Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap

Department of Mathematics, Faculty of Science, Chulalongkorn University
chumphol@chiangmai.ac.th, Krung.S@chula.ac.th,
lchidcha@chula.ac.th

Abstract. The class imbalanced problem occurs in various disciplines when one of target classes has a tiny number of instances comparing to other classes. A typical classifier normally ignores or neglects to detect a minority class due to the small number of class instances. SMOTE is one of over-sampling techniques that remedies this situation. It generates minority instances within the overlapping regions. However, SMOTE randomly synthesizes the minority instances along a line joining a minority instance and its selected nearest neighbours, ignoring nearby majority instances. Our technique called Safe-Level-SMOTE carefully samples minority instances along the same line with different weight degree, called safe level. The safe level computes by using nearest neighbour minority instances. By synthesizing the minority instances more around larger safe level, we achieve a better accuracy performance than SMOTE and Borderline-SMOTE.

Keywords: Class Imbalanced Problem, Over-sampling, SMOTE, Safe Level.

1 Introduction

A dataset is considered to be imbalanced if one of target classes has a tiny number of instances comparing to other classes. In this paper, we consider only two-class case [5], [17]. The title of a smaller class is a minority class, and that of a bigger class is a majority class. The minority class includes a few positive instances, and the majority class includes a lot of negative instances.

In many real-world domains, analysts encounter many class imbalanced problems, such as the detection of unknown and known network intrusions [8], and the detection of oil spills in satellite radar images [13]. In these domains, standard classifiers need to accurately predict a minority class, which is important and rare, but the usual classifiers seldom predict this minority class.

Strategies for dealing with the class imbalanced problem can be grouped into two categories. One is to re-sample an original dataset [11], [14], [15], either by over-sampling a minority class or under-sampling a majority class until two classes are nearly balanced. The second is to use cost sensitive learning by assigning distinct costs to correctly classified instances or classifications errors [7], [9], [16].

Table 1. A confusion matrix for a two-class imbalanced problem

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

The performance of classifiers is customarily evaluated by a confusion matrix as illustrated in Table 1. The rows of the table are the actual class label of an instance, and the columns of the table are the predicted class label of an instance. Typically, the class label of a minority class set as positive, and that of a majority class set as negative. *TP*, *FN*, *FP*, and *TN* are True Positive, False Negative, False Positive, and True Negative, respectively. From Table 1, the six performance measures on classification; *accuracy*, *precision*, *recall*, *F-value*, *TP rate*, and *FP rate*, are defined by formulae in (1)-(6).

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FN} + \text{FP} + \text{TN})\ . \tag{1}$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})\ . \tag{2}$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})\ . \tag{3}$$

$$\text{F-value} = ((1 + \beta)^2 \cdot \text{Recall} \cdot \text{Precision}) / (\beta^2 \cdot \text{Recall} + \text{Precision})\ . \tag{4}$$

$$\text{TP Rate} = \text{TP} / (\text{TP} + \text{FN})\ . \tag{5}$$

$$\text{FP Rate} = \text{FP} / (\text{TN} + \text{FP})\ . \tag{6}$$

The objective of a classifier needs to aim for high prediction performance on a minority class. Considering the definition of *accuracy*, if most instances in a minority class are misclassified and most instances in a majority class are correctly classified by a classifier, the *accuracy* is still high because the large number of negative instances influences the whole classification result on *accuracy*. Note that *precision* and *recall* are effective for this problem because they evaluate the classification rates by concentrating in a minority class. In addition, *F-value* [3] integrating *recall* and *precision*, is used instead of *recall* and *precision*. Its value is large when both *recall* and *precision* are large. The β parameter corresponding to relative importance of *precision* and *recall* is usually set to 1. Furthermore, ROC curve, The Receiver Operating Characteristic curve, is a standard technique for summarizing classifier performance over a range of tradeoffs between *TP rate*, benefits, and *FP rate*, costs. Moreover, AUC [2], Area under ROC, can also be applied to evaluate the performance of a classifier.

