Higgs Boson Prediction Using Machine Learning

Francisco Cloz, Abhi Kamboj, and Cary Chai
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—Machine learning is becoming more prevalent in today's society and is often used by a wide variety of disciplines to provide insights into large data sets. Using data from CERN's Large Hadron Collider on the Higgs Boson particle, this project explores the applications of machine learning in a particular case study in physics by providing an understanding of six machine learning models for a binary data set. This report details analyses and procedures for preconditioning the data as well as the results of which algorithm provided the least error in identifying Higgs Boson particles.**

I. Introduction

The Higgs Boson is an elementary particle which was announced in 2013 and explains the presence of mass in other particles. It was discovered at CERN's Large Hadron Collider by smashing protons together at high speeds which produces smaller particles. The Higgs Boson is one such particle and can be detected by its decay signature despite it resembling those of other background signatures. In the conducted experiments, CERN collected large amounts of data on features produced by these collisions. The challenge of this project is to apply machine learning to CERN's data set to correctly identify Higgs Boson particles.

In this project, six machine learning algorithms were run on a variety of preconditioned data sets in order to identify the optimal algorithm for identifying Higgs Boson particles. The techniques used are detailed in the Precondition and Classification sections.

II. Classification Models

To avoid overfitting, when comparing our algorithms, the labeled data set was divided into training and test sets. Each algorithm was then compared on the basis of its error with respect to the test set.

Model Performance	
Optimizer	Error
Least Squares GD	43.674%
Least Squares SGD	25.924%
LS	25.556%
Linear Ridge	25.819%
LogisticSGD	26.311%
Reg LogisticSGD	30.785%

Figure 1: In testing, it was found the MinMax Normalization and Mean Filling were optimal. Linear models also worked better than logistic ones

A. Gradient and Stochastic Gradient Descent

Gradient descent is a general technique that can be used to update the parameters of various models and cost functions. We used gradient descent (GD), stochastic gradient descent (SGD), and minibatch SGD to evaluate linear and logistic models with the following loss functions: Mean Squared Error (MSE), Mean Absolute Error (MAE), LogCosh, Quantile, and Huber. The best combination was determined to be the linear model using SGD with a LogCosh loss fuction. SGD was used because it calculates quicker and explores more of the loss function. LogCosh behaves like MSE when it is close to zero and behaves like MAE when it's farther from zero providing the best of both functions. Quantile didn't work very well empirically and Huber performed similarly to LogCosh.

B. Least Squares

This was the fastest technique since it solves a set of normal equations to provide a closed form solution for the optimal parameters of a linear model with a Means Squared Error Loss function. This performed faster than gradient descent and similar to gradient descent with MSE loss functions.

C. Ridge Regression

Like Least Squares we solve a set of normal equations of a linear model using an MSE loss function however complex models are penalized by adding the L_2 norm of the parameters scaled by λ to the loss function. As shown in Table 1, this performed slightly better to Least Squares but requires more testing to determine a decent value of λ.

D. Logistic Regression

The logistic function transforms the estimate to a value between 0 and 1 allowing for a more precise error in the loss function of a binary classification model. However, when the regression was run with GD and SGD, it performed much slower than the other models and had a slightly larger error (reference Figure 1). Our implementation was further expanded to do regularized linear regression. Analogous to the Ridge Regression regularizing Least Squares, the regularization of the Logistic regression uses a regularization parameter to penalize complex models to reduce overfitting.

III. Preconditioning

A. Filling in Missing Data

After observing the data set, it was deduced that -999 values were NAN values (unknown) due to their inconsistency with the data. To confirm this, we ran a percentage analysis of each feature for the number of -999 values with some columns having upwards of 70%. This, in addition to the irregularity confirmed that -999 was NAN. Several approaches were taken to deal with the NAN data.

1) **Mean/Median:** Here, the mean for the rest of the data for each feature was taken, ignoring NAN values. For each feature, NAN values were replaced by the calculated mean. A similar process was taken for the median. Each case resulted in accuracy improvements.
2) **Regression Filling:** A model that would be able to predict feature values was trained on the data entries with no missing features. For data entries missing features, the model filled in the NAN values with the predicted values. This method was more complex and time consuming to train but resulted in slightly better accuracy.
3) **Feature Deletion:** Eliminating the features with a 0 or -999 reduced the number of features to 19 which resulted in poor accuracy.

B. Feature Extraction and Augmentation

Five computational methods of feature extraction were tested with varying results. These include removing features with a large variance or correlation.

1) **Plotting:** Initially, all 30 features were plotted against each-other in 2 Dimensional scatter plots. This provided a sense of the data in the CERN data set.
2) **Variance:** Intuitively, if a feature has a large variance it may seem that it is just background noise and can be ignored; however many natural measurements have noise and outliers so it is not enough of a reason to discard the value.
3) **Correlation:** Correlation graphs were created to visualize the data. In this experiment, features which were very highly correlated with another feature were removed as it was likely one feature would be able to predict the other.
4) **LDA:** The LDA algorithm was run on the initial data set as well. The algorithm found a few axes, but they didn't provide much separation between the features within the data.
5) **Recursive Extraction:** Finally, methods for recursively removing features and identifying which functions produced the least errors were created and tested. This method seemed to have the most success and improved the accuracy the most.

C. Normalization

Since the features have different ranges, it is necessary to normalize the data before performing regressions. This prevents features with high values from dominating over features with smaller values when determining the parameters for each model, and also allows GD and SGD to converge faster. The following normalizers were implemented and tested on the various models:

1) **Gaussian Normalizer:** This normalized the features according to the Gaussian distribution and performed slightly worse than other methods.
2) **Decimal Scaling:** This normalization technique scales everything down to a decimal based on the largest value. It performed slightly better than Gaussian Normalizer.
3) **Minmax Normalizer:** This transforms the data to values between 0 and 1 such that the minimum value is mapped to zero and the maximum value is mapped to 1. This performed better than the other normalizers and was therefore used in our final evaluation of the models.
4) **Gaussian Outlier Removal:** This normalization technique removes outliers from specified probability quantiles. This did not perform as well as the other normalizers.

IV. Conclusion

Overall, the linear ridge regression performed best in the tests with the split training data set and is what was used for the competition upload. For preconditioning, a MinMax normalizer and Mean Filler were applied to the data set. The recursive feature extraction also removed feature 11 before running the linear ridge regression. This provided the result of 25.55% error in the online submission.

V. Next Steps

This result was unexpected as our initial belief was that the logistic regression algorithms would produce the least error due to the binary nature of the data set. For further testing, we would like to further explore logistic regression algorithms as well as work more with feature elimination. We would like to explore further implementations of the recursive feature elimination. In addition, we tried creating a neural network which would learn the features' importance. However, it was learning too slowly to be of use. This is something we would like to pursue further in the future though.

Acknowledgements

Thank you to the course staff and professors who have been a great help in teaching machine learning material and in providing support and answers to questions we had throughout the process.