Machine Learning Project 1
Higgs-Boson-Challenge

Vincent Dandenault, Fabrice Nemo, and Jacob Schillemans

Department of Computer Science, EPFL, Switzerland

***Abstract*—Machine Learning is a whole set of techniques that leverage data in order to solve complex tasks, notably classification and regression problems. In this paper, we present our approach to solving the classification problem around the Higgs Boson dataset simulated by the ATLAS experiment from CERN, elaborated in the context of the class CS-433 at EPFL.**

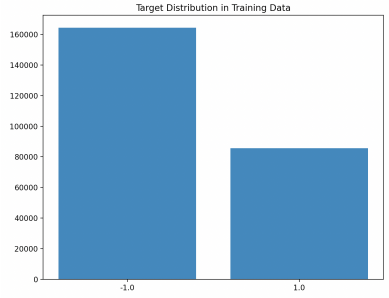
I. Introduction

The goal of the Higgs Boson challenge is to build a classifier capable of predicting the presence or the absence of the particle, based on a series of features provided for each example. Given this classification task, we implemented a series of algorithms in the hopes of finding the best classifier. In fact, we have implemented Gradient Descent, Stochastic Gradient Descent, exact Least Squares (solving normal equations), Ridge Regression, Logistic Regression and Regularized Logistic Regression. In this paper, we first briefly present each algorithm that we implemented, followed by the top 3 performing algorithms. We then discuss additional steps we took to refine our models and make them more performant.

II. Data Exploration

The first step of elaborating the machine learning model is to understand and characterize the input data, as well as the target feature desired. First, the dataset has a high number of features (30), so this might pose a challenge to our algorithms, notably the Logistic Regression algorithm, as it assumes an absence of multicollinearity in the feature vectors. Second, the distribution of labels inside our training data seemed to be unbalanced (see Fig. 1). Thus, we set out to find methods to compensate for this problem with the following approaches (see Discussion section).

Figure 1. Imbalanced target classes in the training dataset



III. Models and Methods

In this section, we present the models used for the task of prediction of the presence or the absence of the Higgs Boson.

A. Gradient Descent and Stochastic Gradient Descent for Least Squares

We first define $\mathcal{L}(w) = \frac{1}{2N}(y - Xw)^\top (y - Xw)$ as our loss function, where N is the size of the data set, y_n is the ground truth label for the data n and $x_n^\top w$ is the predicted label. Its gradient is $\nabla\mathcal{L} = -\frac{1}{N}X^\top(y - Xw)$. The idea of gradient descent is to start from an arbitrary w (initial weights) and compute, $\nabla\mathcal{L}(w)$ in order to determine iteratively, the w with the lowest loss. We update w accordingly to: $w := w - \gamma\nabla\mathcal{L}(w)$, again and again, reducing the loss function at each step. γ is a parameter to determine by what scale we want to change w at each step, if it is too big, the loss function is so that it would lead to a w with a bigger loss, if it is too small, learning is too slow.

B. Normal equations for solving Least Squares and Ridge Regression exactly

The normal equations for least squares are constructed by forming the Gram matrix $A = X^T X$ and $b = X^T y$. Then the linear system is solved by NumPy's package $w^* = np.linalg.solve(A, b)$, this method uses QR decomposition and is very robust. To avoid overfitting by augmenting the feature vector, we implement ridge regression that favors simple models through regularization. w^*_{ridge} is then found by solving the normal equation in the same manner as before, only now the Gram matrix is shifted by $\lambda' * I$ where $\lambda = 2\lambda N$ and N is the number of samples. By 'lifting' the eigenvalues this way, we also fight ill-conditioning.

C. Logistic Regression with Stochastic Gradient Descent

Given our large dataset, we implemented our logistic regression using the Stochastic Gradient Descent (SDG) method [1]. This algorithm uses a standard logistic function (sigmoid) in order to map an input to a binary output. Given the prior knowledge that we had on our data, we decided to use the standard decision boundary of 0.5 to

classify the outputs.

