Machine Learning - Project 1 Report

Kirill IVANOV, Matthias RAMIREZ, Nicolas TALABOT
EPFL, Switzerland

I. Introduction

The EPFL Machine Learning course CS-433 includes two projects, which encourage to apply methods and approaches that were taught during the classes to solve some machine learning tasks.

In project 1, a classification problem has to be solved. That is, given a dataset of features representing the decay signature of a collision event in CERN, one should predict whether this event was a Higgs boson or not. In order to do this, techniques from the first half of the course are used (such as linear, ridge and logistic regression).

II. Methods

A. Data exploration and feature processing

Before applying any classification algorithm, one should take a look to the dataset and check if anything useful can be done with the features. In the given datasets, there are 30 of them.

One of the simplest things to do with the data is to check the feature types and look for missing values. According to the project description [1], there is a categorical feature `PRI_jet_num` with values in the set $\{0, 1, 2, 3\}$. Those values causes the missing values in 9 other dimensions. All other features are real-valued. In addition, `DER_prodeta_jet_jet` can also contain missing values, but there is no strict dependency between this and other features.

The problems of integer features, which should not be treated as real valued, and the missing values can be solved with the dummy variable approach. One can add an extra boolean column, corresponding to missing values in the first feature of the dataset, as well as replace the integer feature by three boolean columns, corresponding to different non-zero values of this feature. All remaining missing values are defined now by these three columns, so no information is lost.

Another useful approach of dealing with features is standard scaling (i.e., zero-centering and making the standard deviation equal to 1). The missing values can be set to 0 (mean value in the scaled dataset), so that they will not impact the model.

The principal component analysis (PCA) can be also performed on the data in order to project it to an orthogonal basis and exclude potential linear dependencies, but the amount of training samples available is enough for the model to handle such dependencies on its own.

Finally, to give the model an ability to learn even more complex dependencies on features, it could be useful to create additional ones, e.g. polynomial: for all real-valued features x_{*i}, include $x_{*i}^2, x_{*i}^3, \dots$ up to some degree d. The optimal value of d can be determined via grid search.

B. Model verification

One of the most popular schemes to keep track of model generalization error and make sure it is not under- or overfitting is the K-fold cross-validation. We can use the mean and standard deviation of the loss (accuracy in our case) to do so. Moreover, it can be combined with grid-search to look for optimal parameters. Finally, for this specific project, even if we don't have the ground truth for the test set labels, submitting on Kaggle allows use to see how our algorithm performs on unseen data. Thus, it gives us an additional verification of the model generalization ability.

III. Results

To implement the methods and approaches described in the previous chapter, we used Python 3.6.2 with NumPy 1.13.3. In order to check the quality of each model, 5-fold cross-validation is used. Since the Kaggle competition uses accuracy as the score, it is reasonable to consider the accuracy as a criterion during the model selection (even though the loss functions to minimize are different). Standard scaling was applied on both training and testing datasets, using the means and standard deviations of the training set. Missing values are set to the means (zero after scaling).

The simplest model to start from is the ordinary least squares (OLS), that is using the normal equations (as a closed-form solution exists). The results for this model are presented in table I. This can be considered as a baseline for future analysis. Furthermore, the train and test accuracy are within the standard deviation from each other, which means that the model performs quite good and the generalization error is small. To this point, the model is really simple so there is little risk to overfit. Therefore, one does not need to employ the regularization techniques.

Table I
OLS: Accuracy on the Training and Testing sets (created after splitting the original training set) using only Least-Squares. Mean and standard deviation were calculated over 5 distinct learnings.

	Accuracy $\pm$ std
Training	0.74465 $\pm$ 0.00040
Testing	0.74408 $\pm$ 0.00047

The next possible step is adding dummy feature columns, as described in section III. As shows table II, this gives a small improvement in both training and testing datasets. However, it is far from being enough.