The content of this paper is organized as follows. Section 2 briefly describes related works for handling the class imbalanced problem. Section 3 describes the details of our over-sampling technique, Safe-Level-SMOTE. Section 4 shows the experimental results by comparing Safe-Level-SMOTE to SMOTE and Borderline-SMOTE. Section 5 summarizes the paper and points out our future works.

2 Related Works

Re-sampling is a preprocessing technique which adjusting the distribution of an imbalanced dataset until it is nearly balanced, before feeding it into any classifiers. The simplest re-sampling techniques are a random over-sampling technique [14] and a random under-sampling technique [14]. The former randomly duplicates positive instances into a minority class, while the latter randomly removes negative instances from a majority class. Both techniques are sampling the dataset until the classes are approximately equally represented. However, the random over-sampling technique may cause the overfitting problem [19] because the technique may create the decision regions smaller and more specific. The random under-sampling technique encounters the problem that diminishing some important information of a dataset. For handling these problems, improved re-sampling techniques were studied and are described as follows.

Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W. (2002) designed the State of the Art over-sampling technique, namely SMOTE, Synthetic Minority Over-sampling TEchnique [4]. It over-samples a minority class by taking each positive instance and generating synthetic instances along a line segments joining their *k* nearest neighbours in the minority class. This causes the selection of a random instance along the line segment between two specific features. The synthetic instances cause a classifier to create larger and less specific decision regions, rather than smaller and more specific regions. More general regions are now learned for positive instances, rather than those being subsumed by negative instances around them. The effect is that decision trees generalize better. However, SMOTE encounters the overgeneralization problem. It blindly generalizes the region of a minority class without considering a majority class. This strategy is particularly problematic in the case of highly skewed class distributions since, in such cases, a minority class is very sparse with respect to a majority class, thus resulting in a greater chance of class mixture.

Han, H., Wang, W., Mao, B. (2005) designed the improvement of SMOTE, namely Borderline-SMOTE [10]. The authors divided positive instances into three regions; noise, borderline, and safe, by considering the number of negative instances on *k* nearest neighbours. Let *n* be the number of negative instances among the *k* nearest neighbours. The three regions are defined by the definitions in Table 2. Borderline-SMOTE uses the same over-sampling technique as SMOTE but it over-samples only the borderline instances of a minority class instead of over-sampling all instances of the class like SMOTE does. Unfortunately, considering two positive instances those *n* values are equal to *k* and *k*-1 for the first and second instances consecutively. These instances are not obviously difference but they are divided into the different regions; noise and borderline. The first instance is declined but the second instance is selected for over-sampling.

Table 2. The definitions of noise, borderline, and safe regions in Borderline-SMOTE

Region	Definition
Noise	$n = k$
Borderline	$½k \le n < k$
Safe	$0 \le n < ½k$

3 Safe-Level-SMOTE

Based on SMOTE, Safe-Level-SMOTE, Safe-Level-Synthetic Minority Over-sampling TEchnique, assigns each positive instance its *safe level* before generating synthetic instances. Each synthetic instance is positioned closer to the largest *safe level* so all synthetic instances are generated only in safe regions.

The *safe level* (*sl*) is defined as formula (7). If the *safe level* of an instance is close to 0, the instance is nearly noise. If it is close to k, the instance is considered safe. The *safe level ratio* is defined as formula (8). It is used for selecting the safe positions to generate synthetic instances.

$$\text{safe level } (sl) = \text{the number of a positive stances in } k \text{ nearest neighbours} . \tag{7}$$

$$\text{safe level ratio} = sl \text{ of a positive instance} \,/\, sl \text{ of a nearest neighbours} . \tag{8}$$

Safe-Level-SMOTE algorithm is showed in Fig. 1. All variables in this algorithm are described as follows. p is an instance in the set of all original positive instances D. n is a selected nearest neighbours of p. s included in the set of all synthetic positive instances D' is a synthetic instance. sl_p and sl_n are *safe level* of p and *safe level* of n respectively. *sl_ratio* is *safe level ratio*. *numattrs* is the number of attributes. *dif* is the difference between the values of n and p at the same attribute id. *gap* is a random fraction of *dif*. $p[i]$, $n[i]$, and $s[i]$ are the numeric values of the instances at i^{th} attribute. p, n, and s are vectors. sl_p, sl_n, *sl_ratio*, *numattrs*, *dif*, and *gap* are scalars.

