Imbalanced Data for Fraud Detection

Chieh Hsu
A53322300
chh008@ucsd.edu

Lu Lu
A53307505
l1lu@ucsd.edu

Abstract

Learning imbalanced data is still a challenge that attracts the attention of academia and industries in the past decade. Even though there are a lot of existing methods in the field, there are still many problems remaining to solve. In this project, two aspects of solutions for imbalanced data classification are investigated. Both aspects are implemented with several different methods for comparison purposes. The methods, experimental settings, and procedures that led to the results are discussed. Furthermore, we propose a new method for improving the performance of the existing methods.

1. Introduction

What is imbalanced data? Imbalanced data is that the distribution of observations in each class is imbalanced. That means your might have a large number of observations for one class, and much fewer observations in another class. For example, in the dataset of detecting unauthorized credit card transaction, it is likely to have 1 unauthorized transaction within 1000 authorized transactions.

In certain areas such as fraud detection, medical diagnosis and rick management, severe imbalance class distribution is usually a common problem, confronting with severely imbalanced data where minority class accounting only for 2%-5%, or even less. Normally, standard algorithms expect data to have balance class distributions or equal misclassification costs. Therefore, in imbalanced data classification, the standard classifier tends to classify the minority as the majority. For instance, the classifier might misjudge the unauthorized transactions as authorized transactions. Then there would be no alert for fraudulent usage of credit card. However, the minority is usually what we are concerned more about, such as the unauthorized transactions.

There are several existing approaches to handling with the imbalanced data classification such as resampling and weight adjustment. In this project, the area we focus on is fraud detection. We implement several methods mentioned above and compared the performance of them. Moreover, we propose a new solution that combines LDA and SMOTE.

2. Description of Methods

The existing methods for addressing imbalanced data classification can be divided into data-level and algorithm-level. On the data-level, oversampling the examples of minority class or undersampling the examples of majority class is able to balance the class ratio. On the algorithm-level, costs of misclassification are adjusted to balance the errors of different classes. We implement several existing methods and proposed a new data-level method as follows.

2.1. Data-level

Since GE Batista et al. [1] concluded that oversampling methods are able to aid in the induction of classifiers that are more accurate than those induced from undersampled datasets, we mainly focus on implementing oversampling methods only and improving them.

2.1.1 Random oversampling (ROS)

Random oversampling randomly selects examples of minority class, and adds their copies into the original dataset for several times. The new dataset then has the balanced class ratio. Even though ROS is very simple, Batista et al. [1] showed that it is competitive to other complex oversampling methods.

2.1.2 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [2] works by selecting a random example from the minority class. Then k nearest neighbors for that example are found. The random selected neighbor is chosen and a synthetic point is generated at the selected point between the original chosen example and its selected neighbor shown in Figure 1. It was the state-of-the-art method for handling imbalanced data classification at that time. Many improvements on SMOTE were then developed, such as

Borderline SMOTE [3], K-means SMOTE [4] and SMOTE + Tomek links [5].

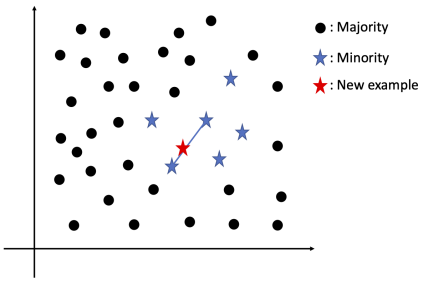


Figure 1. Data generation based on euclidean distance

2.1.3 Borderline SMOTE

SMOTE generates the same number of synthetic examples of each original minority examples without considering their neighboring examples. In regards of k nearest examples for a minority example, if n of them are in majority, where $\frac{k}{2} \leq n < k$, this example is eaily misclassified and close the borderline. Rather than generating synthetic instances for each minority examples, Borderline SMOTE only generates synthetic instances for minority examples closer to the borderline.

2.1.4 LDA+SMOTE

Within-class imbalance is another type for imbalanced data. The examples of the same class may scatter around in the dataspace, which has been shown to greatly depreciate classification performance [6][7]. In addition, within-class imbalance may also result in generating wrong synthetic examples. For example, in Figure 2, synthetic instances for example A should be in majority. In order to address this issue, we first utilized linear discriminant analysis (LDA) to project data into a subspace that better separates two classes. Then we concatenate projections with the original features, and use SMOTE to create a new dataset.

