ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MACHINE LEARNING CS-433

IN SEARCH OF HIGGS BOSON

Gonçalo Gomes
Diogo Soares
Mohamed Saad Eddine El Moutaouakil
Team DSG

***Abstract*—Machine learning is the study of computer algorithms that can be improved automatically through experience and the use of data. This article will discuss using machine learning tools to distinguish signal events that produce Higgs Boson based on background events. A comprehensive exploratory data analysis constitutes our starting point, followed by preprocessing and cleaning the data. Finally, our model is described along with its performance.**

I. Introduction

The Higgs boson is a elementary particle in the standart model of particle physics first introduced in the 1964 PRL symmetry breaking papers by Peter Higgs and co. On December 2013, two of the models' authors received a Nobel Prize after the ATLAS experiment proved their theoretical predictions. The latter consisted of collecting particles resulting from generated protons collisions. Given the decay property of Higgs Boson, its presence can only be inferred from the collected results. Consequently, this paper suggests the application of Machine Learning techniques, specifically Binary Classification, on the ATLAS experiment data to predict the presence of Higgs Boson.

II. Conceptual Framework

A. Dataset

In our project, two datasets were used, one to train and build our models, containing a total of 30 different features and 250,000 data points, and other to try to predict the results based on the models build with the training data set. The prediction feature present in the given training set take value s or b, for the sake of notation, we will consider that $s = 1$ and $b = -1$.

B. Split Data Based On The Number Of Jets

Divide and conquer can be a useful technique in machine learning when one expects different distributions for well defined subsets of the same dataset. After careful analysis of the data we computed the probability of existing a Non defined value in one of the features for each *PRI_jet_num* number defined in $\{0,1,2,3\}$, it was clear that there was a high correlation between the jet number and the number of missing values, therefore we hypothesized that the distribution would be different after data cleaning, which ultimately led us to decide to split the data according to the *PRI_jet_num*.

C. Data Cleaning

Through a simple analysis of the data set, it is visible that some features have outliers, namely *DER_deltaeta_jet_jet*, *DER_mass_jet_jet* and others. This happens most likely due to an out-of-range measurement of decay signature sensors. In order for this anomaly to not affect the models, this non defined values are replaced by some plausible approximations (i.e the mean value of that feature, or since we speculate that this undefined values occur due to an out-of-bound measurement, one can consider replacing them by the maximum/minimum value of each feature).

In order to remove redundancy we can consider two interesting metrics, variance and correlation. Low variance features usually are not useful for models since the amount of information a feature possesses is related to it's variance, similarly high correlation between features isn't adding new information so one of the features can be dropped.

Consequently the data set is passed through a filtering process where features with lower variance than a threshold are dropped. In our particular case, the threshold was set to zero. The motivation for this choice is that it's impossible to get any type of information from features with zero variance, on another hand, since we are comfortable with the run-times, we aren't worried with the computational cost of having a high number of features, therefore we accept any feature with positive variance.

Additionally, we visualized how correlated were the features (figure 1). We removed those that are highly correlated (correlation coefficient exceeding a threshold set empirically to 0.96). It yielded a better runtime.

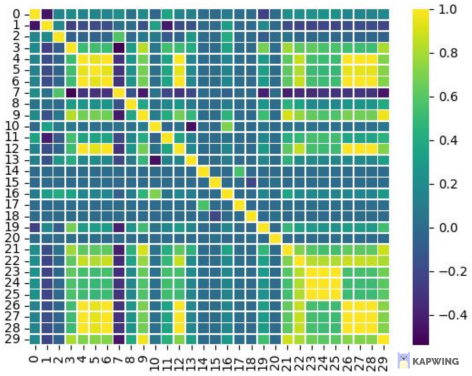

Fig. 1. Correlation matrix

In the end from 30 initial features, we removed 11, 7, 0 features for dataset 0, 1, 23 respectively.

