A Pruning-Based Approach for Searching Precise and Generalized Region for Synthetic Minority Over-Sampling

Kamthorn Puntumapon and Kitsana Waiyamai

Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Thailand, 10240
kamthorn.puntumapon@gmail.com, fengknw@ku.ac.th

Abstract. One solution to deal with class imbalance is to modify its class distribution. Synthetic over-sampling is a well-known method to modify class distribution by generating new synthetic minority data. Synthetic Minority Over-sampling TEchnique (SMOTE) is a state-of-the-art synthetic over-sampling algorithm that generates new synthetic data along the line between the minority data and their selected nearest neighbors. Advantages of SMOTE is to have decision regions larger and less specific to original data. However, its drawback is the over-generalization problem where synthetic data is generated into majority class region. Over-generalization leads to misclassify non-minority class region into minority class. To overcome the over-generalization problem, we propose an algorithm, called TRIM, to search for precise minority region while maintaining its generalization. TRIM iteratively filters out irrelevant majority data from the precise minority region. Output of the algorithm is the multiple set of seed minority data, and each individual set will be used for generating new synthetic data. Compared with state-of-the-art over-sampling algorithms, experimental results show significant performance improvement in terms of F-measure and AUC. This suggests over-generalization has a significant impact on the performance of the synthetic over-sampling method.

1 Introduction

Imbalanced data has been identified as one of ten most challenging problems in data mining [14]. A dataset is considered imbalanced when its class distribution is skewed. The skewed class distribution can be represented as a binary class, i.e., minority and majority class. The minority class is the one having a much smaller proportion of class examples, whereas the majority class contains a much higher proportion of class examples. In classification, we want to correctly classify on the both classes. However, most traditional classifiers are biased towards the larger number of examples. For example, the C4.5 decision tree algorithm [10] assumes data is from a well balanced class distribution. As a result, in cases of imbalanced class distribution, the algorithm is biased toward the majority class and treats the minority class as noise.

Solutions to the imbalanced class problem can be broadly divided into two categories [13]: algorithm level and data level. Data level solutions are independent of the classification algorithm; the preprocessed data can be used by any traditional classification methods. SMOTE [3] is a state-of-the-art of synthetic over-sampling algorithm that generates synthetic data along the line between minority data members and their selected nearest neighbors. The advantage of SMOTE is to have decision regions larger and less specific to the original data [9]. However, it suffers from over-generalization where synthetic data is generated into the majority class region. The reason is that SMOTE selects its neighbors without regard to the majority class. This problem leads to misclassifying non-minority class examples into minority class region.

Recently, many synthetic over-sampling methods [2–4, 7, 8] have been proposed to overcome the imbalanced class problem. Most of the proposed algorithms are based on SMOTE and directly select seed examples to generate new synthetic data. Borderline-SMOTE [7] directly selects seed examples based on the decision boundary. Borderline data is used as a seed example to generate new synthetic data. Safe-Level-SMOTE [2] directly positions synthetic data between two minority examples based on the number of minority data neighbors. However, these methods are not intended to solve the over-generalization problem. To the best of our knowledge, MSYN [4] is the first synthetic over-sampling method that tries to overcome over-generalization. MSYN uses 1-NN's margin to select a synthetic example. However, in case of a very small number of minority data, MSYN has the same problem as other methods as well.

In this paper, we propose an algorithm to overcome the over-generalization problem in imbalanced data. To avoid over-generalization, a greedy filtering strategy is employed to search for precise and generalized sets of minority data. Individually, precise and generalized sets are then used as seed sets to generate synthetic minority data. TRIM is a preprocessing algorithm for synthetic over-sampling methods. Therefore, it can be used as a preprocessor for most existing synthetic over-sampling methods. In this paper, TRIM is used as a preprocessing step to generate synthetic minority data for SMOTE, called TRIM-SMOTE. The experimental results show significant performance improvements in terms of F-measure and AUC over SMOTE. For F-measure, TRIM-SMOTE statistical significance outperforms SMOTE in 26 out of 33 experiments. For AUC, TRIM-SMOTE significance outperforms SMOTE in 22 out of 33 experiments.

2 Related Work

Studies using synthetic minority over-sampling [2–4, 7, 8] as a technique to balance class distribution in the literature are designed based on the Synthetic Minority Over-sampling TEchnique (SMOTE) [3]. SMOTE employs k-nearest neighbors (k-NN) as a range to couple two minority examples; new synthetic data is randomly generated along the line between the two minority examples. That is, SMOTE uses only minority data to generate new synthetic data without regard to the majority class. The way SMOTE generates new data can be interpreted as merging the two minority data into a disjunct with synthetic data. As a result, SMOTE makes decision regions larger and less specific to the original data. That is, the minority class is generalized and even over-generalized. There have been several other techniques to generate minority synthetic data; in this section we describe some significant works relevant to the work here.

Borderline-SMOTE (BSMOTE) [7] uses the basic assumption that data nearby the decision boundary has more chance to be misclassified than data far from the decision boundary. The author further proposed a criteria to identify borderline data based on k-NN; BSMOTE uses both minority and majority data in k-NN. The ratio of the number of neighborhood minority examples is used as criterion to identify importance of borderline data. Only borderline data and their neighbors are used to generate synthetic data. As a result, synthetic data is generated in the overlapping region between two classes. Therefore, borderline SMOTE also suffers from over-generalization. The algorithm may cause severe over-generalization due to the focus sampling in the overlap area.