2.2. Algorithm Level

2.2.1 Cost-sensitive network

Neural network assumes that all classification errors have equal costs. However, the assumption often fails in practice. For example, it is much more dangerous to predict a patient with cancer as healthy than to diagnose a healthy person with cancer. Kukar et al. [8] proposed cost-sensitive learning with neural network to solve this kind of problem. It is able to address imbalanced data as well. Since there are much fewer examples of minority class, so the error of misclassified examples of majority class will dominate the empirical loss. By increasing the weights of error made by

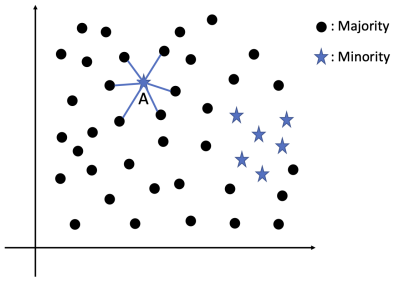


Figure 2. Within-class imbalance

misclassified minority examples, cost-sensitive network is able to balance the error of different classes.

2.2.2 Focal Loss

While cost-sensitive network balances the importance of positive and negative examples, it does not differentiate easy or hard examples. Focal loss [9] instead reshapes the loss function to down-weight easy examples and thus focus training on hard examples. In practice, they combined cost-sensitive weights with focusing parameter and got the best performance. (1) shows the focal loss, where α_t is the cost-sensitive weight, γ is the focusing parameter and p_t is the estimated probability.

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}log(p_t) \tag{1}$$

3. Experimental Setting

In this section, the dataset, environment, and parameters of the training will be discussed. Figure 3 shows the flow chart of the model training. After data preprocessing, data-level or algorithm-level approaches are implemented separately. The data resampling part are removed when evaluating the model.

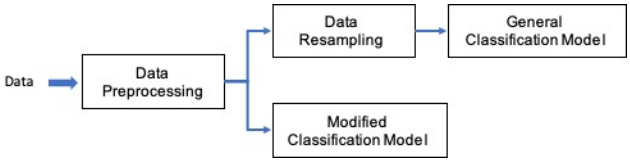


Figure 3. Flow chart of model training

3.1. Dataset

All methods were comprehensively evaluated on the dataset, *TalkingData AdTracking Fraud Detection Challenge*, from Kaggle. Based on the user information, we aim to know if the user installs the app. The user information contains 'ip', 'app', 'device', 'os', 'channel', 'click_id', 'click_time', 'is_attributed', where 'is_attributed' is the label. There are $456,846$ positive examples within

Layer	Output
Linear	$N \times 10$
ReLU	$N \times 10$
Linear	$N \times 10$
ReLU	$N \times 10$
Linear	$N \times 1$
Sigmoid	$N \times 1$

Table 1. Neural network architecture, where N =number of samples

$184,903,490$ labeled data. Since it spends too much time training the model with the complete dataset, we reduced it to the size of $10,000,000$, while keeping the same class ratio. Next, we split the data into three portions, training set: 81%, validation set: 9%, testing set: 10%. The proportion of negative and positive examples is 99.8:0.2. Hence, the dataset is an imbalanced dataset.

3.2. Data preprocessing

The 'ip' and 'channel' are combined to a new feature, 'clicks_by_ip', which represents the number of clicks by certain 'ip'. Instead of using categorical 'ip', 'clicks_by_ip' is more informative and suitable for training the neural network. For the same reason, 'day_of_week' and 'day_of_year' are extracted from 'click_time' [10]. Hence, there are 7 features for each example.

3.3. Linear discriminant analysis (LDA)

For the purpose of generating useful minority examples when implementing SMOTE, LDA is performed to create a new dimension that separates two classes. The linear discriminant is calculated from the training data. After concatenating the projection from LDA to the original features, new data includes 8 features.

3.4. Data resampling

ROS and SMOTE were performed to generate minority examples so that there were equal number of examples for both classes. We tried to augment the dataset by using Borderline SMOTE, but the process was not finished over eight hours. Therefore, we decided not to implement Borderline SMOTE. The reason may be that the neighboring examples for each minority example have to be identified, so the running time increases rapidly with large dataset. Data resampling was removed when evaluating the model.

3.5. General classification model

We used the fully-connected neural network as our general classification model. Table 1 shows its network architecture. Additionally, we choose cross-entropy loss as objective function and Adam [11] as the optimizer. Since we want to evaluate the effect of methods for imbalanced data, we keep the classification model simpler.

Weight $[w_1, w_2]$	γ
$[m \times 100, (1-m) \times 100]$	1
$[m, 1-m]$	2
$[1, \frac{1}{m}]$	3
	4

Table 2. Parameter of cost-sensitive network and focal loss, where m =ratio of minority class, γ = focusing parameter

3.6. Modified classification model

We use the same neural network for modified classification model, but adjust the loss function based on the algorithm-level approaches. 2 shows the cross-entropy loss with cost-sensitive weights $[w_1, w_2]$ and focusing parameter γ. γ is equal to 0 for the cost-sensitive network. Different weights and focusing parameter as shown in Table 2 are tested to find the models with highest validation performance. Three kinds of weights amplify the cost of misclassified minority examples with different degrees.

$$L = (1-x_i)^{\gamma}\{-w_1[y_i log(x_i)] - w_2[(1-y_i)log(1-x_i)]\} \quad (2)$$

3.7. Parameter tuning

Since the overall performance are considered, the model with highest sum of recall, precision and AUC in the validation set is used to classify the testing set. For the cost-sensitive network, model with weight of $[,]$ is the best. For the focal loss, the model with $\gamma = 1$ and weight$= [1, \frac{1}{m}]$ outperform others.

