The Higgs Boson Machine Learning Challenge

Matteo Calafà, Giulia Mescolini, Paolo Motta
First Project for the ”Machine Learning” course at EPFL Lausanne, Switzerland

***Abstract*—The report contains a proposal of solution for the Higgs Boson Machine Learning Challenge, proposed in the framework of the ”Machine Learning” course at EPFL Lausanne. Several algorithms are presented to approach this classification problem on CERN particle accelerator data.**

I. Introduction

The goal of the challenge is to estimate the likelihood that a given event’s signature is the result of a Higgs boson or of some other process/particle. This is because, rather than observing the boson directly, scientists measure the products that result from its decay process which may be similar to other particles’ ones.
In section II, we present the analysis of the database and the meticulous preprocessing; afterwards, in section III, we present the models built with the 6 requested algorithms and the selection of the best hyper-parameters; finally, in section IV, we illustrate their performance.

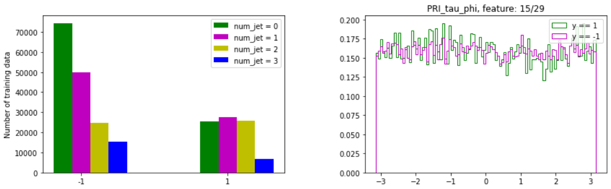
II. Preprocessing

A. First Analysis of the Dataset

The dataset contains 250000 points for training and 568238 for testing with 30 features and their corresponding binary labels (“-1” for “background” and “1” for “signal”), which clearly have to be predicted for the test set values.
First of all, we notice that one feature, *PRI_jet_num*, is the only one to be categorical; it represents the number of jets (showers of hadrons originating from a quark and a gluon, clustered together after being produced in a particle collision) and it ranges from 0 to 3. Inspired by the challenge documentation, we noticed that some features are meaningless for some values of jets, therefore we have split the dataset into 4 subclasses, each one characterized by a different *PRI_jet_num*.

B. Management of Missing Values

From the documentation, we know that each “-999” value in the dataset consists in a missing value; firstly, we decided not to consider features presenting more than 70 % missing data. Then, the remaining missing values have been replaced by the median of the feature which is, according to theory, a robust estimator.



(a) Labels by *PRI_jet_num*   (b) 15th feature for 1^{st} group

Figure 1: Exploratory plots on the training dataset

C. Standardization

In order to ensure a good functioning of the numeric optimization, it is a good practice to standardize the dataset: we subtracted from each feature its mean and divided by its standard deviation. This helps the feature matrix in having a better condition number. Moreover, this is a good practice to balance the weights of all the features for certain operations such as the computation of the distance in the *K-nearest-neighbors method*.

D. Feature Engineering

We plotted the features and ideated strategies to deal with their peculiarities. Two relevant examples of empirical distribution plots are shown in Figure 1.

- **Logaritmic transform:** for positive features, we computed a transformation into $\log(1 + x)$, which helps in reducing or removing the skewness of our original data. Since the distribution of some of our continuous features is non-normal, we applied this strategy to make the data as “normal” as possible.
- **Useless features:** from the observation of the plots (e.g. Figure 1a), we noticed that the empirical distribution of some features does not change with the label, therefore we simplified our model by not considering them.
- **Symmetrical features:** we computed the absolute value of features in columns 14,17,23,26[1], which have a symmetric distribution with respect to 0.
- **Angles:** the features in columns 15,18,20,24,27[1] represent measures of angles. The applied strategy to adapt the periodicity in a regression study was to replace this column with the $\cos(x)$ and $\sin(x)$ transformations.

[1] Note that the columns after the 22th have their ID decreased by one, since the *PRI_jet_num* column has been deleted

However, since the sine feature turns out to be ineffective, it has been discarded to avoid overfitting and only the cosine transformation has been kept.

E. *Polynomial Feature Expansion*

This technique improves the representation power of linear models. For each feature, in addiction to its powers from 1 to M (whose choice is described later), we added as further features its square root and cubic root. Moreover, we included all the pairwise products.
The polynomial degree is one of the hyper-parameters of our model; its optimal value M is found performing 3-fold cross-validation.

F. *Management of Outliers*

To deal with the presence of outliers, we fixed $\alpha = 0.1$ and decided to cap the extreme values of each feature to the α-quantile (for the lower tail) and to the $(1 - \alpha)$-quantile (for the upper tail).

III. Models and Methods

After the pre-processing on the dataset, we implemented several models to solve the classification task which employ linear models and logistic regression. Note that, as stated in II-A, we used only the train data with a specific *PRI_jet_num* to predict the label of test data belonging to the same group. The methods considered are:

- **Gradient Descent**
- **Stochastic Gradient Descent**,
- **Least Squares** with Normal Equations
- **Logistic Regression**

We implemented the regularized versions as well in order to reduce overfitting:

- **Ridge Regression**
- **Lasso Logistic Regression** (Logistic with L1-regularization)
- **Regularized Logistic Regression with L2 norm**

The optimal hyper-parameter λ has been chosen with a 3-fold cross-validation. In Figure 2, we report the accuracy trends, which lead to our choice of hyper-parameters.
It is noteworthy to notice that a *FISTA* method has been implemented to replace and improve in terms of computational time the standard *Gradient Descent* when the reduction of the computational cost is essential, for instance in the previous cross-validation study.

We also implemented the K-nearest classification algorithm, which provides an accuracy of 0.799; nevertheless, this method is not recommended as the ones presented in Table IV because of the high-dimensionality of the problem (= 30) and the high computational time. It is however true that, without an in-depth preprocessing, it turned out to be more accurate than the logistic regression methods. The reason lies in the fact that, when features are in different forms and are not adapted for a logistic model, a spacial method is more suitable because of its simplicity and its lack of any model assumptions.

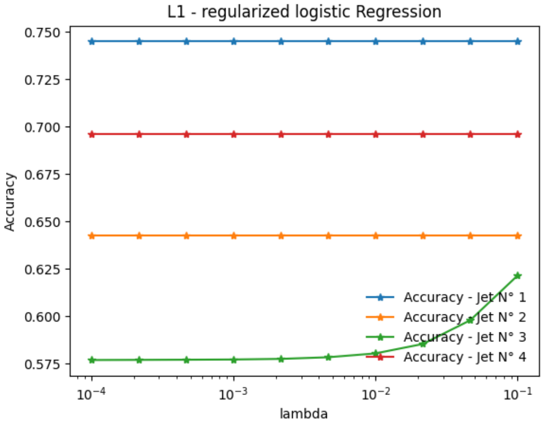


Figure 2: *RMSE* errors and accuracies obtained through a cross-validation for different values of λ (L^1 regularizer).

IV. Results

We report the accuracy results obtained for each method with the optimal choice of hyper-parameters:

Method	Test Accuracy
Least Squares GD	0.690
Least Squares SGD	0.695
Least Squares Normal Eqs.	0.740
Ridge Regression	0.719
Logistic Regression	0.797
Reg. Logistic Regression (L2)	0.832
Lasso Logistic Regression	0.836

V. Conclusion

The best performance, with an accuracy on the test set of 0.836, was obtained with the Lasso Regression; this highlights how even standard models can be powerful in solving complex classification tasks, and how regularization can help to reduce overfitting. In addiction, we want to remark that a relevant part of the success of the task is up to the data preprocessing and interpretation.