Machine Learning Course CS-433
Project 1: Higgs Boson?

Emilie MARCOU, Lucille NIEDERHAUSER, Anabel SALAZAR DORADO

***Abstract*—The aim of this project was to predict the presence or not of a Higgs boson using a variety of predictors. To do so, we used different regression methods. We were able to achieve an accuracy of prediction of 82.4% using a polynomial regression model with ridge regularization and adequate pre-processing.**

I. Introduction

Higg's boson is an elementary particle obtained from the collision of protons at high speed. The collision rarely produces a boson and when it does, the particle rapidly decays into other particles. Therefore, scientists decided to measure the products resulting from its degradation rather than the boson directly. The aim of this project was thus to classify the particles into "signal", i.e., boson or "background", i.e., not boson using the "decay signatures". To tackle this classification problem, we used a variety of machine learning methods.

II. Exploratory Data Analysis

The training set provided contains 250'000 samples and 30 predictors. One predictor is categorical (PRI_jet_num) and the others are continuous. We also saw that 5 features (15, 18, 20, 25 and 28) are angles. For each sample, we are given a label "s" (signal, 1) if a Higg's boson is present or "b" (background, -1) if it is not. We first had a look at the ratio between signal and background in our samples and saw that we had 34.2668% of boson. Our data set is thus slightly unbalanced. We then used box-plots and observed that the features have a very different range of values and many outliers. We did not find any features with particularly low variance, i.e. that contain almost always the same numbers. Finally, we used histograms with different colors for samples associated with "1" and "-1" labels to see the distribution of each feature and how it differs depending on the labels. We saw that features 15, 18 and 20 seem to have really similar distributions for both labels and are thus probably not crucial for the prediction.

III. Pre-processing

The given data set contains many -999 values to indicate that for this sample this feature was not computed or did not make sense. To make sure that these values were not taken into account by our model, we changed them to the median of the corresponding feature. We chose the median, instead of the mean, as we observed many outliers in our features and the median is more robust in this case. We also removed the outliers (values defined as mean + or - 3$\times$standard deviation) and replaced them with the median for the same reason described above. We standardized the data to make sure that all the features were in the same range of values and added a column of 1s to allow our model to have an offset.
To make better use of the categorical feature PRI_jet_num we used one hot encoding. To do so, we created 4 boolean features corresponding to PRI_jet_num = 0, 1, 2 or 3 respectively. Finally, we decided to transform the angles into their sine and cosine to have a better representation of angles. Just leaving the angles in radian could worsen the predictions since for instance -π and π could be interpreted as very different values by the model while they actually are the same angle.
After running our models with this initial pre-processing we thought of ways to improve them. We removed features 15, 18 and 20 since their distributions for 1 and -1 are very similar. We also set the outliers to the mean + or - 3$\times$standard deviation instead of to the median.

IV. Model Comparison

To compare the different models and select the best one, we computed the mean squared error (MSE) obtained by 10-fold cross-validation on our training set. We will detail the process in the subsequent sections and the tuned hyperparameters can be found in table I

A. Least Squares

This model finds the weights corresponding to the best line through the data that minimizes the sum of the squares of the residuals. This is a linear model that has no hyper-parameters to tune. For this model, we found a MSE of 0.3398.

B. Gradient Descent

Another way of finding the optimal weights, is by using the gradient descent (GD) algorithm. This is mostly used for models where resolving an equation to estimate the best weights for each feature is not trivial. We apply this approach in complementary to the least square, even if its equation is easily solved, to compare the two methods. After using 10-fold cross-validation to choose a step size for GD, we were able to optimize our model almost as well as with the least square model, namely, we reached an MSE of 0.3401 with a number of iteration set to 100 and initial weights set to 0.

C. Stochastic Gradient Descent

Since GD requires a lot of computational power for big data sets, it can be useful to use the stochastic gradient descent (SGD) algorithm instead. This algorithm computes the gradient of the loss at only one point randomly chosen at every step. This leads to a smaller computational cost, although it can take longer to reach the minimum of the function to

optimize. This was reflected in our results. Indeed the lowest MSE found with SGD was always significantly higher than the one found with the GD or least squares methods. We in fact obtained a MSE of 0.4219 after 200 iterations and with the initial weights set to 0. This can be explained by the fact that we would probably need a lot more iterations of SGD to find the optimal weights.