D. Pre-processing Training Data

1) Matrix Standardization: Data standardization is the processed used to ensure that internal data is consistent and in the same format in order to make it easier to track and analyse. This process is useful especially when the data has varying scales and the algorithms used make assumptions about the data having a Gaussian distribution (i.e such as linear regression, logistic regression, and linear discriminant analysis) In this case, every feature was standardized for the purpose of having a zero-mean and unit-variance using the z-score normalization formula (1)

$$\tilde{X}_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j} \tag{1}$$

2) Polynomial Regression: The goal of regression analysis is to model the expected value of a dependent variable y in terms of the value of an independent variable x. In simple polynomial regression the model is given by a simple linear expression.

In many cases this linear approach may not be sufficient to fit the data since it could be generated by a non-linear process.

In this case we can model the expected value of y as an n^{th} degree polynomial, yielding the general polynomial regression model (2)

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_n x^n + \epsilon \qquad (2)$$

where ϵ is an unobserved random error with mean zero.

Besides that, it could be the case that the model which we are trying to predict, depends on a multiplication of features (i.e $y = \beta_{00} + \beta_{01} x_1 + \beta_{02} x_2 + \beta_{11} x_1^2 + \beta_{12} x_1 x_2 + \beta_{22} x_2^2$), in this case a model based simply on the polynomial regression stated in (2) wouldn't hold. Therefore, we opted for the strategy where the model is weighted with a multiplication of features. The problem with this strategy is that it increases the complexity of our model exponentially, so we decided to do the multiplication of features only for a low degree. Additionally for higher degrees, the model is weighted normally as a polynomial regression, since it's complexity is much smaller.

E. Classification Algorithms Implemented

Least squares:
implementation of least squares algorithm

Linear regression GD:
implementation of linear regression using gradient descent algorithm

Linear regression SGD:
implementation of linear regression using stochastic gradient descent algorithm

Ridge regression:
implementation of ridge regression using normal equations

Logistic regression:
implementation of logistic regression algorithm using gradient descent with logistic loss function.

Regularized logistic regression:
implementation of logistic regression algorithm with regularization

F. Tuning hyper-parameters

In order to choose the best set of hyper-parameters for each learning algorithm (there are 3 models for the dataset since we splitted it in 3 subgroups), we did a grid search to find the minimum loss possible for each model. A 5-fold cross validation was used, we opted for K-fold cross validation strategy since it ensures that every observation from the original dataset appears in the training and test set making it one of the best approaches.

III. Results

A. Baseline

For the sake of comparison with extended models, a baseline performance was calculated. We used least squares as the baseline model. The data pre-processing described in section II was used. No feature extension was used. The table shown below contains the results for this performance.

Classifier	Categorical Accuracy	F1 Score
Least Squares	0.689	0.659

B. Best Model

Classifier	Categorical Accuracy	F1 Score
Gradient Descent	0.740	0.619
Stochastic GD	0.734	0.621
Ridge Regression	0.817	0.719
Logistic Regression	0.789	0.672
Reg logistic Regression	0.810	0.709

According to the results shown in the table above we opted for ridge regression since it was the classifier that gave better results.

The best performance was achieved by using feature extension together with a simple ridge regression. The hyper-parameters were found using grid-search in the hyper-parameters space, the table below, shows the best hyper-parameters found.

Subset where Pri_Jet_number =	0	1	2 or 3
Optimal_gamma	1e-8	1e-10	1e-12
Max_multi_degre	1	2	2
Max_single_degree	2	4	9

For each subset, a ridge regression fit is performed and produces the best weights which are used to do predictions.

C. Discussion

The model with ridge regression improved the categorical accuracy and the F1 score by 18.6% and 9.1% respectively. The exact results can be seen in the tables above.

Other considerations that could improve the performance of the model:

- In theory regularized logistic regression, with a good set of hyper-parameters, could give us equally good or even better results but, given the high dimensionality of the problem (6 parameters vs 3 parameters with ridge regression), it would be computationally harder to find such parameters.
- Different transformations on the features, for example we could consider log transformations in order to try to compute physical phenomena defined with logarithms.
- More extensive hyperparameter search, due to the exponential cost of grid search and the time we had, we only tried a small subset of hyperparameters.

References

[1] Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyon, Balazs Kegl, and David Rousseau, Learning to discover: the Higgs boson machine learning challenge.
https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf

[2] Wikipedia Higgs boson
https://en.wikipedia.org/wiki/Higgs_boson