EPFL Machine Learning CS-433 - Project 1 report

Edoardo Tarek Hölzl and Hugo Vincent Eliot Bertran Roussel
Department of Computer Science, EFPL Lausanne, Switzerland

***Abstract*—The following presents our work for the machine learning class working on the CERN challenge dataset. We describe the steps we followed to pre-process the data and to train a linear model to get a relatively good accuracy of 82% at infering the presence of the Higgs boson in function of the experiment settings.**

I. Introduction

The Higgs Boson is a subatomic particle whose existence was first theorized in 1964. In the standard model of particles physics it's role is of giving mass to the others elementary particles. It has the particularity to be extremely difficult to observe, being visible only indirectly by the decay of its processes. In 2012 the existence of the Higgs boson was confirmed by the ATLAS and CMS experiments following more than fifty years of research, including the construction of the Large Hadron Collider at the CERN. This discovery was awarded by a physic Nobel prize in 2013.
The CERN laboratory has since then published the Higgs boson machine learning challenge which is the subject of the first project of our machine learning class. We are tasked to predict the presence of the particle using the data processed at the ATLAS detector representing the type, the energy, the mass and the 3D direction of the decay processes at "event" time : when two protons are collided. We will use machine learning techniques to try to predict signal or background given the signature decay.

II. Methods and models

A. Data analysis

Our training dataset is composed of 250.000 events decay signatures with a label s for signal and b for background. Each signature is composed of 30 features given in floating point numbers except the `PRI_jet_num` column which is an integer giving the count of the pseudo-particle associated. We present here some ideas we used to better understand and exploit the dataset. First plotting the features distribution:

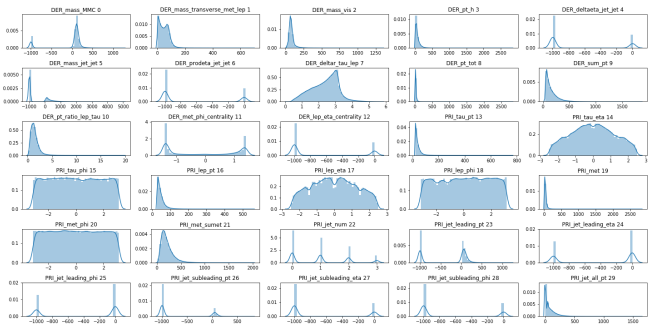

Fig. 1. Features distribution

From this plot we can notice three things: the presence of undefined values at -999 for 11 features, a categorical feature (`PRI_jet_num`) and 3 features that seem to have a uniform distribution (`PHI` features). There are also 2 other `PHI` features, but they are affected by the undefined values. If we plot the distribution of all 5 `PHI` features after removal of undefined values, we see that all 5 of them have a uniform distribution. For this reason, we have decided to drop them, as to reduce the feature space without a considerable loss of information. The resulting number of features is then 25.

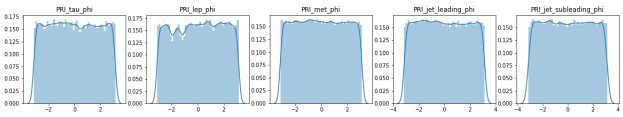

Fig. 2. PHI Features distribution

B. Feature engineering

1) Dataset splitting: After reading the documentation of this dataset, we understood that `PRI_jet_num` is a categorical column, and all features that are affected by undefined values (except for `DER_MASS_MMC`) depend of the number of jets: if `PRI_jet_num == 0`, then all 10 features are undefined, and so on. Hence, we decided to split the data into 3 subsets, depending on the jet number and train a model on each: for `PRI_jet_num == 0`, `PRI_jet_num == 1` and `PRI_jet_num > 1`.

2) Missing data: Now that we have split the data into 3 categories, the only feature with undefined values is `DER_MASS_MMC`. We replace all those by `NaN`, so that they are not taken into consideration when normalizing/standardizing the columns. Also, all feature expansion done on `NaN` data will yield `NaN`. Before training, we replace all those values with 0, and add a flag column, marking all entries that did not initially have a mass.

