Machine Learning Class Project 1

Daniel Smarda (295269); Anthony Piquet (300541); Pierre Erbacher (300533)
CS-433 Machine Learning, Fall 2018
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract—**This report details the first project of the Fall 2018 session of the EPFL Machine Learning course. Given several hundred thousand samples of particle physics data, the task was a binary classification problem using primarily linear regression and logistic regression models. The purpose of this report is to discuss the entire data pipeline, including cleaning, feature selection and engineering, training, model evaluation and selection, and results. Of the core algorithms considered for the project, least squares classification yielded the best results, with a test accuracy of 73.4%, with significant room for improvement possible in more advanced feature selection methods.**

I. Introduction

Modern physics research conducted on unfathomably small subatomic particles at the Large Hadron Collider (LHC) in Geneva is largely mathematical in nature. As such, in 2014 physicists at LHC formally reached out to the data science community by way of a Kaggle competition, the details of which can be found at [1]. This challenge was adapted and assigned as part of the Machine Learning course at EPFL. The problem assigned for the course closely follows the full 2014 Kaggle competition detailed in [1], with the primary exception of a simpler, common accuracy function used as the scoring function.

The goal is to classify whether a given observation corresponds to unimportant "background" information, or a combination of background and relevant particle information. In this paper we describe our attempt to build a model that can accurately classify this data: data processing, hyper-parameter tuning, and model evaluation and selection. We include anomalies we noticed in our data processing and conclude with a discussion of what future we anticipate would result in performance improvements.

II. Methods

A. Pre-processing

As detailed in the competition paper [1], the dataset consists of a few hundred thousand training samples, each of which consist of 32 features. Of these 32 features, 13 of them are raw measurements while the remaining 19 are values calculated (derived) from the 13 raw values. In our analysis we assumed that the derived features were at least as valuable as the raw values as they reflected a domain knowledge that we lacked.

To clean the data and remove the outliers (marked with values of -999, outside the normal range of all attributes), we first removed any column where the average was below a certain threshold, indicating a large number of -999 values. To determine this threshold, we looked at the highest positive average of any column and took this as a loose bound for a normal magnitude of a column mean. In implementation, this resulted in removal of 12 of the initial 32 columns.

Upon completion of feature selection, we cleaned the dataset by adjusting the remaining -999 values. We initially attempted to remove all rows that contained any -999 values (approximately 18% of the rows in the training dataset), but this function required large amounts of memory (a similar memory error also prevented us from attempting to train a model using degree-2 polynomials of the input parameters as described in class). As an alternative method to neutralize the outliers, we replaced all values of -999 with the mean of the values of the column (excluding the -999 values).

After removing all outliers we mathematically standardized the data so that distance calculations were not disproportionately due to certain features.

B. Cross-Validation

To ensure that our accuracy estimates were accurate and unbiased, and because the dataset was not so large as to make computational resources a problem, we used K-Fold cross-validation with K=10 as recommended in [2], with 90% of the data used for training and the remaining 10% used for validation. Because we did not know if the data provided to us had been ordered in some way (e.g., temporally), we shuffled the dataset randomly before partitioning it, using a constant random seed for reproducibility.

We implemented 6 different algorithms to generate the models: 3 variations of Least square, (gradient descent, stochastic gradient descent, and least square), 2 variations of logistic regression (with and without regularization). As least squares and the algorithms that are similar to it are generally more suited to regression than classification problems, we expected the regularized logistic regression function to yield the best results.

For clarification on what data processing steps we executed as part of cross-validation and which we executed outside of cross validation, the method we followed is summarized below:

1) Load all data.
2) Remove columns with too many outliers
3) Shuffle the data
4) For each fold in cross-validation:
    a) On the training dataset:
        i) Replace outliers with column means
        ii) Standardize the training set
        iii) Train the model (generate weight vector w)
    b) On the test dataset:
        i) Replace outliers with column means

ii) Standardize the test set

iii) Generate predictions from teh trained model

c) Generate a validation accuracy based on the generated predictions

5) Average the test accuracy generated by each fold to find the validation accuracy of the model

Note that data standardization and replacement of outliers was intentionally done after the data was split so that the means and other parameters used for standardization were not affected by the the other data subset in the fold.

