Class Project 1: Predicting the presence of Higgs boson from CERN data using binary classification

Romain Wiorowski, Taha Zakariya, Camille Arruat
Nov. 1st, 2021

***Abstract*—With an increasingly amount of available data to carry on complex analysis, automated processes are becoming essential. Through this project, we are aiming to predict the presence of Higgs bosons, based on the decay signature of a collision event. We found that ridge regression ($\lambda = 1e-4$) with a polynomial expansion of the input (degree = 11 for the derived features and 5 for the raw features) is the most adapted model to fit our data, with an accuracy of 0.832 obtained on the test set.**

I. Introduction

The Higgs boson is a particle whose existence is required to explain the standard model of physics. In order to detect it, physicists at CERN conducted particles collisions. However, the Higgs boson being rarely generated from these, many experiments were required and involved the creation of a huge amount of data. Moreover, when produced, the boson rapidly decay into different particles, making its detection difficult.
Through this project, we aimed to classify the decay products of the collisions experiments between signal, corresponding to the presence of a Higgs boson, and background noise. To handle the complex analysis of the data, we used a machine learning approach, avoiding an analytical computation of the model.

II. Method

Our approach to predict the nature of the particles is based on supervised learning. It consists in estimating the optimal parameters of a model (weights) using labelled data. A relation is derived between the data and the corresponding label (signal or background noise), by minimising a loss function, which represents the error of the model prediction compared to the real label.

A. Data pre-processing

We based our work on original data from CERN, consisting in 250'000 events[1]. Each event has 30 features and are labelled 1 (signal) or -1 (background noise). As the relation between an input event and the predicted label is derived from the training, the quality of the training data is crucial. The data pre-processing was determined by testing different configurations and choosing the one that gave us the best accuracy (cf section II-D). The accuracy obtained with ridge regression $\lambda = 1e-4$ after each data pre-processing step is presented in the table I.

1) Dealing with default values (-999): We first visualised the data, plotting the histogram of the values for each feature. We observed that some values were set to -999, corresponding to variables that could not be computed. We tried removing 7 features for which more than 50% of the events did not have a value, but noticed that this leads the performance of the model to decrease a lot. Assuming that the missing data could have some importance, we decided to keep track of them instead: at the end of the pre-processing, we add, for each of the initial 30 columns, a new column which values take 1 if the underlying feature has a default value, 0 otherwise. [1]
Then, we replaced the missing data by the median value of the corresponding feature, so that it does not bias our model.

2) Dealing with the outliers: Outliers are observations that are very different from most of the other observations and they can result in a large error, breaking the learning process. To decrease their influence without removing data points, we reduced the values that were greater (resp. lower) than $median + 5 * std$ (resp. $median - 5 * std$) of the corresponding feature, to $median \pm 2 * std$ depending on if the value was smaller or greater than the median.

3) Standardisation: The features associated with the datapoints being physical variables of different units, their values can be of different order of magnitude. Thus, it does not make sense to compare them in this way. For this reason, we standardised the data by subtracting the mean and dividing by the standard deviation. This procedure was done to each feature separately, except for the 23rd one (PRI_jet_num), which consists in categorical values $\{0,1,2,3\}$.

4) Polynomial expansion: Visualising the distribution of our data for each feature, we observed that the relation between the input and the output is more complex than linear, so using a linear regression to fit the data could lead to under-fitting. To overcome this, we created an input matrix of augmented features, adding a polynomial basis to the original data.
Comparing the performance of a model trained on the derived features only, prefixed DER (accuracy = 0.82), with the same model trained on the raw features, prefixed PRI (accuracy = 0.70), we observed that the derived features have more impact on the predictions. For this reason, we chose an expansion of degree 11 for the derived features and 5 for the raw features.

5) Categorical split: As the last step of the pre-processing, we split the datapoints into 4 groups according to their values of the 23rd feature, keeping track of their IDs for later submission, and computed one weight vector for each category.

Finally, we added the default value markers mentioned in II-A1.

[1] AIcrowd, 2021, ML Higgs

	Raw data	Polynomial expansion	Dealt with outliers + Standardised	Dealt with default values	Category split
Accuracy	0.744	0.806	0.808	0.824	0.832

Table I
ACCURACY OBTAINED WITH RIDGE REGRESSION $\lambda = 1e-4$ AFTER EACH DATA PRE-PROCESSING STEP.

B. Training

To optimise the models, we randomly separated the training data into a training set and a validation set. The optimal value for the hyperparameters of the models were estimated through a grid search, searching for the value giving the best performance on the validation set. For example, the figure 1 in VII) represents the accuracy of the regularised linear model depending on the value of λ (between 1e-10 and 1e-1). We can notice that the performance of the algorithm is the best for $\lambda = 1e-4$.

C. Prediction of the labels

Using regressions for binary classification, we transformed the output of the models into label predictions depending on a threshold. For linear (resp. logistic) models, we predicted the label to be 1 when $f(x) > 0$ (resp. > 0.5) and -1 otherwise.

D. Evaluation of the models

To estimate the generalised error, we used cross-validation over 10 subsets of the data. The performance of the model was evaluated using the accuracy metric. We also computed the f1-score because, as signal appears rarely compared to background, the accuracy could be biased by the unbalanced dataset.