3) Features expansion: In order to handle non-linear dependencies, we applied a few non-linear functions to the initial data. The following transformations were used: polynomial expansion (up to a degree d), $log(1/(1+X))$ and $log(1+X)$ for positive columns, $\sqrt{|X|}$, $sin(x)$ and $cos(X)$. The effect of each of those expansions has been studied on the regularized logistic regression and are portrait in Table 1.

	none	cross mul	log	sin+cos	sqrt	log+sqrt	log+sqrt+sin+cos
accu	76.45%	78.54%	81.27%	79.058%	80.48%	81.688%	81.86%

TABLE I: Accuracy given feature expansion

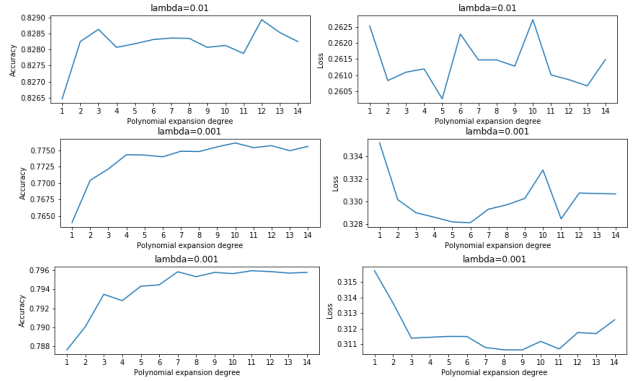

Fig. 3. Polynomial expansion accuracy and loss for each part of the dataset, in function of the degree using Ridge

4) Treating correlated features: After generating a correlation heat map, we saw that a lot of variable were correlated. We tried performing Principle component analysis and projecting the initial features onto an orthogonal subspace generated by selected eigen vectors. Unfortunately, this did not seem to help our accuracy nor our loss. However, we noticed that `DER_sum_pt` is highly correlated with `PRI_met_sumet` and `PRI_jet_all_pt` (correlation coefficient > 0.9), and decided to drop it. This left us with 24 features in total.

C. Models and training

1) Available models and choice: External libraries being forbidden for this project, we decided to use one of the different models seen in class :

- least squares regression (using normal equations)
- Ridge regression
- least squares regression using gradient descent or SGD
- logistic regression with gradient descent
- regularized logistic regression using gradient descent

Since we used feature expansion and ended up with up to 200 features, not using any regularization can be dangerous when fitting a linear model. Indeed it will tend to over fit on the training data. We believe regularized logistic regression and Ridge regression are the most appropriate models to use for this problem, and proceeded to use and compare both of those models.

2) Over fitting: Over fitting can come in different flavors especially when expanding the features with a polynomial basis of high degree. To avoid this we used k-fold validation using k=4 as well as using L2 regularization in both Logistic and Ridge regression.

3) Hyper Parameter tuning: Our model has 2 hyper parameters, namely the polynomial degree and regularization parameter. To find the optimal values we used k-fold cross validation using Ridge regression for degrees 1 up to 15, and also used multiple regularization parameters as to compare the combined effect of both parameters. The obtained results are described in Fig. 3. The same thing should have been done for Logistic regression, but run-time was significant; we thus decided to use the same polynomial degree as in Ridge, and a similar regularization parameter. This assumption could very well be wrong.

4) Training: As mentioned above, the dataset is split into 3 categories. Before training, all columns that are all `NaN` are dropped, as well as columns with 0 variance (the `PRI_jet_num` is also dropped for the 3rd category). This leaves us with 15, 17 and 23 features respectively. We then expand those features and end up with around 200 features per category.

5) Standardization/Normalization: Before training, we applied standardization to all columns (scaling to mean 0 and variance of 1), and normalization $(X - X_{min})/(X_{max} - X_{min})$ in the case of logistic regression, as if values are too large, we risk having an overflow. After all of the pre-processing is done, the bias column is added, as well as the flag column for undefined mass.

III. Results

After all the feature engineering, it is time to compare the accuracy of the two chosen models.

A. Ridge Regression

We have chosen polynomial expansion degrees of [5, 5, 9] and λ = $[10^{-2}, 10^{-3}, 10^{-3}]$ respectively for each category. The choice of these values was done through k-fold cross validation as previously mentioned, and is described in Fig. 3. The obtained accuracy on the test set was 80.3%. Although we did not rank very high, our model has low degrees of polynomial expansion, which reduces over-fitting.

B. Regularized Logistic Regression

For logistic regression, run times prevented us form cross-validating and finding the appropriate values. We thus used a polynomial expansion of degree 1, and $\lambda = 10^{-2}$. For these values, we have obtained an accuracy of 82%. There is an improvement compared to Ridge regression, and this is due to the fact that logistic regression is more appropriate for binary classification. A better score should be attained if proper cross-validation is performed.

IV. Conclusion

This project made us realise that one of the most important step in the data scientist job is to understand and pre-process well the dataset before any kind of learning. After that it is crucial to understand the goal of choosing a certain model compared to another and the issues that can arise with them (e.g overfitting).

This project was a good first example of what a real world machine learning workflow looks like. From dataset inspection, to feature selection and engineering to finally, model selection and training. The reminder on theoretical physics was also appreciated.