Machine Learning Project1

Zehao Chen, Haolong Li, Xuehan Tang
EPFL, Switzerland

***Abstract*—In this paper we research the discovery of the Higgs boson particle at the LHC at CERN using machine learning techniques. Firstly, we group the data into 3 groups and each group has different levels of missing values. Then, we preprocess the dataset, including standardizing, cleaning the data, and building polynomial and crossing basis vectors. To predict the labels, we train a regularized logistic regression model for each group and introduce multi-process programming to accelerate grid searching. The best model we found yields a 0.836 accuracy and a 0.755 F1 score on the AIcrowd leaderboard.**

I. Introduction

The aim of this project is to apply machine learning techniques to actual CERN particle accelerator data to recreate the process of "discovering" the Higgs particle. There is a training dataset with different experiments and its characteristics. And we will do several operations to original dataset. We have to deal with missing values, remove related features, apply different methods and estimate how well is our method. The final goal to find out a best method that has the highest accruacy.

II. EXPLORATORY DATA ANALYSIS

After importing the training set, we find that there are 250,000 events and 30 features. For the prediction values, there are only 'b' for background and 's' for signal. The proportion of Y $\in \{b, c\}$ is 65.73% and 34.27% respectively.

To deal with missing values, we use the median of the rest of the values for the feature.

After observing features, the number of jets of the event can determine the missing values.

• If it had 0 or 1 jets, there will be some missing values in some features.

• If it had either 2 or 3 jets, there are no missing values.

Then, we use 3 methods to deal with different categories.

III. Data Preprocessing

A. Grouping Data

The feature data is divided into 4 categories, with `PRI_jet_num = 0, 1, 2, 3.`, upon inspection, we notice that:

1) `PRI_jet_num = 0`: A specific set S of the features presents missing values.
2) `PRI_jet_num = 1`: A specific subset of the features presents missing values.
3) `PRI_jet_num = 2, 3`: There are no missing values.

We hence group the data into 3 groups, with `PRI_jet_num = 0; 1; 2, 3` for better prediction performance.

After the grouping, we remove the `PRI_jet_num` feature because it no longer provides anymore information than grouping.

B. Feature Processing

1) Removing 0-and-NA-filled features: We notice that some columns are filled with NA values (in the actual data set, elements with value -999.) and 0. These columns are not informative and will also introduce zero standard error when normalizing. Thus, for features that only contain 0 and/or NA values, we discard them.

2) Removing correlating features: After checking the correlation matrix of the different features, we discovered that some features have high correlation with each other (higher than 0.9). In order to avoid singular matrices during computation, we remove the features $feature\_i, feature\_j$ in which $corr(feature\_i, feature\_j) > 0.9$.

3) Feature augmentation: For more flexibility, we augment our features using polynomial expansion and also introduce cross terms, i.e. for features $x_1; x_2; ...; x_m$, we augment them into:

$$x_1, x_1^2, ..., x_1^{deg}; \quad x_2, x_2^2, ..., x_2^{deg}; \quad ...; \quad x_m, x_m^2, ..., x_m^{deg};$$
$$x_1x_2; x_1x_3; ...; x_1x_m; \quad x_2x_3; x_2x_4; ...; x_2x_m;$$
$$...; \quad x_{m-1}x_m$$

Where deg is the hyperparameter describing the scale of polynomial expansion.

4) Normalization: For each feature f_i, we normalize the feature with:

$$f_i = \frac{f_i - \mu_i}{\sigma_i}$$

Where μ_i and σ_i are the mean and standard error without NA values.

C. Handling missing values

For several NA data that are present in features, we simply replace them as 0.

IV. Models and Training

A. Training Algorithm

We simply try to predict the output using several linear regression algorithms and regularized logistic regression algorithm. Results are shown as Table I (without adjusting hyperparameters).

