ML Project 1: spot the Higgs boson

Manuel Leone, Gabriele Macchi, Marco Vicentini
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—Machine Learning techniques are becoming increasingly popular nowadays and they are used in several and different fields of science. These techniques are useful when we deal with complex and high dimensional datasets, which sometimes represent the output of scientific experiments. In this project we implement from scratch some of the most popular methods, applying them to a dataset from CERN, to solve a problem of binary classification.**

I. Introduction

This project aims to find the best binary classification method to distinguish signals of the Higgs Boson from the background noise, and use this results to predict the correct label for each new observation. We used the CERN's dataset which summarizes collision experiments; the raw dataset includes 250.000 labeled events with 30 features each. First of all, we had to clean this huge amount of data via data cleaning and features selection, because a well done preprocessing is a fundamental step to obtain good results in a machine learning project. Then we proceeded with the implementations of the mandatory machine learning methods such as Linear Regression, Ridge Regression, and Logistic Regression. For more informations about implementation and discussion of these algorithms see Sections II-A and II-C.

II. Models and methods

Implementing good machine learning methods requires skills in the fields of *data analysis* and *algorithm design*. Understanding the real meaning behind the data is an important step to choose the right direction in the following implementations. On the other hand, the broad panorama of ML algorithms in literature provides several possible implementations, that if not well implemented could change dramatically the performance of the algorithms. Our analysis will focus on these two aspects, that in this case are theory physic and pure statistical approach. Before getting inside the developing process, the reader is advised that a preliminary **dataset division** step is performed. Indeed, in order to have an idea of the goodness of fit and to compare the methods, the algorithms were trained on 80% of the available dataset and tested on the remaining 20%.

A. Basic ML implementations

The implementation of the base methods consists of **six** well known Machine Learning algorithms such as Gradient Descent, Stochastic Gradient Descent, Least Squares, Ridge Regression, Logistic Regression and Regularized Logistic Regression. The null values of the dataset were initially filled with the median of the respective column. The first computation concerned the entire dataset considered as one. In the table below different hyper-parameters and results of the respective methods are shown. The accuracy achieved is computed both on our test set and AIcrowd submissions. These results, as expected, **were not satisfactory**, so we decided to move on with data processing, as described in the next section.

	Hyper-parameters Used					
Methods	λ	γ	Degree	Max Iterations	Accuracy (*20% test*)	Accuracy (*AIcrowd*)
Gradient Descent	/	10^{-6}	1	2000	0.718	0.413
Stochastic GD	/	10^{-7}	1	1000	0.704	0.412
Least Squares	/	/	1	/	0.741	0.705
Ridge Regression	10^{-8}	/	5	/	0.724	0.736
Logistic Regression	/	10^{-10}	1	10000	0.736	0.414
Reg. Logistic Regression	10^{-8}	10^{-10}	1	10000	0.736	0.414

TABLE I
Accuracies of the six algorithms without feature processing

B. Dataset analysis and preprocessing

The analysis of the dataset has been performed in different steps and each of them led to gradually improve the methods performances. The examinations we carried out are the following:

1) NaN analysis

The official documentation [1] reports the dataset variables as either primitive (`PRI_`), raw quantities directly measured from the collisions detector, or derived (`DER_`), which are computed by the ATLAS physicist using the primitives. This information is useful to understand the physics meaning behind another fundamental aspect: the dataset presents many **-999.0** as placeholders for missing values. So, our first aim was to understand how we could impute these *NaN*.
The analysis of the documentation shows a strict correlation between *NaN* and the `PRI_jet_num` quantity. The jets are, from our understanding of the physic background, the sub particles produced in a collision. The main point to understand here is that some quantities are intentionally undefined because impossible to compute or no-sense with a number of jet equal to 0 or 1. With this cardinal concept in the head, we proceeded to a dataset categorization in four different subsets based on the `PRI_jet_num`. As a result, we produced only columns totally full or empty from *NaN* values, with the only exception of the `DER_mass_MMC` one.
At this point, we dropped all the columns totally filled with *NaN* values. For the remaining missing elements, we tried to use another time classical imputation methods such are using mean or median of the other values in the column. But these procedures never led us to a drastic gain in performance. Skilled by the experience of the previous division, we decided to carry out a second split for each of the 4 datasets based on *NaN* remainings in `DER_mass_MMC`, producing a total of **eight subsets**, now with no missing values at all.

2) Correlation analysis

To better analyze each of the features we move further to plot the scatter matrix between all of them in all the subsets, to understand which of them have high correlation or are skewed. We discovered that at this point of the preprocessing pipeline some columns had the same values and a correlation of **1.00**. We hypothesize that these features have the same physical meaning when considered only in the subset. To simplify our model and improve the prediction we decided do delete one of the two features.