Table II
OLS + BINARY FEATURES: ACCURACY ON THE TRAINING AND TESTING SETS (CREATED AFTER SPLITTING THE ORIGINAL TRAINING SET) USING LEAST-SQUARES AND BINARY FEATURES. MEAN AND STANDARD DEVIATION WERE CALCULATED OVER 5 DISTINCT LEARNINGS.

	Accuracy $\pm$ std
Training	0.74720 $\pm$ 0.00043
Testing	0.74641 $\pm$ 0.00099

The final step in our feature engineering is adding the polynomial features. To achieve this we simply added higher degrees of each features separately, as described in . Thus, we do not consider interactions between features in this model. In order to select the optimal degree, a grid search was performed. Figure 1 illustrates how the model accuracy on training and testing sets changes with degree d.

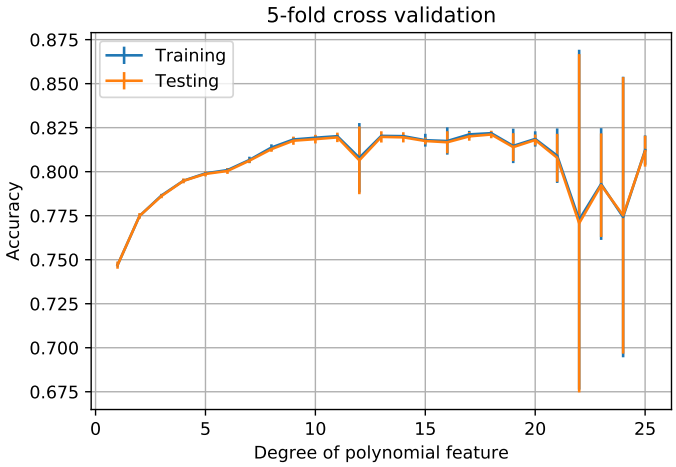

Figure 1. Choosing the degree for the polynomial features. The accuracy on both sets (after splitting the data into train/validation sets) is plotted versus the degree of the polynomial features.

It can be seen that there is a significant improvement in accuracy when the degree is increased, up to $d = 11$, followed by a saturation. Further increase of degree does not get better accuracy, but rather increases the variance of models. This can be considered as the sign of too high complexity of the model. The possible reason why the difference between training and testing accuracy is extremely small is that the training dataset is large enough and covers the feature space quite well, hence the training and testing datasets during cross-validation have equivalent accuracies. To prevent the model from being too complex, one can choose $d = 11$ as the target value for the model.

The final model consists of polynomial features with $d = 11$, as well as the binary dummy feature columns. It needs to be said that the binary features were not used in the polynomial features construction, as well as in the standardization (in order to stay binary, and not real valued). The estimation of the model quality is presented in table III. One can notice the significant improvement in comparison to the previous cases.

Table III
OLS + BINARY & POLYNOMIAL FEATURES: ACCURACY ON THE TRAINING AND TESTING SETS (CREATED AFTER SPLITTING THE ORIGINAL TRAINING SET) USING LEAST-SQUARES, BINARY AND POLYNOMIAL FEATURES. MEAN AND STANDARD DEVIATION WERE CALCULATED OVER 5 DISTINCT LEARNINGS.

	Accuracy $\pm$ std
Training	0.81901 $\pm$ 0.00318
Testing	0.81708 $\pm$ 0.00338

IV. FUTURE DIRECTIONS

There are some possible data analysis directions which were not covered in the following report. One can take into consideration feature distributions. Moreover, it could be also possible to make use of the difference in distribution of features with respect to the labels.

V. CONCLUSION

Within a framework of the first project in course CS-433, a classification problem of finding the Higgs boson has been studied. Several data preprocessing techniques were used with a least squares method, which has allowed us to achieve an accuracy of ≈ 0.82 on the unseen test set. However, with more time and knowledge, no doubt that a higher accuracy could be achieved.

REFERENCES

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, "Learning to discover: the higgs boson machine learning challenge - documentation," *CERN Open Data Portal*, 2014.