After assigning the *safe level* to p and the *safe level* to n, the algorithm calculates the *safe level ratio*. There are five cases corresponding to the value of *safe level ratio* showed in the lines 12 to 28 of Fig. 1.

The first case showed in the lines 12 to 14 of Fig. 1. The *safe level ratio* is equal to ∞ and the *safe level* of p is equal to 0. It means that both p and n are noises. If this case occurs, synthetic instance will not be generated because the algorithm does not want to emphasize the important of noise regions.

The second case showed in the lines 17 to 19 of Fig. 1. The *safe level ratio* is equal to ∞ and the *safe level* of p is not equal to 0. It means that n is noise. If this case occurs, a synthetic instance will be generated far from noise instance n by duplicating p because the algorithm want to avoid the noise instance n.

The third case showed in the lines 20 to 22 of Fig. 1. The *safe level ratio* is equal to 1. It means that the *safe level* of p and n are the same. If this case occurs, a synthetic instance will be generated along the line between p and n because p is as safe as n.

The fourth case showed in the lines 23 to 25 of Fig. 1. The *safe level ratio* is greater than 1. It means that the *safe level* of p is greater than that of n. If this case occurs, a synthetic instance is positioned closer to p because p is safer than n. The synthetic instance will be generated in the range [0, 1 / *safe level ratio*].

The fifth case showed in the lines 26 to 28 of Fig. 1. The *safe level ratio* is less than 1. It means that the *safe level* of p is less than that of n. If this case occurs, a synthetic instance is positioned closer to n because n is safer than p. The synthetic instance will be generated in the range [1 - *safe level ratio*, 1].

After each iteration of *for* loop in line 2 finishes, if the first case does not occurs, a synthetic instance *s* will be generated along the specific-ranged line between *p* and *n*, and then *s* will be added to *D'*.

After the algorithm terminates, it returns a set of all synthetic instances *D'*. The algorithm generates |*D*| - *t* synthetic instances where |*D*| is the number of all positive instances in *D*, and *t* is the number of instances that satisfy the first case.

```
Algorithm: Safe-Level-SMOTE
Input: a set of all original positive instances D
Output: a set of all synthetic positive instances D'
 1. D' = $\varnothing$
 2. for each positive instance p in D {
 3.    compute k nearest neighbours for p in D and
       randomly select one from the k nearest neighbours, call it n
 4.    $sl_p$ = the number of positive stances in k nearest neighbours for p in D
 5.    $sl_n$ = the number of positive stances in k nearest neighbours for n in D
 6.    if ($sl_n \neq 0$) {  ; sl is safe level.
 7.       sl_ratio = $sl_p$ / $sl_n$  ; sl_ratio is safe level ratio.
 8.    }
 9.    else {
10.       sl_ratio = $\infty$
11.    }
12.    if (sl_ratio = $\infty$ AND $sl_p = 0$) {  ; the 1st case
13.       does not generate positive synthetic instance
14.    }
15.    else {
16.       for (atti = 1 to numattrs) {  ; numattrs is the number of attributes.
17.          if (sl_ratio = $\infty$ AND $sl_p \neq 0$) {  ; the 2nd case
18.             gap = 0
19.          }
20.          else if (sl_ratio = 1) {  ; the 3rd case
21.             random a number between 0 and 1, call it gap
22.          }
23.          else if (sl_ratio > 1) {  ; the 4th case
24.             random a number between 0 and 1/sl_ratio, call it gap
25.          }
26.          else if (sl_ratio < 1) {  ; the 5th case
27.             random a number between 1-sl_ratio and 1, call it gap
28.          }
29.          dif = n[atti] - p[atti]
30.          s[atti] = p[atti] + gap·dif
31.       }
32.       D' = D' $\cup$ {s}
33.    }
34. }
35. return D'
```

Fig. 1. Safe-Level-SMOTE algorithm

4 Experiments

In our experiments, we use four performance measures; *precision*, *recall*, *F-value*, and AUC, for evaluating the performance of three over-sampling techniques; Safe-Level-SMOTE, SMOTE, and Borderline-SMOTE. The value of β in *F-value* is set to 1 and the value of k in all over-sampling techniques are set to 5. The performance measures are evaluated through 10-fold cross-validation. Three classifiers; decision trees C4.5 [18], Naïve Bayes [12], and support vector machines (SVMs) [6], are applied as classifiers in the experiments. We use two quantitative datasets from UCI Repository of Machine Learning Databases [1]; Satimage and Haberman, illustrated in Table 3. The first to last column of the table represents the dataset name, the number of instances, the number of attributes, the number of positive instances, the number of negative instances, and the percent of a minority class, respectively.

