Machine Learning on Higgs Boson Dataset

Paul Feng, Thibaud Perret, Raphal Madillo
Machine Learning - Project 1, EPFL, Switzerland

***Abstract*—This paper is about making machine learning techniques work on physics field dataset. The report go through differents prediction algorithms comparing their performance, running time, robustness and accuracy.**

I. Introduction

The 4th july 2012, CERN announced they discovered the existence of a new elementary particle : Higgs Boson. The existence of this elementary particle introduced in 1964 have been searched actively during approximetly 50 years, to complete the Standard Model of physics.
To predict the existence of this elementary particle, a lot of collision of particules have been run in CERN particle accelerators producing smaller and instable particles like Higgs boson. They emit a singular signal that can be caught and saved with high accuracy sensors.
Our goal is to detect the Higgs Boson, and recognize its signature to give an accurate prediction if it have been created temporary given the data set of a collision. To achive this, a linear regression or classification algorithm is trained on a CERN particle accelerator (where prediction is given) dataset that confirm Higgs Boson signature or not during a colision.

II. Models and Methods

A. Prediction algorithms implemented

To try to solve this problem, we implemented, tested and improved 3 algorithm :

Least squares
Least squares is one of the best linear regression tool to resolve the problem given cause it is really efficient, easy to implement and there is no parameter to set. So it's the perfect algorithm to dive into the problem and set a first good prediction. Nevertheless it can gives bad results cause it easly overfits when we build polynomial from our input. The lack of parameters results in a lack of control of the quality of the prediction.

Logistic regression
Logistic regression is a classification algorithm so it seems more appropriate to solve our problem at first look. However it's quite slow to run because it compute at each step a gradient and a loss. It also have the disadvantage to be very fragile, because if you have high values (by putting some input data at high degree) the exponential of the logistic function rapidly gives overflow.

Ridge regression
Last but not least ridge regression.This algorithm is a variant of least squares but it has the benfit to penalize overfit with a lambda parameter, so it's more robust to greater degree of polynomial basis vectors. It also keep the advantages of least squares such as fast execution. The downside of ridge regression is actualy that a lambda have to be choosed to gives us a good prediction and it isn't a simple task, in addition at first look this algorithm isn't really done for classification.

Let's prove with examples some points we admited previously.
First let's prove that least square may overfit data. To prove this we build an augmented feature vector by adding polynomial basis up to degree 20. We split our dataset in two so we are able to train on the first part and test on second part (for this example we took a split the dataset in 2). To set the lambda, we performed a grid search over lambdas values (we found as one of best lamda=0.008).

Table I
Comparation of prediction for regression algorithms

	Least squares	Ridge regression
Prediction ratio	0.56	0.64

So we pointed out that ridge regression is more accurate dealing with high degree for augmented feature vector (so more complex data vector).
Second we show that without data preprocessing (like -999 entries etc...) the most natural algorithm for binary prediction, logistic regression have the lead. We set the different parameters to max iteration for logistic regression equal 10 000, gamma for logistic regression equal 1e-6 and lambda for ridge regression is equal to 3e-15. For this example, we set the split ratio to 0.8, so 80% of the data goes for training and the other 20% for testing.

This was our first output and we thought it would be clearly worse to more focus on logistic regression.
But the more we progressed the more we have seen it's

Table II
COMPARATION OF PREDICTION WITHOUT DATA PREPROCESSING

	Least squares	Ridge regression	Logistic regression
Prediction ratio	0.703	0.706	0.797

inconvenients, cross validations were time consuming and barely relevant because of the overflow encountered. The choice of the degree and gamma were taken at the edge of the overflow, ensuring a compromise between speed of the computation and robustness. We finally turned back to ridge regression for more stability.
We didn't tried gradient descent cause it has same problem than logistic regression: it was too slow. In addition it has wasn't in first hand for classification so we kept it out of our research.

B. Preprocess of input data

We did some analysis on the data because we saw that going blindly would only get us so far. The 23rd column of the data is a column which only contains integer values ranging from 0 to 3. Reading on the document explaining how they got the values, we learnt that these values represented the number of jets that were created when two particles collide. This meant that our data could be categorized into 4 different data sets, one for each value of number of jets. Some more analysis revealed that the nan-values were clearly dependent on the category the data was in. We tried looking at the percentage of nan values per column per category and got these results:

0 jet	1 jets	2 jets	3 jets
0 : 26 %	0 : 9 %	0 : 5 %	0 : 6 %
4 : 100 %	4 : 100 %		
5 : 100 %	5 : 100 %		
6 : 100 %	6 : 100 %		
12 : 100 %	12 : 100 %		
23 : 100 %	26 : 100 %		
24 : 100 %	27 : 100 %		
25 : 100 %	28 : 100 %		
26 : 100 %			
27 : 100 %			
28 : 100 %			

Therefore, we could simply remove the column full of nans for jet 0 and 1, and treat the nan differently for the column 0 in all category. We tried replacing the nan values by the mean, the median and the most frequent values, but we saw that keeping the median was the most efficient technique. We also remarked that last column for jet 0 is always 0, so we removed it since we did not have use of that.

C. Final submissions

Once we knew how to treat the data, we could do some cross validation. We separated the data into training and testing data using a ratio that is similar to the ratio of the given data to the prediction data (0.3). We then build a polynomial matrix for the training data, and run Ridge regression. The goal was to find the optimal degree for the polynomial matrix and the optimal lambda for the Ridge regression. We do all this a high number of times with different training and testing data each time. The way we decide we would select the best models was by choosing the lowest RMSE for the testing data, or the best prediction we did for the testing part. The RMSE for different degree is not really usable since its explodes when we go into bigger degrees. This way, we learnt the optimal weights with the training data and used this to try and predict the testing data.

III. CONCLUSION

To conclude, we see that our predictions gives us a pretty good approximation of the real results with 0.811 of right predictions. Although our current place of 115th could be better, we feel like there was a good progression in our understanding of these Machine Learning techniques. The Ridge regression clearly was our best shot because we could pick the best parameters more quickly than with logistic regression. The understanding of the data is also a very important step towards better predictions, because a sophisticated algorithm wont work as good as it could if the data had been treated specifically.