3) Skewness removal

The scatter plot matrix visualization pointed out the presence of many right-skewness distributions features. We applied **cube root transformation** to these columns to transform them into centered distributions.

4) Adding 1s column and standardization

The last steps of the main preprocessing phase are data standardization and adding an all-ones column. The former is performed to achieve better computation performance. The ones column precedes the feature matrix to be used as constant for the weights vector.

5) Polynomial features

We proceeded to polynomial expansion of each subset by taking powers and stacking them together. A note on our polynomial implementation is that we perform the raising only from degree 1, as we already added an all ones column. The exact degree chosen via cross-validation for each subset is presented in II-C.

C. Model selection and processing

With our processed data, we can move to the second improvement task we have to perform: tuning the hyper-parameters of our best models. We chose to dedicate our efforts only on Ridge Regression and Regularized Logistic Regression, as we saw at the very beginning of the project (TABLE I), that the former outperforms the others, while the second should be the best for classification as for Machine Learning theory. We choose the Regularized version instead of the normal one to improve computational efficiency. For both of them, we started by choosing the polynomial degree of all the subsets before moving on to the hyper-parameters selections.

For the Regularized Logistic Regression, we noticed almost immediately that using polynomial did not increase our performance sufficiently, so we decided to use only the first-order degree. For the Ridge Regression our first tests pointed out an opposite situation, with a decreasing test loss by increasing the polynomial to a certain degree.

After our previous tests, we performed **12-fold cross-validation** for the Ridge regression on our train set to choose the polynomial degree and the λ parameter. For the Regularized Logistic Regression we could not do the same k-fold due to time constraints of the cross-validation, so we tuned λ and γ with a simpler **8-fold cross-validation**. The hyper-parameters best choices for both methods are shown in TABLE II.

	Ridge Regression		Reg. Logistic Regression	
Data-set	degree	λ	λ	γ
Jet 0 without mass	7	$2.81 * 10^{-8}$	10^{-10}	$1.61 * 10^{-4}$
Jet 0 with mass	5	$1.76 * 10^{-8}$	10^{-10}	$6.72 * 10^{-6}$
Jet 1 without mass	5	$3.09 * 10^{-6}$	10^{-10}	$1.61 * 10^{-4}$
Jet 1 with mass	9	$2.95 * 10^{-7}$	10^{-10}	$6.72 * 10^{-6}$
Jet 2 without mass	6	10^{-10}	10^{-10}	$3.56 * 10^{-4}$
Jet 2 with mass	9	10^{-10}	10^{-10}	$6.72 * 10^{-6}$
Jet 3 without mass	5	$1.68 * 10^{-9}$	10^{-10}	$7.88 * 10^{-4}$
Jet 3 with mass	8	$1.04 * 10^{-9}$	10^{-10}	$1.49 * 10^{-5}$

TABLE II
HYPER-PARAMETERS SELECTION RESULTS

III. RESULTS

The final results we achieved are computed with Ridge Regression and Regularized Logistic Regression using the splitted dataset with no *NaN*, where the hyper-parameters and degrees are the ones obtained via cross-validation. For the Regularized Logistic, we used 100.000 iterations. The accuracy is computed on AIcrowd.

	Accuracy	F1-score
Ridge Regression	**0.824**	0.733
Reg. Logistic Regression	0.756	0.606

TABLE III
FINAL RESULTS FOR RIDGE AND REGULARIZED LOGISTIC REGRESSION

From the theory we expected the results of Regularized Logistic Regression to be better than the one of the Ridge Regression, but maybe this was not the case for this particular problem or we did not manage to find the right hyper-parameters which led to this condition. We could also hypothesize that the using of different polynomials basis for the different subsets led a very good fit for the Ridge Regression. This could improve significantly his performance and balance the troubles in using a linear regression for a classification.

Another aspect we can easily observe is that the overall accuracy increased considerably, signal that using a good data preprocessing is a fundamental step towards the implementation of a good Machine Learning system.

IV. SUMMARY

Overall, we can conclude we are sufficiently satisfied with the results we achieved with our implementation, in particular if we consider that we just used basic machine learning tools. Anyway improving the accuracy is possible, but it would require extra efforts both in data analysis and hyper-parameters tuning.

We learnt how preprocessing is crucial when dealing with Machine Learning implementation. We could have increased the goodness of the result by using optimal methods to clean our dataset or by adding important features, as example by performing cross-terms expansions. We also think that the presence of an expert in the topics we were dealing with would be helpful too. In this way we could have performed a better selection of the features based on deeper and insightful knowledge. It is also true that the implementation of more advanced methods such as Neural Networks or Random Forests would have helped us in reaching higher values of accuracy.

REFERENCES

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