III. Results

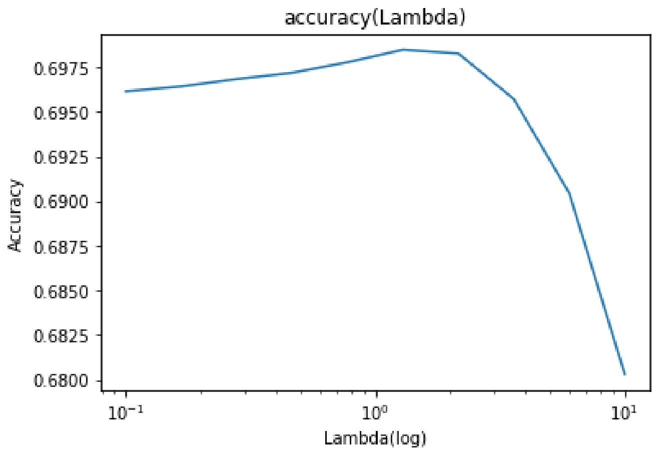

Figure 1. Accuracy depending on λ for regularized logistic regression with γ = 0.01.

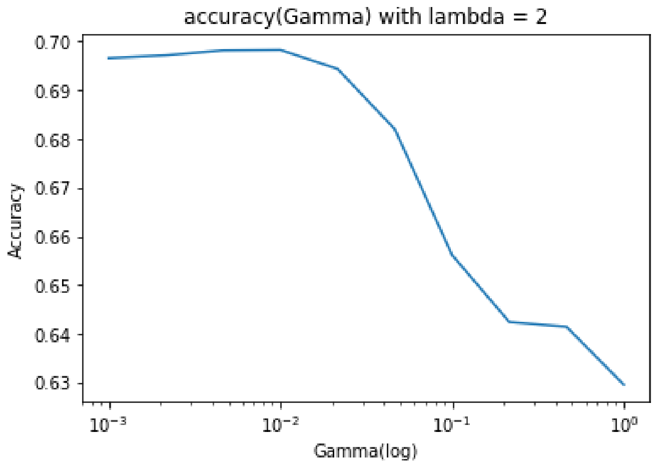

Figure 2. Accuracy depending on γ for regularized logistic regression.

To choose the best λ value we trained our models for different values on a log-scale from 0.001 to 1000 and then compared the accuracy of each resulting model. We found an optimal λ for all algorithms that took it as an input. On Figure 1 we show the results of the tuning for the Logistic Regression, as that was what we predicted would have the best testing accuracy. In the example, we can see the accuracy evolution depending on λ for the regularized logistic regression, and that the best regularization seems to be around $\lambda = 1.5$ (we only show a domain of 0.1 through 10 for visual clarity on the figure). Similarly, the learning rate γ is shown on the Figure 2. If the learning rate is too high, the gradient descent step size is large and the algorithm and may never reach any local minimum of the loss function. If the learning rate is too small, the training requires lot of iterations to reaching the minimum of the loss function. In the example plot for logistic regression shown in Figure 2, the optimal value is $\gamma = 0.01$.

Of the 6 algorithms with tuned parameters, the highest testing accuracy value we obtained was approximately 73.4% using least-squares classification.

It is surprising that the least-squares algorithm performed better than the logistic regression. One anomaly that we noticed before training the data was that even after processing and standardizing the data, some values (<1%) were still several times greater than 1, indicating that perhaps there were some data outliers that our preprocessing methods did not catch. While 73.4% is certainly significantly better than the baseline ”coin-flip” 50% accuracy for a random binary classifier, there is still significant room for improvement. Given library constraints of the project, the area where there is most room for improvement of our model is feature engineering and selection. Due to time constraints, our feature selection process of dropping features with outliers was quite naive. To improve performance, the most intuitive next step would be more advanced feature selection methods such as statistical comparison (e.g., forward selection) models described in [3]. Particular attention could be paid to the 13 raw features to probe whether our assumption that the physicist-provided derived variables added invaluable domain-specific information may be invalid. Additionally, using ensemble methods such as basic neural networks or simple voting schemes between the algorithms we implemented in this project may produce higher accurracies (though at the expense of additional computation).

IV. Conclusion

For a classification problem as complex as the Higgs Boson problem, algorithmically rigorous methods such as support vector machines and ensemble methods are commonly used. Even without tested libraries, however, we obtained a model with significant improvement over the baseline classifier with numerous possibilities for accuracy improvement readily available. Understanding and removing observed anomalies as well as slightly more advanced feature selection could result in an even more robust classifier.

References

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, “Higgs challenge,” *Laboratoire de l'Accélérateur Linéaire*, 2014.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R.* Springer, 2013.

[3] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, 2003. [Online]. Available: http://jmlr.csail.mit.edu/papers/volume3/guyon03a/guyon03a.pdf