Highly Imbalanced Learning

Paul Viallard
paul.viallard@etu.univ-st-etienne.fr

Omar Elsabrout
omar.elsabrout@etu.univ-st-etienne.fr

30 November 2018

Abstract

Binary classification with strong class imbalance can be found in many real-world classification problems. From trying to predict events such as network intrusion and bank fraud to a patient's medical diagnosis, the goal in these cases is to be able to identify instances of the minority class that is underrepresented in the dataset. This, of course, presents a big challenge as most predictive models tend to ignore the more critical minority class while deceptively giving high accuracy results by favoring the majority class.

1 Introduction

In this lab session we explore the problem of having a severe imbalance in our provided datasets. Such imbalance makes it hard to converge into a model that has strict decision boundaries with wide margins to successfully classify new data points at test time by conventional techniques.

Hence, we decided to follow three algorithms and their combinations to tackle this problem of imbalance in the given dataset. Those algorithms are Gradient Boosting, SMOTE and Tomek Links. Each one is discussed in its own section.

2 Dataset

In order to understand and experience handling such problem, we observe a fraud detection dataset with 10,000 data examples. The imbalance in such dataset is having almost all of 9,787 labels being negative and only 213 being positive. We receive the dataset in form of two CSV files. Thus, we process those two files, `train.csv` and `train_labels.csv`, using Pandas[1] package in Python. The dataset has 53 features of numerical integers and floats that contribute into a binary classification model. In the `train_labels.csv` file we have either 0 for negative labels or 1 for positive ones.

3 Gradient Boosting

Boosting algorithms play a crucial role in dealing with bias variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias and variance), and is considered to be more effective. A sincere understanding of Gradient Boosting is our main target out of applying it on our dataset. We use the algorithm provided by Scikit-learn[2] package in Python.

Explaining how Gradient Boosting works is not our main issue in this document. On the other hand, we seek to understand the parameters of Gradient Boosting while doing the exercise and we can safely dictate that the overall parameters can be divided into three categories:

- Tree-specific parameters.
- Boosting parameters.
- Miscellaneous parameters.

We experimented with all types of parameters in order to optimize our model and adapt it to the dataset. However, we reached our maximum F1 score while tuning these parameters:

- `min_samples_split`: It defines the minimum number of samples (or observations) which are required in a node to be considered for splitting. It is also used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
- `min_samples_leaf`: Generally, lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.
- `max_depth`: It is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- `max_features`: It is the number of features to consider while searching for a best split. As a thumb-rule, square root of the total number of features works great but we checked up to 30% of the total number of features.
- `learning_rate`: Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well.

We finally executed this model to train on the provided training dataset with K-fold cross validation to obtain an F1 score, a recall and a precision. Results are discussed in section 5.

4 SMOTE and Tomek Links

As the exercise requires, we choose two strategies to improve the Gradient Boosting classifier we implemented and then compare their effects. We choose SMOTE as a method of oversampling positive examples before training our classifier. In addition, we also choose Tomek Links as our second methods to do undersampling on the negative data points. We decided not to modify parameters of both as their auto mode is optimized for our data. On the other hand, we decided to fix the random seed for both to have consistent results.

5 Results

In this section, we present our findings of applying the previously discussed strategies on the input dataset.

Gradient Boosting:

	precision	recall	f1-score
non-default	0.27 (+/- 0.78)	0.17 (+/- 0.09)	0.84 (+/- 0.11)
default	0.21 (+/- 0.78)	0.09 (+/- 0.09)	0.32 (+/- 0.11)
average	0.24 (+/- 0.78)	0.13 (+/- 0.09)	0.57 (+/- 0.11)

Gradient Boosting with SMOTE:

	precision	recall	f1-score
non-default	0.29 (+/- 0.2)	0.23 (+/- 0.11)	0.89 (+/- 0.09)
default	0.23 (+/- 0.2)	0.12 (+/- 0.11)	0.33 (+/- 0.09)
average	0.26 (+/- 0.2)	0.18 (+/- 0.11)	0.61 (+/- 0.09)

Gradient Boosting With Tomek Links:

	precision	recall	f1-score
non-default	0.25 (+/- 0.19)	0.28 (+/- 0.11)	0.85 (+/- 0.08)
default	0.22 (+/- 0.19)	0.09 (+/- 0.11)	0.31 (+/- 0.08)
average	0.23 (+/- 0.19)	0.19 (+/- 0.11)	0.6 (+/- 0.08)

6 Conclusion and Perspectives

At the end of the exercise, we learn the benefits of applying undersampling and oversampling using SMOTE and Tomek Links. The increase in the f1-score is due to increasing the diversity in the positive data points or removing negative data points that overlap with the positive ones. We also attach a CSV file that contains our label predictions of the `test.csv` file using our best classifier which is Gradient Boosting with SMOTE. We understand that there is a space for improvements in tuning the parameters and we will update them once we reach a new result.

References

[1] pandas: Python Data Analysis Library. Online, 2012. URL `http://pandas.pydata.org/`.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.