validation. We finally selected 4 models of Logistic Regression for our final submission that scored 0.836 on the AIcrowd leaderboard.

In future work and provided with more time and computing power at our disposal, it would have been interesting to investigate more into Gradient Boosting and Neural networks methods as those were the top methods on the original Kaggle competition.

References

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau. (2014) Learning to discover: the higgs boson machine learning challenge. [Online]. Available: http://higgsml.lal.in2p3.fr/documentation

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.