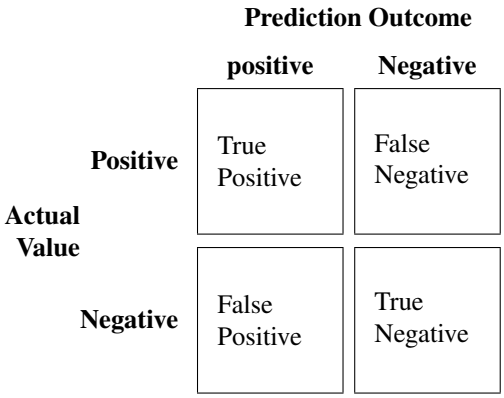
4. Result

4.1. Performance metrics

We usually use accuracy to evaluate the performance of the model. However, in imbalanced data classification, accuracy is not a proper metric to evaluate the model. The accuracy of the model is basically the total number of correct predictions divided by total number of predictions. Because there is a large portion of the majority examples in both the number of correct prediction and the total number of prediction, even though we are not able to classify minority correctly, we may still get very high accuracy in evaluation. Therefore, recall, precision, ROC curve and AUC are used for the evaluation.

	NN	ROS	SMOTE	Cost-sensitive	Focal loss	LDA+SMOTE
Recall	0.080	**0.770**	0.758	0.080	0.409	0.765
Precision	**0.654**	0.019	0.021	0.607	0.624	0.020
AUC	0.816	**0.907**	0.889	0.858	0.886	0.897

Table 3. Testing performance of each method



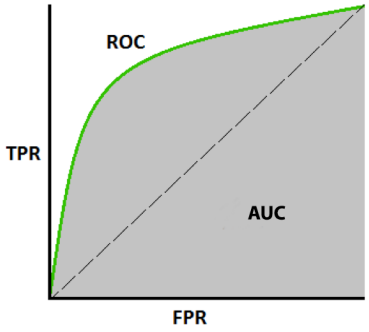


Figure 4. ROC curve and AUC.

4.1.1 Recall

$$Recall/TPR = \frac{TP}{TP + FN}$$

The recall is the total number of True Positive(TP) divided by total number of True Positive(TP) and False Negative(FN). Therefore, it is also called TPR(True Positive Rate). The recall of a class expresses how well the model is able to detect that class.

4.1.2 Precision

$$Precision = \frac{TP}{TP + FP}$$

The precision is the total number of True Positive(TP) divided by total number of True Positive(TP) and False Positive(FP). The precision of a class define how trustable the result is when the model answers that a point belongs to that class.

4.1.3 ROC curve and AUC

ROC is a probability curve that plotted with true positive rate(TPR) against the false positive rate(FPR) where true positive rate is on y-axis and false positive is on the x-axis shown in Figure4. It summarizes the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

AUC is based on the ROC curve that acts a little bit as a scalar value that summarises the entire ROC curve. AUC is the area under the ROC curve, representing degree or measure of separability. For the best case, the AUC tends toward 1.0. For example, the higher the AUC, the better the model is at distinguishing between patients with disease and no disease. For the worst cast, AUC tends toward 0.5. A high AUC score means that the model does not sacrifice the precision to get a good recall.

4.2. Performance of methods

Table 3 shows the testing performance of each method, where NN represents the general classification model without data resampling. Data-level methods achieve the similar performance and outperform algorithm-level methods in aspect of the recall. However, data-level methods works poorly in precision of validation and testing. The reason may be that the class ratio is balanced during training, but during evaluation, the class ratio is imbalanced.

Figure 5 shows the training and validation performance of each method. Data-level methods are more stable than algorithm-level methods. Since there are more data in one epoch, the variance of the gradient is smaller resulting in stable training.

5. Conclusion

In this project, we identify that each existing method has some drawbacks to the dataset we choose. To address this, we propose LDA+SMOTE that tries to project data into a subspace that separates two classes better. Then use SMOTE to create a new dataset. Compared with other improvements on SMOTE, our approach is simple and time saving. However, the results show that the performance of LDA+SMOTE we proposed is similar to ROS and SMOTE.

