Project Report - Discovering the Higgs Particle

Francesco Intoci, Mattia Mariantoni, Theresa Stadler
Department of Computer Science, EPFL, Switzerland

***Abstract*—This short report summarises our attempt to build a robust and accurate predictive model to distinguish the decay signature of the Higgs boson from that of other fast decaying particles. We find that already simple linear classification methods achieve a better than baseline performance and that feature selection can further improve classification accuracy.**

I. Introduction

The discovery of the Higgs boson [2] at the Large Hadron Collider (LHC) in 2012 [1] ended the hunt for the "God particle" that lasted for nearly 50 years [3]. A main challenge the scientists at the LHC had to overcome is that one can never directly observe a Higgs boson. The Higgs boson, like most particles, is unstable and, immediately after being produced, undergoes a process called particle decay. However, as collisions of protons produce a whole host of unstable particles, finding the unique *decay signature* of the Higgs boson is a challenging classification task.

A. Task Description

In this short report, we describe our attempt to reproduce the search for the Higgs boson. Given access to an open-source dataset containing the decay signatures of collision events, we attempted to build an accurate and robust prediction model to distinguish the Higgs boson's decay trace from other particles.

II. Background

A. Data Description and Processing

The data provided comprised 250,000 labelled training examples and 568,238 unlabelled test records generated by the ATLAS full detector simulator [4]. We denote as $\mathcal{D} = (X, \mathbf{y})$ a dataset where $X \in \mathbb{R}^{N \times D}$ is a matrix with the i-th row containing the D-dimensional feature vector $\mathbf{x}_i^T$ of the i-the record and $\mathbf{y}$ is a column vector that contains the labels $y_i \in \{b, s\}$ of all N records. In the original dataset, each feature vector $\mathbf{x}_i$ contained $d = 30$ numerical features of which 29 were continuous and 1 discrete.

Feature selection. Section III presents experimental results on the classification performance across different feature sets. Here, we describe in short the different feature engineering strategies that were tested[1].

MostInfo. All models were tested on a reduced set of features that only pertained the 20 features with the highest linear correlation with the target variable.

ImputeJet. 7 out of the 29 continuous data features were undefined for a large number of records ($177,457$ of $250,000$ labelled records) with `PRI_jet_num` ≤ 1 [5]. In the original data, undefined values were encoded as -999. We ran all models on a dataset in which undefined value where imputed with the mean of defined values to avoid a potential skew of weights towards these outliers.

ExpandPoly. To enable non-linear classification boundaries, we tested a polynomial basis expansion of the data with up to a maximum degree of 4.

Standardisation. If not specified otherwise, all data was standardised to the mean and standard deviation (s.d.) of the *current training set* before training. If cross-validation was applied, the mean and s.d. of the current training set were used to avoid optimism bias [6].

B. Classification Methods

Formally, the task of this challenge was to find a *binary classification function* $f : \mathbb{R}^d \to \{b, s\}$ that takes as input a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and outputs a guess about the correct label $\hat{y}_i \in \{b, s\}$. To obtain f we run a *training algorithm* $\texttt{Algo} : (\mathcal{D}, \theta) \to f_{\texttt{Algo},\mathcal{D}}$ on a dataset $\mathcal{D}$ to obtain a trained classifier $f_{\texttt{Algo},\mathcal{D}}$ where θ is a set of training hyperparameters. We describe in short the four main training algorithms implemented and tested as part of this project.

`Base` The `Base` algorithm extracts relative class frequencies from its training set and given a non-labelled record outputs a random label according to

$$f_{\texttt{Base},\mathcal{D}}(\mathbf{x}_i) = \begin{cases} b & \text{with } \pi_b = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{y_n = b} \\ s & \text{with } \pi_s = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{y_n = s} \end{cases} \tag{1}$$

where $N = |\mathcal{D}|$ is the size of the classifiers training set and π_b, π_s are the probabilities of observing each class.

`LS` Given a dataset $\mathcal{D} = (X, \mathbf{y})$, the linear least-squares algorithm `LS` outputs a model

$$f_{\texttt{LS},\mathcal{D}}(\mathbf{x}_i) = g(\mathbf{x}_i^T \mathbf{w}_{\texttt{LS}}) \tag{2}$$

where $\mathbf{w}_{\texttt{LS}} = (X^T X)^{-1} X^T \mathbf{y}$ is the weights vector that minimises the mean squared error (MSE) between true and predicted labels in the train set $\mathcal{D}$ and g is a threshold function that outputs $\hat{y}_i = b$ if $\mathbf{x}_i^T \mathbf{w}_{\texttt{LS}} \leq 0$ and $\hat{y}_i = s$ otherwise.

