Report machine learning project 1

Victor Le, João Silveira, Santiago Garcia
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—In this paper, we explain the methodology followed to try to solve the challenge of finding the Higgs Boson with the machine learning techniques learnt in class. We talk about the feature preprocessing, algorithm used, and final results of our model.**

I. Introduction

The aim of this project is to develop and test a machine learning model meant to solve the challenge of finding the Higgs Boson. The dataset contains several features that describe collision events. From that data we predict whether each event originated a Higgs Boson.

For this binary classification we use the methods learnt in class and try to improve on them using several techniques which will be discussed further. This is the main focus of this report since it is the part of the project that enables us to achieve better results upon finding the most effective learning method for our problem.

II. Exploratory data analysis

We observed that in the provided datasets, train and test, 11 columns of 30 contain omitted data.

The dataset contains more -1 predictions than 1. Indeed, in Kaggle, the submission sample which always predict -1 regardless of the features, performs 67%. We are facing an imbalanced classification problem.

There are two types of features: Primitive, raw values measured by the detector, and derived, computed from the primitive features by the knowledge domain expert.

All the data types are floating numbers except the PRI_jet_num column, which contains categorical data, the value is an integer that can only be equal to 0, 1, 2 or 3.

III. Cleaning the data : Splitting the data

To use machine learning techniques, clean data is required. As said before, only 19 columns contain fully meaningful values. If we decided to work with only these columns, we would be losing 36.6% (11/30) of the potential features.

Fortunately, there is a logic for the appearance of missing data, it depends on the PRI_jet_num value. For example, if the latter is equal to 1, only 7 columns will all contain NaN (-999) values. So we decided to split the dataset into subsets by grouping by the PRI_jet_num column which gives us 4 groups of data. However, inside of each group there is still a difference, the column DER_mass_MMC can have real values or missing data. So applying the same process as before, we group by the presence of missing data or not, ending with 8 subsets of the original data (2 for each PRI_jet_num group).

Finally, in each group, we removed the columns that have low variance. For example in the $PRI\_jet\_all\_pt$ column in the $PRI\_jet\_num = 1$ group, all the values are equal to 0. This is an unnecessary feature that does not provide any relevant information.

IV. Machine learning method used

We tried all the machine learning methods seen in class and we decided on using the ridge regression with least squares to address the problem. This method provides us the direct solution, in contrast with gradient descent which requires the computation of several iterations. We need a method that is fast to compute since we have to build a model for each subset of the data. The advantage of using ridge regression instead of the more simple least squares method is that it helps to avoid overfitting by penalizing the model by adding the L2-norm to the loss function to calculate the optimal weights.

V. Model comparison

To decide whether a model is better than another we rely on the accuracy indicator (number of correct predictions divided by the number of all predictions made). In order not to overfit the dataset, we use 5-fold cross validation. We could have increased the k to have a better generalization but we concluded that $k = 5$ is a correct split as increasing k too much, also increases the computation time.

VI. Feature processing

To make our model fit complex functions, we used multiple features generation, for some of the non NaN columns we applied the following functions:

- Polynomial : $x -> x^n$
- Combination : $x_i, x_j -> x_i x_j / i \neq j$
- Others functions : $x -> log(x)^n$ or $tanh(x)^n$ or $\sqrt{x}$

The usage of these functions or not and the order n, are part of the hyper-parameters for each model.

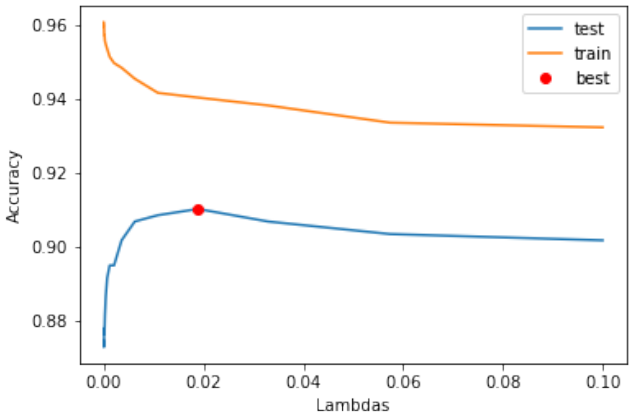

Figure 1. Accuracy following the lambdas for subset $PRI\_jet\_num = 2$ and $DER\_mass\_MMC = -999$