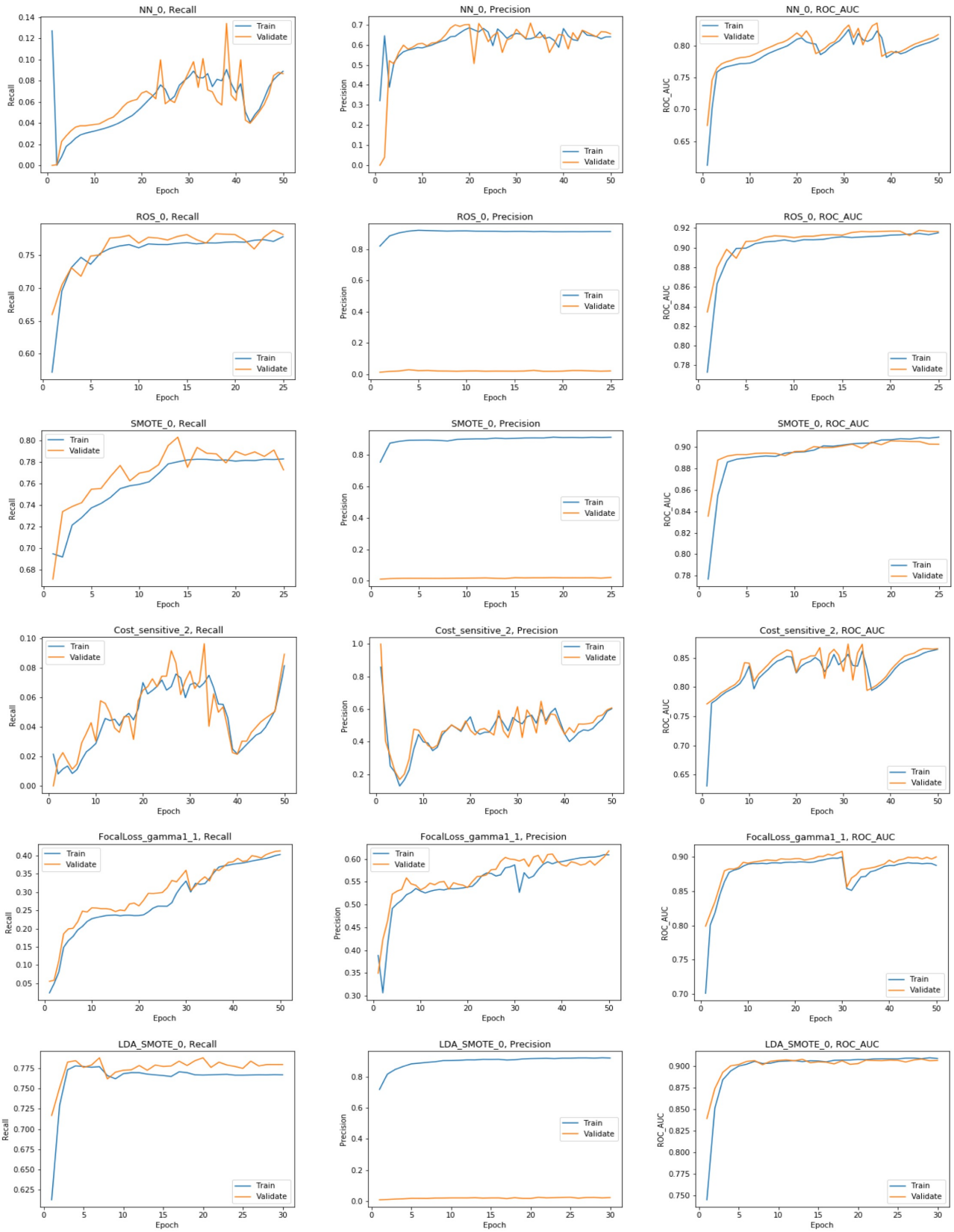


Figure 5. Training and validation performance of each method

There might be some aspects that we neglect so that the result of LDA+SMOTE does not match the expectation. We will keep working on this field.

References

[1] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[4] Felix Last, Georgios Douzas, and Fernando Bacao. Oversampling for imbalanced learning based on k-means and smote. *arXiv preprint arXiv:1711.00837*, 2017.

[5] Gustavo EAPA Batista, Ana LC Bazzan, Maria Carolina Monard, et al. Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18, 2003.

[6] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM Sigkdd Explorations Newsletter*, 6(1):40–49, 2004.

[7] Ronaldo C Prati, Gustavo EAPA Batista, and Maria Carolina Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *Mexican international conference on artificial intelligence*, pages 312–321. Springer, 2004.

[8] Matjaz Kukar, Igor Kononenko, et al. Cost-sensitive learning with neural networks. In *ECAI*, volume 98, pages 445–449, 1998.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[10] João Pedro Peinado. Talkingdata xgboost - lb: 0.966.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.