ML Higgs classification

Mirali Ahmadli, Antonino Emanuele Scurria, Bartlomiej Borzyszkowski
EPFL, Swiss Federal Institute of Technology Lausanne

***Abstract*—In this report, we analyze our results on the Higgs Boson challenge, obtained using various machine learning (ML) techniques. We apply numerous data processing methods and implement several basic models, including linear and logistic regression. Additionally, we propose custom modifications of the baseline methods, using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which further improves the performance. Moreover, we apply a k-fold cross validation over the provided training set, use a hyper parameter search and implement a polynomial feature expansion. Our best model based on the regularized logistic regression with BFGS achieves 81.8% accuracy in the AI-Crowd challenge. We compare the performance of our methods and discuss their implementation below.**

I. Introduction

The Higgs Boson is an elementary particle in the Standard Model of physics which explains why other particles have mass. Our goal in this project is to apply machine learning techniques to the particle accelerator data from CERN in order to recreate the process of "discovering" the Higgs particle. Specifically, we aim to solve a binary classification problem to determine whether an event in the provided dataset is a relevant signal (a Higgs Boson) or a background noise. Our findings show the importance of the data pre-processing stage and implementation of the polynomial feature expansion. Moreover, we indicate that the regression models (e.g. linear regression) achieve worse performance on the given task than logistic regression methods, which are more suitable for the classification purposes. Our code is available for reproducibility at: github.com/CS-433/ml-project-1-ml-mops

II. Data preparation

In this section, we make an overview of the dataset and detail our preprocessing and feature engineering techniques.

A. Exploratory Data Analysis and Pre-processing

- The dataset contains 250,000 labeled observations represented by 30 numerical features. Originally, about half of the features are primitive, whereas the other half is derived from these raw features. Each feature vector represents the decay signature of a collision event.
- The labels are defined as "signal" (Higgs Boson - 1) or "background" (noise - 0). Our goal is to solve a binary classification task, namely to predict the correct label of an event, given an input feature vector.
- "Undefined" missing values are denoted by -999.0. These values are meaningless, therefore they should be cleaned in the pre-processing stage so that our models are not affected by learning them.
- Our study of the literature [1] indicates that these "undefined" values are not random and in fact, systematically missing, based on the *PRI_jet_num* feature e.g. *PRI_jet_leading_eta* is undefined if *PRI_jet_num* = 0.
- Plotting the feature distributions, multiple positively skewed distributions were observed (Fig. 1).

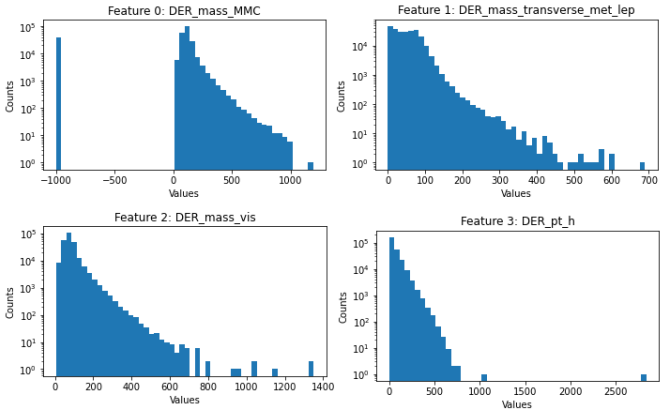

Fig. 1. Feature distribution of the first features in the dataset. The Y axis is in log scale. This plot is meant to showcase the tendency for our dataset to have positively-skewed distributions.

B. Feature engineering

- (1) Replacing "undefined" values: We first convert -999 values to NaNs to handle them easily and then impute them by mean of the corresponding feature.
- (2) One-hot encoding: After visualizing the features, we see that *PRI_jet_num* feature is in $[0, 3]$ range and this column can be encoded. We split this column into 3 parts: *PRI_jet_num* $= 0, 1, \geq 2$ and do one-hot encoding.
- (3) Removing outliers using modified IQR method: Instead of taking first and last quartile, we take first and last 5% (q_5, q_{95}, $iqr = q_{95} - q_5$). In normally distributed data, it would mean $q_{95}, q_5 = mean \pm 1.65std$ and we would keep values in range $mean \pm (1.65std + c * 3.3std)$. We choose $c = 1.5$ to remove only statistically very extreme values
- (4) Log transformation: To handle skewed distributions, we apply log transformation to positive continuous features, so that data is more interpretable.
- (5) Standardizing dataset: Finally, we standardize train and test dataset using mean and standard deviation of the train dataset.
- (6) Data Augmentation: As linear models can fundamentally only provide linear decision boundary, we have tested data augmentation with polynomial bases of different degrees for the various models to get non-linear

decision boundary and boost the performance. We define the polynomial degree as a hyperperameter that can be manipulated in our pipeline. The transformation describing the data augmentation is as follows:

$$x \to 1, x, ..., x^{degree} \tag{1}$$

III. Methodology

This section discusses our data analysis methodology, including implementation of the algorithms, their evaluation and hyperparameter optimization.

A. Cross Validation

Cross validation is a powerful method to assess the performance of the models and avoid overfitting the training set. In this project, we use a k-fold cross validation and set $k = 5$ by default. We randomly split the training data into k subsets. During training, we use $k - 1$ subsets to train the model, and leave the remaining one for evaluation. The training and evaluation operations are repeated in a loop until all available data have been been used for training and testing.