D. Polynomial Expansion

So far, all the models described are linear. To introduce flexibility, we constructed the polynomial basis function expansion of the features and then ran least squares regression on the new, expanded data set. To find out to which degree we should construct the polynomial expansion of each feature, we looked at each feature individually for the degree giving the smallest MSE. We then modified our data set by adding the polynomial expansion of each feature up to its optimal degree. With this method, we were able to construct a model giving us a smaller MSE than previous models i.e., 0.2725. However, when we submitted the predicted results *AIcrowd*, we found a lower accuracy than expected, namely 53.5 %, which was much lower than with other models. This probably means that we have over-fitted our training data set.

E. Ridge regularization

To solve the over-fitting problem, we added a regularization term modulated by the hyper-parameter λ. We performed a 10-fold cross-validation on the polynomial expansion described in the section above with different lambdas to find the one resulting in the smallest MSE. By doing this, we also obtained an MSE of 0.2725. However, this lead to a better accuracy on *AIcrowd* which supports our hypothesis that we were previously over-fitting the data.

We tried a second method, that combines the polynomial expansion of our features and ridge regression, to see if we were able to get better results. The aim was to find the best degree-lambda combination. To do so, we tried different polynomial expansions of all the features up to the same degree with different lambdas. This second method gave us similar results, i.e., a MSE of 0.2726.

Lastly, we ran this last method with the second pre-processing method we mentioned in section III. This further improved the model, resulting in a MSE of 0.2648, see Polynomial Expansion + Ridge (3) in table I

F. Logistic Regression

Since our problem is a binary classification, it seems logical to use logistic regression to predict the outcome as it is a classic for this type of problem. To implement it, we first had to modify the labels of our training set, since our output is either 1 or -1 and the logistic regression algorithm uses 0 or 1 labels, in most cases. With this function, we cannot find the optimal weights simply by solving an equation, thus, we decided to use the GD algorithm which is faster but more complex than SGD. We previously saw that it was a good way of finding the optimal weights. We tuned the gamma hyperparameter but we fixed both the maximum number of iterations to 200 and the initial weights to 0. This led to a MSE of 0.3706

G. Polynomial Regularized Logistic Regression

Finally, since the polynomial expansion of our features improved our results in the regression, we decided to try it on the logistic regression. To prevent over-fitting, we used the ridge regularization method. We obtained an MSE of 0.3668.

V. Results

Table I shows a summary of our models and the best test MSE with their corresponding hyper-parameters. Our best model is the third polynomial expansion combined with Ridge.

Algorithm	MSE	Hyperparameters
Least Square	0.3398	-
Gradient Descent (GD)	0.3401	$\gamma = 0.1$ max_iter = 100
Stochastic Gradient Descent (SGD)	0.4219	$\gamma = 0.001$ max_iter = 200
Polynomial Expansion	0.2725	array of best degrees*
Polynomial Expansion + Ridge (1)	0.2725	$\lambda = 10^{-8}$ array of best degrees*
Polynomial Expansion + Ridge (2)	0.2726	degree = 10 $\lambda = 10^{-4}$
Polynomial Expansion + Ridge (3)	0.2648	degree = 10 $\lambda = 10^{-7}$
Logistic Regression (LR) with GD	0.3706	max_iters = 200 $\gamma = 0.0072$
Polynomial LR with GD + ridge	0.3668	degree = 2 $\lambda = 1$

TABLE I: MSE for each model and their hyperparameters. *The offset and the dummy variables are not expanded. The degrees used for each feature (columns 1-35) are [10, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 9, 10, 1, 5, 4, 1, 10, 1, 9, 4, 10, 5, 5, 10, 1, 4, 1, 1, 4, 1, 1]

VI. Discussion

Even though we have a binary classification model, we can see that our best results are not obtained using logistic regression. This is quite surprising as generally, logistic regression gives best results in exactly this kind of problem. In the case of linear and polynomial regressions, we can use the least square method to find the exact minimum of our loss function, which is not possible with logistic regression models. Therefore, we had to use the GD algorithm which might not be able to minimize the loss function perfectly, due to our choice of step size and number of iterations, see IV-B.

VII. Conclusion

Throughout this project, we have built different regression models in order to predict the presence of the Higg's boson. Using a penalized polynomial expansion with adequate feature pre-processing, we were able to predict the presence of a boson on an unseen data set with an accuracy of 82.4%.