CS433-Machine Learning Project 1

Gerald Sula - Ridha Chahed - Walid Ben Naceur
Department of Computer Science, EPFL, Switzerland

***Abstract*—Machine learning has become very important and offers tools and techniques to deal with a lot of problems in many scientific fields. In this paper, we explore and compare different supervised learning algorithms and how they deal with a data-set from CERN in the field of physics to predict the presence of the Higgs Boson.**

I. Introduction

The Higgs Boson is an elementary particle which has been an important subject of research in the physics community. Collision experiments, like the Atlas experiment at CERN, have produced large amounts of data which can be used to identify signals emitted by the Higgs Boson. In this project, we are confronted with a binary classification problem. Our aim is to predict CERN's particle collision events as either signals coming from the Higgs Boson, or simply as background noise. We have used methods from exploratory data analysis, feature processing, hyper-parameter estimation through cross validation, visualization and implementation of six basic machine learning algorithms to obtain the results that we will describe in the following sections of this paper.

II. Models and Methods

A. *Implementing the mandatory machine learning methods*

We first started by simply implementing six methods presented in the lectures, and proceeded to a first run of each to get a first glance of how the algorithms performed on the raw dataset.

Figure 1 : Raw results on data-set

	Parameters Used				
Methods	λ	γ	**Degree**	**Max_Iter**	**Accuracy (%)**
Gradient Descent	NA	0.1	1	500	74.43
Stochastic GD	NA	0.1	1	500	74.43
Least Squares	NA	NA	1	NA	74.50
Ridge Regression	0.0001	NA	7	NA	80.80
Logistic Regression	NA	0.5	1	1000	72.79
Regularized Logistic Regression	0.0001	0.5	1	1000	72.75

B. *Exploratory data analysis*

We decided to get a better sense of the data in order to develop a better model for this classification task.

- The data-set contains 250.000 points with 30 features. The features are divided into 2 categories : *raw quantities* measured by the experiment's detectors, and *derived variables* computed by CERN's physicists by using the raw quantities.
- 11 out of the 30 features presented many missing values or that those values could not be computed. Those values are represented by -999.0.
- All of those features are floating point numbers except for the number of jets, labeled *PRI_jet _num*, which can take 4 integer values in {0,1,2,3}.
- *PRI_jet _num* represents the number of jet pseudo-particles appearing in the detector. We noticed that distribution of missing values highly depends on the jet feature. We decided to partition our dataset based on the jet _num value in order to avoid the Simpson's Paradox phenomenon. At first, we considered dividing our data-set into 4, (one data-set per jet_num value), however the data-sets when jet_num were 2 and 3 were significantly smaller, and overall the performances were worse than when we partitioned into 3 data-sets. Hence, **tX0** corresponds to all data points where jet_num = 0; **tX1** corresponds to all data points where jet_num = 1; and **tX2** corresponds to all data points where jet_num = 2 or 3.
- We then plotted, for each data-set, the distribution of missing values per feature, and the distribution of labels shown in figures [2] and [3].
- Following figure [2], we dropped all features that were absent in the corresponding tX data-set, and for feature 1, we replaced each missing value by the median among the values that were present in each data-set.
- Following feature 3, we tried to use a balanced data-set for training each of the tX models, since the training data is skewed. However, leaving the data as is for training yielded better performance, hence we did not toss away data.

Figure 2 : Distribution of missing values per feature

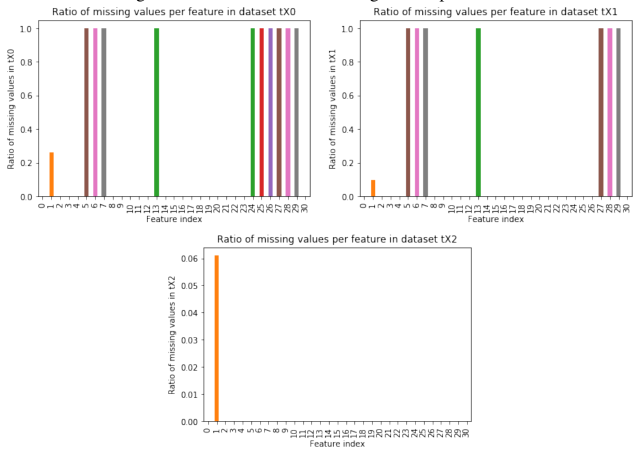
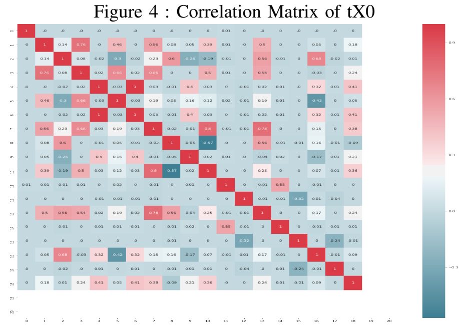


Figure 3 : Distribution of labels per dataset

	tX0	**tX1**	**tX2**	**Initial Data-set**
Label +1	74421	49834	40078	164333
Label -1	25492	27710	32465	85667

- For each data-set, after removal of the missing features, we have computed the correlation matrix of the remaining features, and tried to apply dimensionnality reduction. We removed the features that were highly correlated with all others (where the absolute value of the correlation coefficient was above 0.8).

