Imbalanced Datasets: The Effects of Oversampling Methods on Gradient Boosting Algorithms

Joseph Renner and Oussama Bouldjedri

December 18, 2017

1 Introduction

Imbalanced datasets present problems for many classification algorithms. If the algorithm optimizes accuracy, a model trained on a highly imbalanced dataset can predict the majority class for every sample, simply because there aren't enough of the minority class examples to leverage the model into predicting a minority class. For example, if a dataset is 99% one class, and 1% another, the model can predict the majority class for every test example and achieve 99% accuracy. There are many ways to overcome this problem. In this report, three strategies, SMOTE plus gradient boosting, oversampling plus gradient boosting, and just gradient boosting will be implemented and compared using an imbalanced fraud detection dataset.

2 SMOTE vs Random Oversampling

SMOTE stands for Synthetic Minority Over-Sampling Technique. As seen by it's title, it is an oversampling technique, meaning it adds minority class examples to the dataset in order to make the dataset more balanced. As opposed to random oversampling, which simply duplicates random examples in the minority class until the the dataset is sufficiently balanced, SMOTE generates new synthetic minority class examples. This is achieved by, for each minority example, find its k-nearest minority class neighbors, randomly select j of these neighbors, then generate minority class examples that lie along the line between the original example and it's j counterpoints. This results in generated minority class samples that are not exactly the same as the original minority class samples, adding diversity to the minority class when compared to generic oversampling. However, cons of SMOTE include making the learning process longer, as the SMOTE process needs to calculate k-nearest neighbors for its minority class, and it needs calculate the lines between each sample and j of its neighbors.

3 Boosting

Boosting refers to combining many weak (high-bias, low variance) models. In this experiment, AdaBoost (short for adaptive boosting) will be used. In AdaBoost, weak learners are fit to repeatedly modified versions of the original dataset. The weak learners are combined using a weighted majority vote to produce a final prediction. At each iteration of the algorithm, weights are applied to each example in the training set. At the start, the weights for all examples are set to $1/N$. At each subsequent iteration, the weights for misclassified examples are increased and the correctly classified examples have their weights decreased. The effect of this is that examples that are harder to classify are given larger weights so that the classifiers put more emphasis on correctly predicting these examples.

In an imbalanced dataset situation, this is a nice feature because the minority class examples will have higher weights because they are harder to classify, resulting in a more robust model.

4 Experimental Setup

SMOTE and oversampling will be applied to an imbalanced dataset, then AdaBoost will be used to classify the resulting datasets. To gauge the effect of SMOTE and random oversampling independently, an AdaBoost model will be fit to the original imbalanced dataset as well. Each approach will be ran five times, and the average accuracies, precisions, and recalls of the three approaches on a validation set will be compared. The sampling will be done using a Python library called imlearn, and the AdaBoost implementation used will be from sklearn.

4.1 Dataset

The dataset that will be used in the experiment is the small (70mb) dataset of fraud detections. In this dataset, there are 366667 examples, of which 836 (0.2%) are positive. From this, a validation set of 10000 total examples is sampled.

4.2 Hyperparameters

Since there are over 300,000 majority class examples, if we oversample to the point where the minority positive class has the same number of examples as the majority negative class, the dataset will become too large for my old (ancient) laptop to run AdaBoost on. Thus, we will undersample the majority class to have 100,000 examples, and use the two oversampling techniques to increase the number of minority samples to 100,000. We will also try the same experiments with the number of examples of each class set to 10,000, to see the effect of having less oversampling and possibly more diversity among positive examples. SMOTE will be used with the number of nearest neighbors ($k = j$ in this case)

Table 1: Results of the Techniques

Approach	# of Negative Examples	# of Positive Examples	Avg Accuracy	Avg Precision	Avg Recall
Adaboost	355856	811	**0.9976**	0.0	0.0
Oversampling + Adaboost	100000	100000	0.8189	0.0109	0.7231
SMOTE + Adaboost	100000	100000	0.8272	0.0092	0.7619
Oversampling + Adaboost	10000	10000	0.8123	0.0085	0.7273
SMOTE + Adaboost	10000	10000	0.8123	**0.0116**	**0.7857**

equal to 5. AdaBoost will be used with the number of weak learners equal to 50.

5 Results

The results are presented in Table 1. As expected, running Adaboost without dataset modifications produces the highest accuracy on the validation set, since it predicts only negative. Thus, the precision and recall are equal to 0. The accuracies of Adaboost with sampling techniques are lower, but since they actually predict positive examples, the recalls are relatively high. However, the precisions are all around 1%, meaning they are only correct in their positive predictions 1% of the time.

5.1 Analysis

On average, the SMOTE technique with Adaboost performs better than random oversampling with Adaboost. While the accuracy and precision of random oversampling is better than that of SMOTE with 100,000 samples for each class, the recall is worse than SMOTE. In the case of fraud detection, this implies the SMOTE technique is more effective, since false positives are much less expensive than false negatives. Furthermore, when the number of samples per class is dropped to 10,000, SMOTE with Adaboost produces the best precision and recall out of all techniques and dataset sizes. This is because with less examples, there is more diversity; and also, with less examples, there is less probability of generating positive examples that are not representative of the class.

Without any resampling at all, the resulting Adaboost model is useless, as it always predicts negative for every datapoint.

6 Conclusion

With a highly imbalanced dataset, using resampling techniques can be crucial to making your model work. As seen in the above experiments, using gradient boosting without any dataset modifications can result in a useless model. Random oversampling and SMOTE can increase the usefulness of the resulting models, especially in a situation where false negatives can be very expensive, such as fraud detection. SMOTE is seen to be a superior method to random oversampling when paired with gradient boosting in this case, as it produces models which have a better recall.