D. Regularized Ridge Regression

Furthermore, because our original data had 30 features, we decided to limit the capacity of our model by introducing a regularization term in order to avoid overfitting the model. Thus, we added a regularization term (λ) [1], limiting the number of parameters our model can have, which in turn allows our model to generalize better to unseen data.

IV. Results

In this section, we present the best results from our top three performing models. To measure these classifiers, used two common loss functions: least squares (the square of the error) and cross entropy loss (for the logistic regression models). Finally, we submitted our model's prediction to a hidden validation dataset, which in turn calculated the accuracy and F1 Score of our model. Our top 3 performing models on the hidden validation dataset are defined as such:

1) RRFE: Ridge Regression with feature engineering and pre-processing
2) LSWFE: Least Square model with feature engineering and pre-processing (see Discussion section)
3) LSMSGD: Least Square Model with Stochastic Gradient Descent ($\gamma = 3.10^{-7}$, initial w set as the null vector, 50000 iterations. The gradient is computed with a random subset with 1% of the train set)

Models	Accuracy	F1-Score
RRFE	0.823	0.728
LSWFE	0.764	0.662
LSMSGD	0.736	0.536

V. Discussion

In this section, we elaborate on the different approaches we used to improve our models performances.

A. Pre-processing

As the challenge describes [2], variables may be undefined(value = -999.0) so we imputed those values by the mean of the feature. Whether these values were undefined depended on the PRI_jet_num feature, so another option was to split the data up in different subsets. We replaced these with values on the edge of a chosen percentile of the feature. Using this approach in a preprocessing step improved the accuracy of our models.

B. Cross Validation

Given that we were limited in the submissions of our models, we implemented a 5-fold cross validation to evaluate the performance of our algorithms with the average over the fold test accuracies. This allowed us to better select the best model before submitting to the hidden validation dataset. Furthermore, we can combine this approach with grid search to tune the hyperparameters, again, allowing us to select a better model (see Hyperparameter tuning section).

C. Standardization

We introduced the standardization of our input data to our models and we noticed that, while it has been reported in the literature that standardization of the continuous features helps to improve the quality of the data, this trick did not give us a better accuracy when validating our models. Thus, we decided to not use it in our final submissions.

D. Polynomial Expansion

We implemented polynomial expansion as a method of feature engineering, to increase to make the linear models more powerful. This allows our algorithm to search in more features and select the best ones, thus exploring non-linear relationships between the features and the target.

E. Correlated features

After doing some research on the data [3], it turned out that there were some feature vectors were highly correlated. This allowed us to ignore the three subleading features. Ignoring other columns didn't improve the accuracy, but made the calculations much more computationally efficient.

F. Hyperparameter tuning

Lastly, some of our models included hyperparameters that were originally selected from the literature. However, these parameters can be optimized for this specific problem [4]. For our project, we selected a grid search for our hyperparameter search, as this approach seemed feasible given our model capacity and the complexity of our data. To do so, we simply declared a large grid of potential values of hyperparameters, again based on the literature, and systematically searched over each combination of hyperparameters. In sum, we kept the hyperparameters that yielded the lowest loss and the best accuracy score, derived from our cross-validation approach.

VI. Conclusion

In conclusion, we elaborated a few traditional machine learning algorithms to solve our problem. We then implemented polynomial expansion to improve on our results, as well as standardized our dataset and included a hyperparameter search for some of our models. We then implemented a cross validation algorithm for internal validation of our solution. Eventually, our best model (RRFE) yields an accuracy of 0.822% and an F1 score of 0.722%.

REFERENCES

[1] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.

[2] C. Adam-Bourdariosa, G. Cowanb, C. Germain, I. Guyond, B. Kégl, and D. Rousseau, "Higgs challenge," 2014.

[3] Shubrendra, "Higgs boson analysis," https://www.kaggle.com/code/shubhendra7/higgs-boson-analysis, 2018.

[4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.