Figure 4 : Correlation Matrix of tX0



C. *Feature engineering*

The task of predicting the presence of a Higgs Boson is too complex to be captured by a linear model only based on our current features. Hence, we introduced non linearity by expanding the feature space in order to fit more complex hypotheses.We added **pair terms**, i.e $x_{i,j} = x_i \times x_j$. As Taylor Series expansion can accurately approximate many of the usual functions (sine,cosine,exponential,log...), we only introduced polynomial terms of each feature x_i^k, for k between 1 and 15. The maximum degree of 15 was determined by a 10-fold cross-validation.

D. *Applying transformations*

For optimisations purposes (speeding up training), we applied several transformations to the data, which improved the overall accuracy and F1 score compared to when we did not apply those transformations.

1) **Rescaling the features** For each of the features, we computed its minimum and maximum value, and applied the following function to each point :

$$x_{i,j} \rightarrow \frac{x_{i,j} - min(x_i)}{max(x_i) - min(x_i)}$$

2) **Applying the log transform** As some distributions of remaining features had a large skewness, and that model training behaves better when the data follows a normal distribution, we applied the each point the following transformation to eliminate the skewness :

$$x_{i,j} \rightarrow log(1 + x_{i,j})$$

(Note that since we have rescaled the features, the log is well defined)

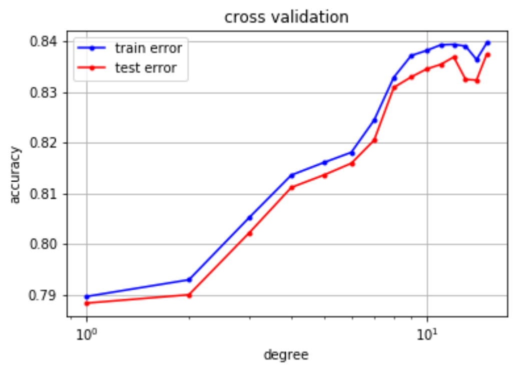
3) **Standardizing the data** Finally, we made our data follow a (0;1) normal distribution by applying the transformation :

$$x_{i,j} \rightarrow \frac{x_{i,j} - \mu(x_i)}{\sigma(x_i}$$

E. *Cross-Validation*

We used a 10-fold cross validation in order to determine the best λ,degree for our polynomial expansion, and learning rate γ. We determined that the best degree was 15, and computed the best lambda when doing ridge regression. We show our cross-validation plot with our training error and test error. Note that instead of MSE, the accuracy cost function is used. We would have expected each of our three models to have different best degrees by the cross-validation, however, the same degrees of 15 were selected.

Figure 5 : Cross Validation plot for tX0



F. *Testing and Avoiding Data Leakage*

Efforts have been made to avoid data leakage during cross validation and testing. Data leakage refers to the accident share of information between the train set and the validation or the test set. Indeed, for the data's prepocessing we apply several transformations like normalization. To do so we compute several variables per features. If this process is done on all the data before the train/test/validation split, we're introducing future information into the training predictor variables. Therefore, in the case of the normalization for example, feature normalization is done over the training data and the mean and variance are saved. Then we apply feature normalization to the predictor variables of the test and validation data sets using the training mean and variances.

III. Results

The following results obtained are contained in figure 6 after preprocessing and training.

Figure 6 : Distribution of missing values per feature

	Least Squares	Ridge Regression	Regularized Logistic Regression
Accuracy	83.3 %	81.9	70.4 %
F1-Score	74.8 %	72.2	61.7%

We have had the best scores using least squares, quite surprisingly, even though the performance of ridge regression was not that far away (but training was much longer for ridge and logistic regression). We would have expected regularized logistic regression to perform way better, being that we are dealing with a binary classification task. As we are dealing with a classification task, using the mean square error cost function does not make much sense. We might penalize well classified points, which is not meaningful. We used an other cost metric : **-accuracy**, defined by $-1_{y_i = predictLabel(x_i, w)}$. This might be the main reason as to why least squares and ridge regression perform way better, even though they are originally designed for regression tasks.

IV. Summary and Discussion

The prediction results of the algorithms were surprising, considering that we were dealing with a binary classification task. We had expected logistic regression (or the regularized version of it) to clearly outperform all the other algorithms, but our best results came from least squares and ridge regression. Some ideas could be explored in order to expand the project to get better scores such as :

- Implementing a **neural network** in order to train the model
- Expanding the feature space by adding even more feature (instead of pairs $x_{i,j}$, consider adding triplets $x_{i,j,k} = x_i \times x_j \times x_k$ and higher order cross terms as well. As this would add many features, one could consider **dimensionnality reduction** techniques, such as principal component analysis
- Instead of replacing some of the missing values by the median, consider training a model to predict the missing values, and retrain the model afterwards.

Although those tasks were more advanced than what we have currently seen in the lectures,they would not be impossible to implement using only numpy, and we believe that they could likely make us go beyond the 0.90 accuracy score.

References

[1] P. Onyisi, Higgs boson FAQ. University of Texas ATLAS group, 2012

[2] "CS433 epfl, machine learning course," https://mlo.epfl.ch/page-146520-en-html/, accessed: 2019-10-21.