Machine Learning Project 1: Prediction of particles' types from CERN data on the *Higgs Boson*

EMILE BOURBAN AND FLORIAN DELBERGHE

Compiled September 4, 2020

Testing theoretical models has always been a challenge. In physics for example, when the standard model changes, a flock of experiments must be conducted to check whether the theory holds up. In addition to the new hardware that must be built to run the experiments, they also require a good way to gather and analyze this data. For this last part the recent advances in computer science can help. We try to provide in this project a machine learning based tool that can accurately predict the nature of a particle and that wouldn't require simulation or theoretical knowledge.

1. INTRODUCTION

The CERN has, for the last few years conducted a series of collision experiments to try and detect the *Higgs boson*. Those experiments have created a large amount of data and trying to detect exactly the signature of the God particle has proven difficult. It has a need for large computer models of the collisions and to have simulations that the data can be compared to. This process is long and expensive, so the goal of this project was to try to build a strong and reliable way to predict the nature of captured particle. For this we used collision data from the CERN's experiments. They were divided into two data sets, one for the training of the model and the second one to test the predictions of our algorithm. In this project, we implement several machine learning algorithms with varying degrees of accuracy and complexity, to find the model that is the most accurate and can best predict whether or not the detected particle is boson.

2. METHODS

Our first step to predict the particles' identity from the data was to analyze our dataset using the histograms of the values in each column. This combined with the correlation matrix for every parameter allowed us to visualize the data and think about its processing before the development of the prediction algorithms. We noticed that some of the features seem to have wrong values, written as a -999.0, as well as a lot of 0.0 values in the last feature. We also noticed in the 23$^{\text{rd}}$ feature column that the values were categorical (integers: $\{0, 1, 2, 3\}$). To determine the quality of our implementations, we decided to use the accuracy of the prediction of the model (calculated during the cross validation) instead its loss because the expected result is a binary classification for which it is possible to have a large loss but still get a good accuracy and inversely. We then proceeded to implement different machine learning algorithms that we trained on the data set. Each time, cross-validation (5 sub-samples) was applied to the method in order to have a more robust result and good estimator of the prediction accuracy. We first tested our different algorithms with the raw data, and then tried different cleaning process that all increased our accuracy: we removed the wrong values (-999.0), removed the outliers that were three time outside of the variance and standardized the data in order to have more homogeneous values.

A. Linear Regression

The simplest way to fit the model is using a linear regression, this will find coefficient of proportionality between the features $(x_1, \ldots, x_n)$ and the predicted value y. The accuracy of this model was improved by the further addition of an offset term.

A.1. Gradient Descent

The first implemented solving algorithm uses the gradient descent method. This is an iterative way of finding the minimum of the cost function. Being iterative, it is not the fastest way of finding the minimum of the function. This can be improved a bit by using the stochastic gradient descent that will only compute the gradient on a randomly selected subset of the features. The issues with these algorithms are that they can be divergent depending on the γ, that they take time to compute and that the stopping condition needs to be arbitrarily determined which is not ideal. In our case, this method diverges for polynomial regression with degrees ≥ 1

A.2. Least Squares

A faster way of finding the minimum of the loss function is using the least squares method. Assuming that the loss function has a unique solution for a global minimum, we can solve this equation for $\mathbf{w}$:

$$(\mathbf{X}^\top \mathbf{X})\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

This gives us a good solution of the minimum of the cost function by solving a linear system this is not an iterative solution reducing the complexity of the optimization. This can be useful since later we will improve the prediction using polynomial regression and this increases the number of feature even further increasing the complexity and so the run time.

B. Polynomial Regression

Teh next step to improve our prediction was to use higher polynomial degrees while still using the least squares method. This is helpful as it can refine the prediction since the type of particle is probably dependent on more than a linear proportionality with the features. For this kind of regression, we build a matrix containing the initial features values, an offset and the features columns to the power n, for $2 \leq n \leq max\_deg$ We tested the models with degrees from 0 to 22 in order to find the maximum degree before which overfitting appears. We saw that the accuracy on the cross validation increases up to degree 9 (where it reaches 81%), confirming our hypothesis of non-linearity. It then plateaus until degree 18 and finally starts going down again (Fig.1). This decrease is due to the occurrence of overfitting for degrees higher than 18. We can detect this decrease of accuracy when there is an overfit thanks to the cross validation. Indeed, an overfit will cause each sub-sample to have very different predictive parameter weights and thus their average over all the cross validation sub-samples will be meaningless. For these reasons, we decided to use polynomial regression at degree 10 for all of our following predictions since a slightly higher degree yields diminishing returns.

