Machine Learning Project 1

James Bardet, Nicolas Delamaide, Ilyas Benadada
CS-433 EPFL

***Abstract*—The goal of this project was to implement a classification model that is able to predict whether or not a Higgs boson was created in the LHC at CERN. We pre-processed the data, implemented six models for the task and did feature engineering. We achieved an accuracy of 82.4% using least squares and augmenting the feature vector with a polynomial of basis 13.**

I. Introduction

In this paper we will explore a dataset from CERN particle accelerator and try to use machine learning methods to predict if a Higgs boson was created during the acceleration process.

A. Exploratory data analysis

The data consists of 250'000 data points with 30 features and are accompanied with a prediction that take only two values: s and b. To have a numerical prediction, we assume that s is corresponding to 1 and b corresponding to -1.

B. Categorical feature

We have analysed the type and name of all our features. We have found that all our features have floating values except for *PRI jet num* which is an integer ranging from 0 to 3.

II. Models and methods

A. Feature processing and selection

1) Finding and replacing Missing data:
We noticed that some of the features, meaning some of the columns of our data, contain the value -999, which is far from the mean or median of the column. Therefore, we can consider this value as a missing data. The -999 values data are inconsistent and should either be replaced or deleted. If the feature contains a lot of missing data then it should be excluded because this feature will only worsen the computational cost of our data. We have spotted all the missing data of our train data, then we calculated the percentage of missing data (-999 value) of each feature. In addition, we have set a threshold of 70 percent to know if we should replace the missing values or exclude the feature. By consequence features that have more than 70 percent of missing data will be deleted, in the other cases the missing data will be replaced with the median of the feature. We have chosen the median and not the mean because it is more robust with respect to outliers.

2) Standarization:
Each feature has its own range of values which may mislead the prediction result that will take into account only the features that have big values. Consequently, we will apply Standarization to all features expect *PRI jet num*.

To standardize our data, we will first demean each feature and then divide it by the standard deviation.

$$standardized\, x_j = (x_j - \mu_j)/\sigma_j \tag{1}$$

with x_j the j_{th} feature, μ_j the mean of the j_{th} feature and σ_j the standard deviation of the corresponding feature.

3) Correlation matrix:
In order to reduce the dimensions and computational cost of our problem, we have computed a correlation matrix that calculates the correlation of all our features. The correlation values are between -1 and 1, and two features are highly correlated if the absolute value of their correlation is near 1. Thus, we have set a correlation threshold equal to 0.9 to spot the values that are highly correlated.

B. Models

- Least squares optimized by gradient descent (GD) and by stochastic gradient descent (SGD)

We optimized the two above models over the Mean Squared Error (MSE) cost function $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\intercal \mathbf{w})^2$ to find the best weight vector $\mathbf{w}$, with N the number of samples. The updates rules at each iteration are :

$$\begin{aligned} \mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^t) \qquad (GD) \\ \mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}_n(\mathbf{w}^t) \qquad (SGD) \end{aligned} \tag{2}$$

where n is chosen at random among the N samples.

- Least squares and Ridge regression using normal equations

To find the best weight vector $\mathbf{w}$, we simply solved the normal equations using the Numpy solver for linear systems. The normal equations to solve are respectively :

$$\begin{aligned} \mathbf{X}^\intercal \mathbf{X}\mathbf{w} &= \mathbf{X}\mathbf{y} \\ (\mathbf{X}^\intercal \mathbf{X} + 2N * \lambda \mathbf{I})\mathbf{w} &= \mathbf{X}\mathbf{y} \end{aligned} \tag{3}$$

We can see that ridge regression is simply least squares using L2-regularization, that is we add the penalty term $\lambda\|\mathbf{w}\|^2$ to the loss to penalize large model weights.

- Logistic regression and Regularized logistic regression optimized by GD

The loss for logistic regression is $\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} ln[1 + exp(\mathbf{x}_n^\intercal \mathbf{w})] - y_n \mathbf{x}_n^\intercal \mathbf{w}$. The loss for the regularized logistic regression is the same but with the L2-regularization penalty. At each iteration we update the weight $\mathbf{w}$ according to the update rule for GD given in equation (2).

Parameter selection. To find the model parameters γ and λ we did a grid search over the ranges $[10^{-5}, 10^{-1}]$ and $[10^{-5}, 0]$ respectively. For each value we performed 4-fold cross validation and chose the model parameters that resulted in the lowest train and test loss. We also compared the accuracy of our models using 4-fold cross validation.