The experimental results on the two datasets are illustrated in Fig. 2. The x-axis in these figures represents the over-sampling percent on a minority class. The y-axis in these figures represents the four performance measures; *precision*, *recall*, *F-value*, and AUC, in order from Fig. 2 (a) to Fig. 2 (d). In these figures, ORG, SMOTE, BORD, and SAFE are the label of the original dataset, SMOTE, Borderline-SMOTE, and Safe-Level-SMOTE, respectively.

For Satimage dataset, we select the class label 4 as the minority class and merge the remainder classes as the majority class because we only study the two-class imbalanced problem. The results on *F-value* using decision trees C4.5 are illustrated in Fig. 2 (c). It is apparent that *F-value* is improved when over-sampling percent on the minority class is increased. Moreover, Safe-Level-SMOTE achieved higher *F-value* than SMOTE and Borderline-SMOTE. The results on *recall* using Naïve Bayes are illustrated in Fig. 2 (b). Analyzing the figure, Borderline-SMOTE gains the higher performance on *recall*, while Safe-Level-SMOTE comes second.

For Haberman dataset, the minority class is about one quarter of the whole dataset. The results on *precision* using decision trees C4.5 are illustrated in Fig. 2 (a). The performance of Safe-Level-SMOTE is the best performance on *precision*. The results on AUC using SVMs are illustrated in Fig. 2 (d). Analyzing the figure, Safe-Level-SMOTE and SMOTE show similar performance on AUC. In addition, Borderline-SMOTE shows poor performance on higher percent.

For all experimental results, Safe-Level-SMOTE obviously achieve higher performance on *precision* and *F-value* than SMOTE and Borderline-SMOTE when decision trees C4.5 are applied as classifiers. Borderline-SMOTE only achieve a better performance on *recall* when Naive Bayes are applied as classifiers since the independent assumption on the borderline region is valid. Moreover, the SVMs show no improvement in all over-sampling techniques. Theses causes by the convex regions of all over-sampling techniques are similar. Therefore, the results of hyperplanes in SVMs are indistinguishable.

Table 3. The descriptions of UCI datasets in the experiments

Name	Instance	Attribute	Positive	Negative	% Minority
Satimage	6,435	37	626	5,809	9.73
Haberman	306	4	81	225	26.47

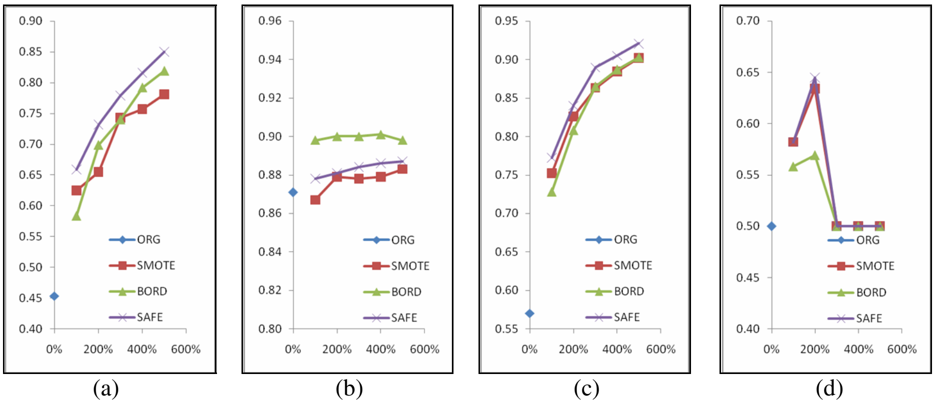

Fig. 2. The experimental results; (a) Precision evaluated by applying C4.5 with Haberman, (b) Recall evaluated by applying Naïve Bayes with Satimage, (c) F-value evaluated by applying C4.5 with Satimage, (d) AUC evaluated by applying SVMs with Haberman

5 Conclusion

The class imbalanced problem has got more attentions among data miners. There are many techniques for handling such problem. However, traditional data mining techniques are still unsatisfactory. We present an efficient technique called Safe-Level-SMOTE to handle this class imbalanced problem.