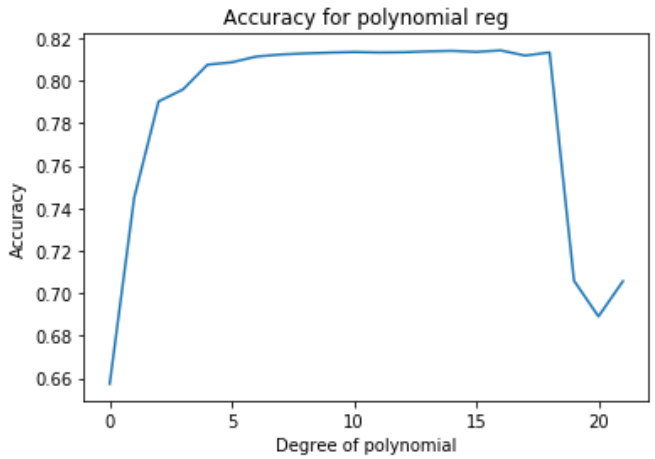

Fig. 1. Accuracy in function of the degree of the polynomial regression. We can see that the accuracy increases until degree 10, stagnates around 81% and decreases dramatically after degree 18.

C. Ridge regression

For higher degrees, fitting methods can sometimes overfit the model by converging to a minimum loss with a weight vector of large norm. The ridge regression prevents this kind of behavior by introducing a regularization parameter $\Omega(w) = \lambda \|w\|_2^2$ which is added to the cost function used previously and we will now be solving the equation for $\mathbf{w}$:

$$(\mathbf{X}^\top \mathbf{X})\mathbf{w} + \lambda \|\mathbf{w}\|_2^2 = \mathbf{X}^\top \mathbf{y}$$

This is done using the same equation solving algorithm as before but we can now have more robust results and we reach an accuracy over 81%

D. Catergory Separation

From our analysis of the different feature of the data, we had noticed that the 23rd feature was categorical. This seemed to correspond to the amount of error values in the row. In order to have a better prediction, we chose to separate the data into four parts according to this category. This separation gave matrices where some of the columns can have only erroneous values. Those will have the effect of adding an extra offset term to the data, or making the matrix non inversible for the cases -999.0 and 0.0 respectively. For those reasons, we removed the columns containing solely those values, thus preventing additional noise and errors during the solving of the systems. We then trained our algorithm separately for each category so as to have four different weight vectors. We measured our prediction accuracy with the test set by first testing the category of the data point and then applying the corresponding weights to make our prediction. Upon submission, we find an accuracy of over 82.5%

Figure 2 shows the accuracy depending on the lambda and the category of the data

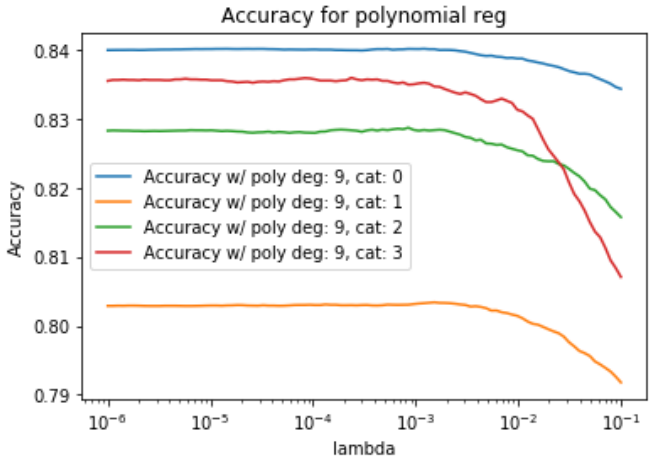

Fig. 2. This shows the accuracies obtained during the cross validation depending on the categories of the data, in function of the lambda for the ridge regression

E. Logistic Regression with Regularisation

Until now, we only treated the binary classification (boson/not boson) as a regression which can lead to errors, because our prediction can be much greater/smaller than the expected value of 1 or -1. A solution to this problem would have been to use the logistic regression and thus transform our prediction in a bounded interval. Furthermore, by adding a regularization term to the loss, we could have ensured that we will converge the correct values. However, we were not able to implement this method correctly and in an efficient way and thus we could not obtain meaningful results from it.

3. RESULTS

Our final method uses the ridge regression to predicts the outputs, this method is quite fast since it only needs to solve a system of equations instead of solving iteratively like a gradient descent would. It also has added robustness compared to the least squares since the weights are penalized by their norm preventing overfitting to some extent. For the degree that we used for the regression, we ended up with 10. As another improvement, we split the data into four parts (splitting by rows). Indeed, we saw that the 23rd column had categorical value depending it seemed on the error percentage of each measurements. This split

had us removing part of the columns to some of the categorized data since they only contained wrong values they would have brought unwanted noise during the optimization. Even though we removed columns for the final predictions staying at a degre of 10 did not worsen our results which could have been the case because of overfitting. For the predication, we did the same treatment to the data, then computed the predicted particle type for each category with a different weight. Finally, the data was reassembled and submitted

4. DISCUSSION

We obtained the best accuracy for our system using the ridge regression method with degree 9. However, in theory the logistic regression, in particular the regularized logistic regression should have improved the accuracy of the model. In our case, at degree 1 the logistic regression gives us a lower accuracy value compared to the simple least squares method with the same degree. For higher degrees our method diverges and predicts only 1 or -1 vectors.