III. MODELS

1) Linear regression: Applying a linear regression is the simplest way to model data, assuming a proportional relation between the inputs and the outputs ($y_{pred} = \mathbf{x}^T\mathbf{w}$). For this model, we used the Mean Square Error (MSE, eq. 4 in appendix) as the loss function.
We used 2 different approaches to optimise the weights of the regressions.
Gradient descent (GD): First, we used gradient descent to minimise the weights of a linear model. It is an iterative method which consists in moving the weight vector $\mathbf{w}$ at each step, depending on the direction of the gradient of the cost function ∇L (eq. 5 in appendix).
The stochastic gradient descent (SGD) is similar, but faster than gradient descent to optimise the weight. It consists in computing the loss function over a batch of datapoints instead of the whole training set.
Normal equations: Assuming that the cost function is strictly convex, the second method we used for linear models, is an analytical approach. We solved the following normal equation:

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

where $\mathbf{X}$ is the matrix of the inputs values, $\mathbf{y}$ is the vector of outputs.
Ridge regression: However, using supervised learning, it can happen that the model adapts too well to the data, fitting the noise. In particular, adding a polynomial basis with high degrees to the input, requires a more complex model (more parameters), which can lead to over-fitting. To prevent that, we used L_2-regularisation (ridge regression) to penalising weights vector of large norm. This introduces the hyperparameter λ and modifies the normal equations as follows:

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

2) Logistic models: In binary classification, linear regression can be biased by unbalanced classes and extreme values. To deal with that, we implemented the logistic regression which bounds the predictions values between 0 and 1 ($y_{pred} = \sigma(\mathbf{x}^T\mathbf{w})$). For this model, the error is given by the logistic loss (eq. 7 in section VII). We optimised the algorithm through gradient descent and added regularisation.

IV. RESULTS

Using a linear regression, we obtained an accuracy of 0.707 (solving the normal equations). We then reached an accuracy of 0.715 when implementing a logistic model (GD, iters=30, γ=1e-3). Finally, we found that the best approach to fit our data is to train a ridge regression (eq. 2) with λ=1e-4, and a polynomial expansion of the input (degree = 11 for the derived features and 5 for the raw features), increasing the accuracy to 0.832. This is the final model we used for predictions.
The prediction of the label for the 568'238 events of the test set, were obtained by pre-processing the data the same way we did for the training set, i.e. using the same parameters. We, then, applied our trained model. We reached an accuracy of 0.832 after submitting our results on AIcrowd (ID 164433).

V. DISCUSSION

We first observed that our final approach is faster than gradient descent. Indeed, it only implies one equation solving, while gradient descent is an iterative method with computation at each step. Then, we noticed that ridge regression is more robust than normal linear regression with respect to over-fitting, as we added regularisation to penalise large weights vectors. We also obtained a better performance of the algorithm when adding a polynomial basis to the input. Indeed, to use a linear regression, we need to show we identified that there are some linear behaviour in our data. Yet, from the visualisation of the data, the distribution of the values seemed more complex than a linear behaviour. However, when using polynomial expansion, the logistic regression did not perform better than the linear regression unlike what we expected for binary classification.

VI. SUMMARY

To classify decay product between signal and background noise, we trained models based on linear and logistic regressions, and optimised the parameters through gradient descent or by solving the normal equations. Our final method uses ridge regression combined with polynomial expansion, for which we obtained an accuracy of 0.832.

References

[1] A. Vidhya, "Dealing with missing values in python – a complete guide," 2021.

VII. Appendix

A. Standardisation

$$X_{standard} = \frac{X_{original} - mean}{std} \tag{3}$$

where $mean$ (resp. std) is the average value (resp. standard deviation) of the original data.

B. Hyperparameters estimation

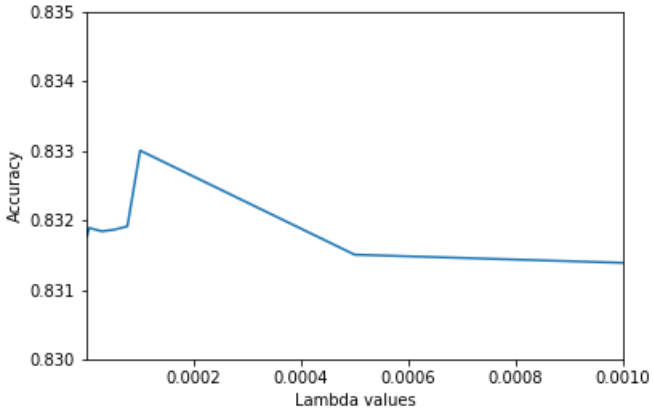

Figure 1. Optimisation of the hyperparameter λ for ridge regression.

C. Mean Square Error (MSE)

$$MSE(\mathbf{w}) := \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x_n}^T \mathbf{w})^2 \tag{4}$$

where y_n is the real output associated to the input $\mathbf{x_n}$, and $\mathbf{w}$ are the weights of the linear regression[2].

D. Gradient descent

$$\mathbf{w} = \mathbf{w} - \gamma \nabla L \tag{5}$$

where the hyperparameter γ is the step-size.

E. Sigmoid

$$\sigma(x) := \frac{1}{1 + e^{-x}} \tag{6}$$

F. Logistic loss

$$L_{logi} := \sum_{n=1}^{N} -y_n \log y_{pred} + (1 - y_n) \log (1 - y_{pred}) \tag{7}$$

where $y_{pred} = \sigma(\mathbf{x_n}^T \mathbf{w})$ is the predicted output.

[2]M. Jaggi, M. Emtiyaz Khan, 2016, Optimization