B. Methods

We implement six models presented during the lectures and required in the project description. We note that the task we are facing is a classification problem, therefore regression models (e.g. linear regression) are not suitable for this task without the feature augmentation. For the classification models, we apply a simple logistic regression as well as its regularized version. Moreover, as specified in the project requirements, we set the batch_size to 1 for the stochastic gradient descent.

1) Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm: Our contribution also includes extension of the logistic regression methods by BFGS, Gradient based Quasi-Newton optimization algorithm, that attempts to solve a general nonlinear optimization problem without any constraints [2–5]. We decided to use it as it is known to be good in finding local minima in less steps. It determines the descent direction by preconditioning the gradient with inverse of Hessian matrix.

2) Model Selection: To choose the best model and verify our preprocessing steps, we test ran models by increasing preprocessing complexity and reported the 5-fold cross-validated accuracy for comparison (Tab. I). It starts with raw data and we incrementally increase the complexity. To verify preprocessing steps, we only report our results for naive and BFGS + $\mathcal{L}^2$ Regularized Logistic Regression models.

C. Hyperparameters tuning

To choose the correct hyperparameters, we implement a grid search over a set of different possible values. We define 3 major hyperparameters: step size γ_{init}, weight decay λ, and a polynomial degree deg. Additionally, we consider γ_{decay} and γ_{fin} as the optimization algorithm should make smaller steps closer to the minimum. We train the models with all possible combinations of these parameters in order to obtain the best configuration for each algorithm. It makes our search robust as we noticed that manipulation of one hyperparameter often affect the other one, i.e. change of the polynomial degree requires to change also the learning rate.

IV. Experimental results

In this section, we present our results for preprocessing steps and the best results achieved by each of our models, together with an optimal set of hyperparameters used for the training (Tab. II). All reported experiments were performed on the provided training data using 5-fold Cross Validation.

TABLE I
Overview of preprocessing. Numerals indicate preprocessing steps in Feature engineering section.

Conditions	Log Reg	BFGS $\mathcal{L}^2$ Reg. Log Reg
I: Raw	$65,13\%$	$75,02$
II: I + (1) + (2) + (5)	$72,43\%$	$75,2$
III: II + (3)	$73,04\%$	$75,55$
IV: II + (4)	$73,81\%$	$76,59$
V: II + (3) + (4)	$74,2\%$	$76,83$
VI: V + (6), $degree = 2$	$78,52\%$	$81,13$
VII: V + (6), $degree = 3$	$81,0\%$	$81,8$
VIII: V + (6), $degree = 4$	$77,96\%$	$81,68$

TABLE II
Overview of the models and results.

Model	Accuracy	γ_{init}	λ	deg
Linear regression GD	$77,2\%$	0.01	-	2
Linear regression SGD	$71,6\%$	0.01	-	1
Least Squares	$79,7\%$	-	-	3
Ridge Regression	$79,9\%$	-	0.0001	4
Logistic Regression	$81,0\%$	0.01	-	3
$\mathcal{L}^2$ Reg. Logistic Regression	$81,1\%$	0.01	0.0001	3
BFGS Logistic Regression	$81,7\%$	1	-	3
BFGS $\mathcal{L}^2$ Reg. Logistic Regression	$81,8\%$	1	0.0001	3

A. Preprocessing Steps

From Table I, we can see that every step of preprocessing increased accuracy and proved to be useful. Moreover, we can see that when we use polynomial features with degree 4 performs worse than with degree 3 for both models.

B. Best model

The best performance is obtained through the BFGS $\mathcal{L}^2$ regularized logistic regression with an overall accuracy of 0.818 on AI-Crowd. The model used feature augmentation of degree 3 and a regularization factor equal to 0.0001. To obtain fast convergence of the model, we exploited the BFGS algorithm with $c = 0.05$ for Armijo rule constant and $\beta = 0.8$ learning rate decay constant.

V. Conclusion

In this project, we analyzed the Higgs Boson data from CERN and designed a machine learning pipeline to solve a binary classification problem applied to physics. Our experiments show that careful data pre-processing and feature engineering have significant influence on the results. Moreover, we achieved remarkable performance improvement thanks to applying polynomial features, which allow to capture the nonlinear relationship in data. Finally, hyperparameter search as well as cross-validation of our models allowed us to achieve competitive results in the challenge.

References

[1] Claire Adam-Bourdarios, Glen Cowan, Cécile Germain, Isabelle Guyon, Balàzs Kégl, David Rousseau, "Learning to discover: the Higgs boson machine learning challenge", CERN, 2014

[2] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms", J. Inst. Math. Appl. 6, 76–90 (1970) https://doi.org/10.1093/imamat/6.1.76

[3] R. Fletcher, "A new approach to variable metric algorithms", Comp. J. 13, 317–322 (1970) https://doi.org/10.1093/comjnl/13.3.317

[4] D. F. Goldfarb, "A family of variable-metric methods derived by variational means", Math. Comp. 24, 23–26 (1970) https://www.ams.org/journals/mcom/1970-24-109/S0025-5718-1970-0258249-6/

[5] D. Shanno, "Conditioning of quasi-Newton methods for function minimization", Math. Comp. 24, 647–656 (1970) https://www.ams.org/journals/mcom/1970-24-111/S0025-5718-1970-0274029-X/