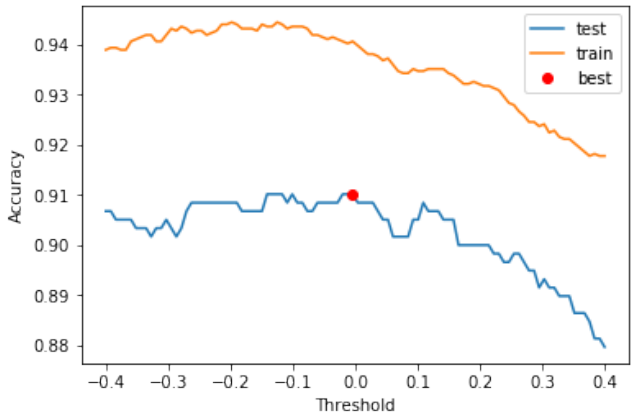

Figure 2. Accuracy following the prediction threshold for subset $PRI\_jet\_num = 2$ and $DER\_mass\_MMC = -999$

VII. Hyper-parameter tuning

Since we are using linear regression to classify, there is a prediction threshold which determines the output class. If the value is above the threshold, it outputs 1 otherwise -1. We consider this threshold as a hyper-parameter for each model. It can be a way to address the problem of imbalanced dataset. But on the other hand, if we apply this on a subset that is not imbalanced, we may overfit the data. This hyper-parameter is relatively dangerous to tune. We see on figure 2 that the best threshold is near 0.

Lambda is a parameter between 0 and 1 that multiplies the euclidean norm of the cost function to avoid overfitting. The closest this parameter is to 0, the more similar the function is to a least squares function. As we can see on figure 1, a value of lambda is selected for which the accuracy of the test set is maximized.

For variable combinations, we decided to combine mainly the derived variables. We calculated mutual information of each feature, which gives us a good indication of their importance, with respect to the prediction and applied the combinations to the top of each subset. We decided to keep the rest of the variables too, because as said on [1], a variable that is completely useless by itself can provide a significant performance improvement when taken with others and if interdependence is suspected, expand your feature set by constructing conjunctive or products of features.

VIII. Results

The parameters used to obtain the results are shown on table I. With these parameters we obtained an accuracy of 0.83791 on the public leaderboard of the Kaggle competition.

IX. Possible improvements

As we said in the section II, the dataset is imbalanced. We found in the literature that there are several possibilities to fixing this problem. We could have used over-sampling or under-sampling to balance the dataset. Another thing that can improve the comparison between two models is to use the F1-score, which takes into account the precision and the recall. It provides a more realistic measurement of the classifier performance.

Subset	Lambda	Threshold	Train Accuracy	Test Accuracy
0,0	7.2789e-05	0	0.8155	0.8145
0,1	1e-08	0	0.9504	0.9499
1,0	default	0	0.8064	0.8028
1,1	0.1373	0	0.9316	0.9215
2,0	2.3950e-07	0	0.8453	0.8395
2,1	0.0417	-0.03232	0.9394	0.9044
3,0	7.2789e-05	0	0.8451	0.8326
3,1	0.5298	0	0.9450	0.9288

Table I
Optimal Hyperparameters

X. Summary

The goal of the project was to try to develop a machine learning model that is able to solve the challenge of finding the Higgs Boson. The model used was ridge regression, in which a small value of the euclidean norm was added to avoid overfitting of the data. Before applying the model to optimize the weights, data cleaning and feature processing was done. An exploratory analysis gave the clue to subset the data in categories, and after normalizing the data, new features, such as polynomials and combinations of variables, were created. For each subset of the data, a different model was optimized, changing the hyper-parameters slightly to maximize accuracy. The combination of the different models, depending on the best features available, gave the best result we could achieve.

References

[1] Isabelle Guyon and Andr Elisseeff. An introduction to variable and feature selection. 3:1157–1182.