Higgs Boson

Alexis Dewaele, Alexis Mermet, Robin Leurent

October 2018

Abstract

In this report we describe the progression of our implementation of machine learning algorithms, using the Higgs Boson dataset. We will quickly describe the algorithms, then explain the feature engineering and data processing we used and report our results.

1 Introduction

In March 2013, the existence of the Higgs Boson was confirmed by Peter Higgs and François Englert. This proof was a huge step forward in our knowledge of the universe. Investigations on this elementary particle began at the LHC, colliding protons together in an attempt to create and study the Higgs boson. Using measurements of decaying particles this model tries to predict whether or not the Higgs boson was created by the collision.

2 Models overview

2.1 Gradient Descent

This is the first model we implemented on our data. We didn't apply any feature engineering in this case except the standardization of the input data (2.5) and we used the Mean Square Error (MSE) loss function. With a gamma of 0.07 and a total of 4000 iterations, we achieved a score of 0.74426 (our best score with this algorithm). The SGD algorithm is using only one sample of data in each iteration. We implemented it using the Mean Absolute Error (MAE) cost function. The best result we obtained was a score of 0.74430 with a total of 2000 iterations and a gamma of 10^{-4}. We quickly decided not to use these algorithms since we know that Least Square gives the best result we can get for GD (2.2).

2.2 Least Squares

The Least Square algorithm tries to solve the normal equation

$$y_n = \mathbf{x_n}^T \mathbf{w} = \mathbf{x_n}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{1}$$

which gives the optimal weights for GD. We were still using no data processing except standardization and no new features. With this algorithm we obtained at best a score of 0.74759. From this point, we began to use some feature engineering (2.6) to increase the correctness of our predictions.

2.3 Ridge Regression

We spent most of our time working on the Ridge Regression algorithm. This model allows to avoid overfitting and underfitting by penalizing the L2-norm of the weights. This algorithm use the Least Squares model and adds to it the penalizing parameter. It tries to solve the equation for W:

$$\mathbf{X}^\top \mathbf{X}\mathbf{W} + \lambda \mathbf{W} = \mathbf{X}y \tag{2}$$

With lambda set to 10^{-5}, we obtained a score of 0.7552. Since it was better than all of our obtained scores, we optimized this function by expanding the features using the methods in Section 2.6 and by preprocessing the data (2.5). We achieved a final score of 0.83656.

2.4 Logistic Regression

In this section we are going to talk about the two last algorithm we have implemented. First of all, they are supposed to give us a better score since in this project we are working on a classification model and these algorithms aim to do so. We implemented the log-loss function to compute the loss for Logistic Regression and also a Sigmoid function mapping our values from $\mathbb{R}$ to the interval [0,1]. We need to find the W that minimizes the loss function:

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} ln(1 + \exp(\mathbf{x_n}^T \mathbf{w})) - y_n \mathbf{x_n}^T \mathbf{w} \tag{3}$$

We also realized that logistic regression needs to predict 0's and 1's. Since we have 1's and -1's, we processed the Y list to convert -1's in 0's [2]. For the Simple Logistic Regression, with 4000 iterations and a gamma of 10^{-6}, we obtained a score of 0.764 which is an improvement compared to all the previous scores without processing. In order to apply data processing (2.5) and feature engineering (2.6) without overfitting we needed to regularized this algorithm. Such as for the Ridge Regression, we penalize the L2-norm of the weights. By regularizing, we obtained a score of 0.83806 with 1000 iterations, a gamma of 10^{-6} and a lambda of 10^{-4},which is as expected better than ridge regression.

2.5 Data processing

We used different approaches to clean our data (some of the ideas written here were not used in the end to obtain our best score):

1. *Features not in the final code:* We first noticed that our input data has some values that are measurement errors according to the subject, the **-999**. To solve this issue, we decided to compute for each column the mean of said column without taking into account the -999 values. Finally we replace the -999 values with this computed mean. With this approach using the Ridge Regression we achieved to predict 1.5% more values. We also tried to discard columns uncorrelated to the Y vector [1], but it would always reduce our predictions.

2. *Separating data:* We found then that all the lines in the train set can be separated in 6 subgroups according to the index of their -999 values (Figure 1). We then computed one model for each subgroup. We separated the test lines in the same subgroups, and predicted each value with the corresponding model. We obtained 0.6% better predictions than when replacing measurements errors.

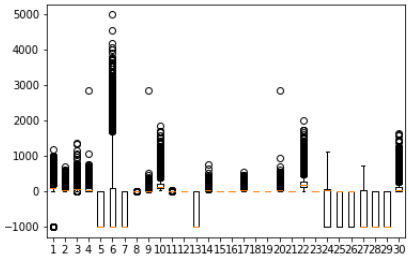

Figure 1: Distribution of the values on each column

3. *Standardization:* We also standardized each column in our data set by using subtracting the mean and dividing the column by the standard deviation.

2.6 Feature engineering

1. *Add polynomial basis:*

$$y_n = w_0 + \sum_{i=1}^{m} \sum_{j=1}^{d} w_{ij} * x_{n_i}^{\ j} \qquad (4)$$

To add features to our data set, we first thought about adding a polynomial basis of degree 4 to our input. We tried several degree values but as we can see on Fig.2, the value 4 minimized the RMSE on the test set. The polynomial extension allows us to increase the representational power of our model.

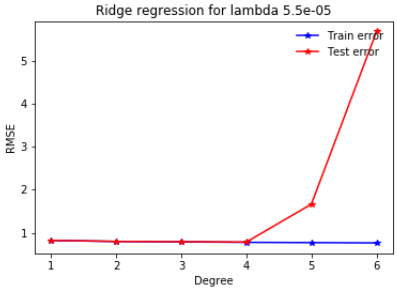

Figure 2: Evolution of the RMSE according to the degree of the polynomial expansion

2. *Multiply and add columns with each others:* We create new features by multiplying and adding correlated features together in order to get more data to train.

3. *Use functions to obtain new data points:* We extended our data set by applying several functions to it and concatenating the result to the matrix. We apply separately, the square root, cubic root, logarithm, reciprocal, cosinus, sinus and exponential functions [3]. These functions allowed us to gather more information about our data set by changing its distribution shape and as such, allowed us to make better predictions.

3 Results

The best score we got was 0.83806. To achieve this result, we used the algorithm Regularized Logistic Regression. We divided the data in 6 subgroups according to the -999 values and replaced the -1 values in the y array by 0. We expanded the features on each subsets, and standardized each subset independently. We separated the data into a test and train set and verified with different seeds that we were not overfitting by computing a model on the train set and predict the test set with this model. We then trained one model for each subset. We applied the exact same method on the test set, with the same means and standard deviations than for the train set, and predicted the y values. Compared to the raw ridge regression, with which we obtained 0.7452, this is a substantial improvement.

4 Conclusion

Using Regularized Logistic Regression, data processing and feature engineering we achieved to create a model that can predict if a given collision has created the Higgs Boson with an accuracy of 83.806%.

References

[1] Discover feature engineering, how to engineer features and how to get good at it. https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/.

[2] Discussion over logistic regression. https://stackoverflow.com/questions/36229340/divide-by-zero-encountered-in-log-when-not-dividing-by-zero/36229376.

[3] Feature engineering discussion. https://datascience.stackexchange.com/questions/21650/feature-transformation-on-input-data.