`Ridge` The ridge regression algorithm `Ridge` is a variant of least-squares training in which the optimal weights vector is additionally constrained to minimise the L_2 norm of the weights vector $||\mathbf{w}||_2^2$ with $\mathbf{w}_{\texttt{Ridge}} = (X^T X + 2N\lambda I)^{-1} X^T \mathbf{y}$. λ is a training hyperparameter that controls the trade-off between finding the optimal MSE and parameter shrinkage.

`LogReg` A classifier trained under the logistic regression algorithm `LogReg` outputs $\hat{y}_i = b$ if $\mathbf{x}_i^T \mathbf{w}_{\texttt{LogReg}} \leq \frac{1}{2}$ and

[1] See also `Run.ipynb` and `Data Exploration.ipynb` for details

$\hat{y}_i = s$ otherwise. The optimal weights vector is found by minimising the negative log-likelihood (NLL) of the train set under a logistic regression model with

$$\mathbf{w}_{\texttt{LogReg}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \log[1 + \exp(\mathbf{x}_i^T \mathbf{w})] - y_i \mathbf{x}_i^T \mathbf{w} \quad (3)$$

We implemented gradient descent (GD) on a variant of the NLL loss that is *normalised to the size of the train set*. This enabled us to find the optimal step size γ through cross-validation independent of the train set size. GD was initialised with a weights vector of all 0s.

C. Evaluation Setup

The original challenge used a formal objective function referred to as *approximate median significance* [4]. We used a simpler evaluation metric to select the best performing model. Each trained model was evaluated based on its *prediction accuracy* on a hold-out set not used during training.
The prediction accuracy over a test set $\mathcal{D}_{\texttt{test}}$ is defined as

$$\texttt{Accuracy} = \mathop{\mathbb{E}}_{\mathcal{D}_{\texttt{test}}} [\mathbb{1}_{y_i = \hat{y}_i}] \quad (4)$$

where $\hat{y}_i = f_{\texttt{Algo},\mathcal{D}_{\texttt{train}}}(\mathbf{x}_i)$ is the predictions produced by a trained model evaluated over the test set and $\mathbb{1}_{y_i=\hat{y}_i}$ the indicator function equal to 1 if $y_i = \hat{y}_i$ and 0 otherwise.

Evaluation set. To avoid overfitting to the final test set and enable error analysis, a hold-out set was used for model selection. To create this evaluation set, the 250,000 labelled records contained in `train.csv` were split into a training set $\mathcal{D}_{\texttt{train}} = (X_{\texttt{train}}, \mathbf{y}_{\texttt{train}})$ with $|\mathcal{D}_{\texttt{train}}| = 175,000$ and an evaluation set $\mathcal{D}_{\texttt{eval}} = (X_{\texttt{eval}}, \mathbf{y}_{\texttt{eval}})$ with $|\mathcal{D}_{\texttt{eval}}| = 75,000$. In subsequent experiments, $\mathcal{D}_{\texttt{train}}$ was used to train a candidate classifier $f_{\texttt{Algo},\mathcal{D}_{\texttt{train}}}$. The performance of each trained classifier was then evaluated as the prediction accuracy on the hold-out set $\mathcal{D}_{\texttt{eval}}$.

Cross-validation for hyperparameter and feature selection. As described above, some training algorithms take as input a set of hyperparameters θ. To find the optimal set of hyperparameters θ^*, k-fold cross-validation with $k = 4$ over $\mathcal{D}_{\texttt{train}}$ was used. θ^* was selected as the parameter set with the *minimum average misclassification rate* defined as $1 - \texttt{Accuracy}$ over all k test sets. The same procedure was used to select the best performing feature set for each model.

After model and hyperparameter selection, a final model was trained on the combined train set $f_{\texttt{Algo},\{\mathcal{D}_{\texttt{train}},\mathcal{D}_{\texttt{eval}}\}}$ and the test set predictions were submitted to the challenge platform.