Algorithm name	Accuracy
Linear regression using gradient descent	0.683
Linear regression using stochastic gradient descent	0.678
Ridge regression using normal equations	0.729
Regularized logistic regression algorithm	0.834

TABLE I
ACCURACY UNDER DIFFERENT PREDICTING ALGORITHMS

It's advisable to filter algorithms whose performance is awful to reduce our workload in the future. Thus, we select regularized logistic regression algorithm, which performs best among all algorithms, as the predicting model. The update function of it is defined as Equation 4 and supplementary definitions are shown in Equation 1˜ 3. This algorithm introduces sigmoid function to overcome awful performance under extreme values and unbalanced data, which is a obvious drawback in other linear regression algorithms.

$$L(w) = 2\lambda w + \frac{1}{N}\sum_{n=1}^{N} -y_n x_n^T w + log(1 + e^{x_n^T w}) \quad (1)$$

$$\sigma(x) = \frac{e^x}{1+e^x} \quad (2)$$

$$\nabla^2 L(w) = 2\lambda + \frac{1}{N}\sum_{n=1}^{N} \sigma(x_n^T w)(1-\sigma(x_n^T w))x_n x_n^T \quad (3)$$

$$w_{t+1} = w_t - \lambda \nabla^2 L(w_t)^{-1} \nabla L(w_t) \quad (4)$$

B. Validation Dataset

To avoid over-fitting, we divide our data obtained from train.csv into 2 parts: training dataset (80%) and validation set (20%). We train the model for 200 iterations and pick the hyperparameters that optimizes the accuracy.

C. Hyperparameters Selection

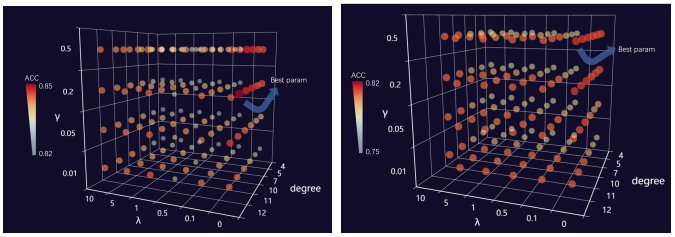
According to regularized logistic regression algorithm, there are 3 vital hyperparameters: λ: it decides how do we penalize the absolute number or square of w to avoid over-fitting, degree: it decides how many degrees will the x be extended, γ: it decides how fast will our model learn.

To select best hyperparameters for our model, we perform grid searching. But as we know, grid searching is really time-consuming. To settle this problem, we adopt **multi-process programming** to run algorithms under different hyperparameters concurrently. It is obvious that algorithms under different hyperparameters are orthogonal so they are perfectly suitable for being conducted simultaneously. After applying multi-process programming and running python scripts on a server with Intel(R) Xeon(R) Gold 6148 CPU (40 cores), we gain 3x improvement in executing time.

V. SUMMARY

After paying a lot of attention to the data processing and augmentation, we separately search the optimal hyperparameters in 3 categories we mentioned in Section III-A as is shown in Figure 1. We select the model which gain highest accuracy on the validation dataset as the our final hyperparameters, as shown in II. The submission scored 0.836 in accuracy and 0.755 in F1 score on the AIcrowd leaderboard. The relationship between accuracy and iterations is shown as Figure 2. Accuracy drops rapidly after several iterations because the inverse matrix in Equation 4 is not solvable.

In future work, deep learning can be introduced to solve problems like it. The great descriptive power of deep learning architectures and their capabilities to hierarchically describe complex data and signals into high make them suitable to solve problems with higher dimensions [1].



(a) Category 0 (b) Category 1

(c) Category 2

Fig. 1. Optimal grid searching hyperparameters

	λ	degree	γ	Accuracy
category 0	0	11	0.2	0.8496
category 1	0	12	0.5	0.8119
category 2	0	10	0.5	0.8449

TABLE II
HYPERPARAMETER SELECTIONS OF DIFFERENT SUBMODELS

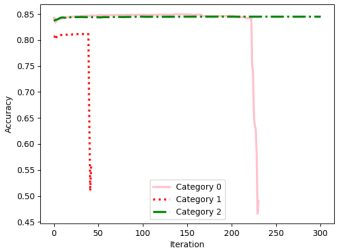


Fig. 2. The relationship between accuracy and iterations

REFERENCES

[1] A. PICON RUIZ, A. ALVAREZ GILA, U. Irusta, J. ECHAZARRA HUGUET *et al.*, "Why deep learning performs better than classical machine learning?" *Dyna Ingenieria E Industria*, 2020.