Feature augmentation. To reduce over-fitting with our linear models and try to make our data more linearly separable we augmented our feature vector. For that we added a polynomial degree of arbitrary degree M : $\phi(\mathbf{X_i}) = [\mathbf{X_i}, \mathbf{X_i}^2, ..., \mathbf{X_i}^M]$ where $\mathbf{X_i}$ is a column of the feature matrix $\mathbf{X}$. The degree M was selected using a grid search from 2 to 15. However we only used the resulting extended feature matrix in conjunction with least squares (using normal equations) and ridge regression. We did not use this trick with other methods because raising the features to powers caused overflows during the optimization process.

Decision boundary. Lastly, we looked at the Receiver Operator Characteristic (ROC) of the method that yielded the best results on the classification to fine-tune the decision boundary.

III. Results

Following our grid search for the γ and λ model parameters, we chose for each model the parameters that resulted in the smallest train and test loss. The γ values for least squares GD, least squares SGD, logistic regression and regularized logistic regression are respectively : 10^{-1}, $4{*}10^{-3}$, 10^{-5}, 10^{-5}. The λ values for ridge regression and regularized logistic regression are both 10^{-4}.

The results of the 4-fold cross validation of the accuracy of our models using the parameters described above are summarized on Table I. These values will serve as baselines for comparing our feature engineering choices.

Model	Mean accuracy $\pm$ std
Least Squares GD	$74.46\% \pm 0.11$
Least Squares SGD	$69.05\% \pm 0.95$
Least squares	$74.65\% \pm 0.12$
Ridge regression	$74.65\% \pm 0.12$
Logistic regression	$75.16\% \pm 0.10$
Regularized logistic regression	$75.16\% \pm 0.10$

TABLE I
MEAN ACCURACY OF THE METHODS AFTER 4-FOLD CROSS VALIDATION.

To try to get better results, we augmented our features using polynomials as we described earlier. To choose the best polynomial basis, we performed a grid search from a degree 2 to 15. For least_squares the best accuracy was 82.21% using a polynomial basis of 13 while for ridge we achieved an accuracy of 82.14% for the basis 11.

Finally, we computed the False Positive Rate and True Positive Rate for different decision thresholds (ranging from -1 to 1) to plot the ROC curve of the method with the highest accuracy, least squares. The curve can be seen on Figure 1. We obtained an Area Under the Curve (AUC) of 0.69.

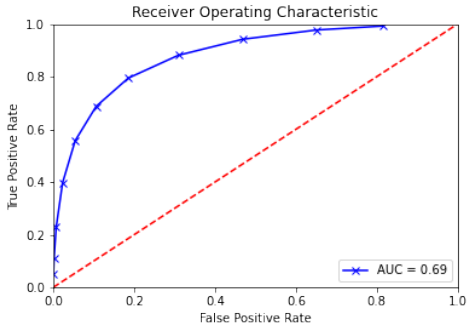

Fig. 1. ROC curve for least squares method.

IV. Discussion

From Table I we can see that the best methods are the logistic regressions with accuracies of 75.16% on the non-augmented data. The linear methods are not far behind with accuracies around 74.5%. However, they take the lead when we augment the data using polynomials, achieving accuracies north of 82%, especially least squares and ridge using normal equations. This is because adding features when we use linear models reduces over-fitting and can increase the linear separability of the data. We also used the polynomial features with logistic regression but it produced worse accuracies at around 66%, which is probably due to the fact that the data became linearly separable and we know that logistic regression doesn't perform well in this case. One of the drawbacks we encountered with augmenting our features with polynomials is that we couldn't use the least squares methods with GD and SGD because it caused overflows in the computation on the gradient. This didn't happen with least squares and ridge because they solve normal equations.

Our best model is thus least squares using normal equations with augmented features using a polynomial basis of degree 13. Since the goal of this project was to detect the Higgs boson, we thought that having few false negatives, or high sensitivity, would be better for this application. This resulted in us doing the ROC curve of Figure 1. However we left the default decision threshold of 0 for our submission as it provided the best accuracy, with a sensitivity of 68.7% and specificity of 89.4%.

We also tried to engineer other features, such as applying the following transformations on the data : logarithm, sinus, absolute value and inverse. Unfortunately we didn't get better accuracies than the best model we described previously. Nevertheless, we could probably improve our accuracy by doing better feature engineering our maybe better data preprocessing. Alternatively, other models might be better suited for this classification task.

V. Summary

In the end, least square using normal equations is the best model when paired with features augmented with a polynomial of degree 13. It achieved 82.4% accuracy and a F1 of 0.728 on the given test set.