The experiments show that the performance of Safe-Level-SMOTE evaluated by *precision* and *F-value* are better than that of SMOTE and Borderline-SMOTE when decision trees C4.5 are applied as classifiers. This comes from the fact that Safe-Level-SMOTE carefully over-samples a dataset. Each synthetic instance is generated in safe position by considering the *safe level ratio* of instances. In contrast, SMOTE and Borderline-SMOTE may generate synthetic instances in unsuitable locations, such as overlapping regions and noise regions. We can conclude that synthetic instances generated in safe positions can improve prediction performance of classifiers on the minority class.

Although the experimental results have provided evidence that Safe-Level-SMOTE can be successful classified numeric datasets in the class imbalanced problem, there are several future works left to be studied in this line of research. First, different definitions to assign *safe level* would be valuable. Second, additional methods to classify datasets which have nominal attributes are useful. Third, automatic determination of the amount of synthetic instances generated by Safe-Level-SMOTE should be addressed.

References

1. Blake, C., Merz, C.: UCI Repository of Machine Learning Databases. Department of Information and Computer Sciences, University of California, Irvine, CA, USA (1998), `http://archive.ics.uci.edu/ml/`
2. Bradley, A.: The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition 30(6), 1145–1159 (1997)

3. Buckland, M., Gey, F.: The Relationship between Recall and Precision. Journal of the American Society for Information Science 45(1), 12–19 (1994)
4. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
5. Chawla, N., Japkowicz, N., Kolcz, A.: Editorial: Special Issue on Learning from Imbalanced Data Sets. SIGKDD Explorations 6(1), 1–6 (2004)
6. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
7. Domingos, P.: Metacost: A General Method for Making Classifiers Cost-sensitive. In: The 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999), pp. 155–164. ACM Press, San Diego (1999)
8. Fan, W., Miller, M., Stolfo, S., Lee, W., Chan, P.: Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. In: The 1st IEEE International Conference on Data Mining (ICDM 2001), San Jose, CA, USA, pp. 123–130 (2001)
9. Fan, W., Salvatore, S., Zhang, J., Chan, P.: AdaCost: misclassification cost-sensitive boosting. In: The 16th International Conference on Machine Learning (ICML 1999), Bled, Slovenia, pp. 97–105 (1999)
10. Han, H., Wang, W., Mao, B.: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)
11. Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies. In: the 2000 International Conference on Artificial Intelligence (IC-AI 2000), Las Vegas, NV, USA, pp. 111–117 (2000)
12. Kamber, M., Han, J.: Data mining: Concepts and Techniques, 2nd edn., pp. 279–327. Morgan-Kaufman, NY, USA (2000)
13. Kubat, M., Holte, R., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30, 195–215 (1998)
14. Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: The 14th International Conference on Machine Learning (ICML 1997), pp. 179–186. Morgan Kaufmann, Nashville (1997)
15. Lewis, D., Catlett, J.: Uncertainty Sampling for Supervised Learning. In: The 11th International Conference on Machine Learning (ICML 1994), pp. 148–156. Morgan Kaufmann, New Brunswick (1994)
16. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Brunk, C.: Reducing Misclassification Costs. In: The 11th International Conference on Machine Learning (ICML 1994), pp. 217–225. Morgan Kaufmann, San Francisco (1994)
17. Prati, R., Batista, G., Monard, M.: Class Imbalances versus Class Overlapping: an Analysis of a Learning System Behavior. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 312–321. Springer, Heidelberg (2004)
18. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1992)
19. Tetko, I., Livingstone, D., Luik, A.: Neural network studies. 1. Comparison of Overfitting and Overtraining. Chemical Information and Computer Sciences 35, 826–833 (1995)