III. Results

Cross-validation on $\mathcal{D}_{\texttt{train}}$ showed that expanding the dataset with a polynomial basis with maximum degree 4 (labelled `ExpandPoly` in Fig. 1) yielded the highest accuracy across all classifiers. Average test accuracy increased from 72.7% on the raw to 75.49% on the modified dataset under a `LogReg` model with $\gamma = 0.01$. Removing uninformative features decreased average classification performance to 71.41%. Imputing undefined values did not significantly affect performance.[2].

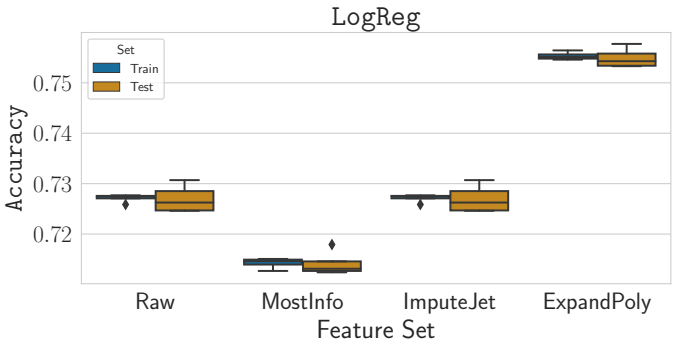

Fig. 1. Prediction accuracy across feature sets for the best performing model `LogReg` trained under gradient descent with $\gamma = 0.01$

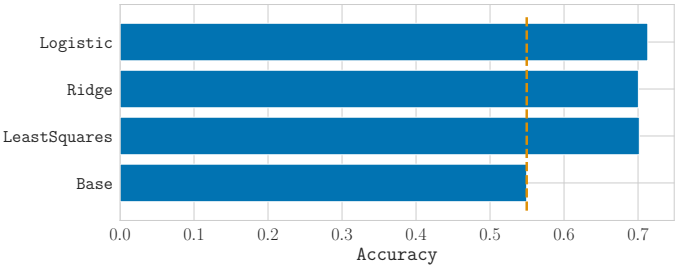

Fig. 2. Comparison of prediction accuracy on the pre-processed dataset $\mathcal{D}_{\texttt{eval}}$.

Based on these results, we compared the performance of all training algorithms on the expanded dataset. Fig. 2 shows the prediction accuracy evaluated over $\mathcal{D}_{\texttt{eval}}$. Cross-validation over $\mathcal{D}_{\texttt{train}}$ was used to find the optimal step size $\gamma = 0.01$ for `LogReg` training and the optimal trade-off parameter $\lambda = 4.8 \cdot 10{-}4$ for ridge regression.
All models significantly outperformed the baseline `Base` with 54.95%. The logistic regression model `LogReg` reached the highest prediction accuracy with 71.32%. After training a model with the same parameter settings over the combined train set (all labelled records), test set accuracy reported by the challenge platform was 75.7%.

IV. Discussion

Our experimental results highlight the complexity of the classification task at hand. None of the simplistic feature processing steps yielded a satisfying improvement in classification accuracy. Other feature selection strategies, such as removing redundant features with high cross-correlation, not discussed here, neither improved performance. More work would be needed to understand the physical meaning of the data features to add more informative or remove redundant features.
The most surprising result we obtained is the relatively good performance of simple models such as the (regularised) MSE estimator `Ridge`. It is unexpected that models trained via `LogReg` did not outperform this simple model with a larger margin. A likely explanation is that due to computational

[2]Results on all models and further detailed analysis in `Run.ipynb`

constraints `LogReg` training was terminated before loss convergence (after a maximum number of $20,000$ iterations) and did thus not converge to its optimum solution.

REFERENCES

[1] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. A. Khalek, A. A. Abdelalim, R. Aben, B. Abi, M. Abolins, O. AbouZeid *et al.*, "Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc," *Physics Letters B*, vol. 716, no. 1, pp. 1–29, 2012.

[2] P. W. Higgs, "Broken symmetries and the masses of gauge bosons," *Physical Review Letters*, vol. 13, no. 16, p. 508, 1964.

[3] L. P. at Ten, "The Higgs boson: Revealing nature's secrets."

[4] C. Adam-Bourdarios, G. Cowan, C. German, I. Guyon, B. Kegl, and D. Rousseau, "Learning to discover: the higgs boson machine learning challenge," 2014, accessed 2020-10-01.

[5] ATLAS collaboration, "Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014." 10.7483/OPENDATA.ATLAS.ZBP2.M5T8, 2014.

[6] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.