Safe-Level-SMOTE (SSMOTE) [2] has been proposed to directly position synthetic examples instead of random positioning. The safe level criteria is based on the number of neighborhood minority data in k-NN. That is, the higher number of neighborhood minority data, the safer the position is. Since Safe-Level-SMOTE is based on SMOTE, new synthetic data are generate on a line between two minority examples. A new synthetic example is generated nearby a minority example that has higher number of neighborhood minority data.

To the best of our knowledge, Margin-guided Synthetic Over-sampling (MSYN) [4] is the first synthetic over-sampling method that claims to overcome over-generalization. MSYN uses 1-NN margin to estimate goodness of the synthetic data. The algorithm bias prefers synthetic data that has a large margin on both minority and majority classes. The synthetic data is ranked by the margin, MSYN then selects the top M values for use as synthetic minority data. MSYN trends to generate new synthetic data on a well separated region while avoiding the boundary region. Although MSYN can be used to avoid over-generalization, it uses all features to select a synthetic example. However, many existing classification methods usually use several good features to classify data. In this work, we propose a method to select precise seed sets on particular features. The proposed method searches one feature space at a time to filter out irrelevant data while maintaining seed data.

3 Methodology

The goal of the TRIM algorithm is to avoid over-generalization. The basic idea is to identify sets of minority data having the best compromise between generalization and precision. The algorithm starts with a single set containing all the data. For each feature, every splitting point is identified and evaluated based on the TRIM criteria, described below. A splitting point is identified by taking the midpoint between two adjacent sorted values having different classes. The best splitting point will be used to split the data into two sets, i.e., left and right sets. Iteratively, each set of minority data is split into smaller sets until the stopping criterion is satisfied. The final sets are minority datasets are used as seed data to generate synthetic data via SMOTE, namely TRIM-SMOTE.

3.1 TRIM Criteria

To avoid over-generalization, the seed data should be precise while maintaining their generalization. Therefore, the TRIM criteria, Equation (1), is used as a measure of precision and generalization. The higher the $TRIM$ value, the more precise and generalized the seed data.

$$TRIM = \frac{|minority|^2}{N} \tag{1}$$

where $|minority|$ is the size of the minority data. To get better seed data, TRIM Gain ($T - Gain$) is evaluated against two splitting datasets, i.e, left and right sets. $T - Gain$ is compared to $TRIM$. If $T - Gain > TRIM$, a better seed data is obtained by a binary splitting operation. Define $|minority_{left}|$ and $|minority_{right}|$ as the number of minority data in the left and right subsets; let N be the total number of examples, N_{left}, and N_{right} be the number of data in the left and right subsets. With this notation,, $T - Gain$ can be calculated as

$$T - Gain = max(\frac{|minority_{left}|^2}{N_{left}}, \frac{|minority_{right}|^2}{N_{right}}) \tag{2}$$

The equation (2) is designed to capture two characteristics of SMOTE. The first characteristic is to generate new synthetic data from a couple of minority examples. Therefore, only minority data is used to evaluate precision. The second characteristic is synthetic data is always generated within the convex hull of the minority data. Therefore, another objective of $T - Gain$ is to identify irrelevant majority data located outside the convex hull and filter them out.

Fig. 1 shows a dataset containing two classes. Minority examples are shown as diamonds, and majority examples are stars. The convex hull of the minority class is illustrated by a dashed line. The relevant majority data is A,B,C,D, and E. The rest of the majority data are irrelevant. The seed data lie within the solid line. Similar to decision trees, seed data are modeled with a hyper-rectangle. Thus, the irrelevant majority data that locate inside the solid line will be mis-identified as relevant majority data.

The standard maximum function is used in Equation 2 to focus on improvement of the seed set. The first objective is to filter out irrelevant majority data located outside the convex hull of the minority data. We want to focus on improvement of seed data while ignore irrelevant data. The second objective is to identify which set of data is splitting seed data. To obtain a more precise seed data, some minority examples are split from the old seed data. The max is used to detect which of the subsets to use as splitting seed data. The splitting seed data is then compared against the old seed data to evaluate the improvement.

3.2 TRIM Algorithm

The TRIM algorithm uses a greedy approach to search for set of minority data while filtering out irrelevant data. Although this approach does not guarantee finding the global optimum, it provides a good approximation to the optimal set. The following pseudocode describes the algorithm.

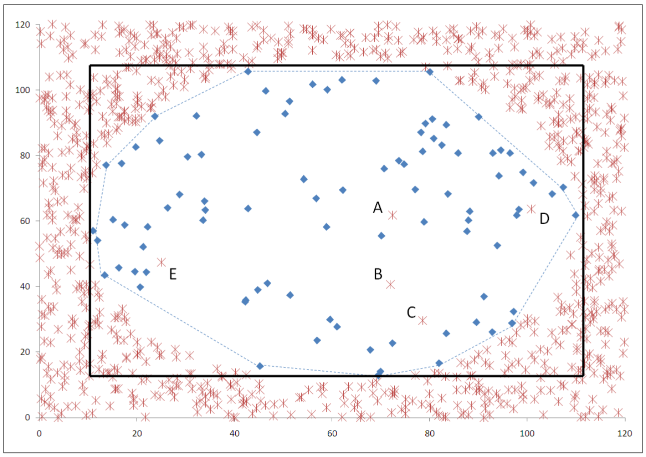

Fig. 1. The artificial dataset with its convex hull and seed set boundary after processing by TRIM

```
Algorithm TRIM(D)
Input: data $D$
Output: list $Seed$

1.  add $D$ to $Leaf$
2.  While $Leaf$ and $Candidate$ is not empty{
3.   For each $Leaf$ {
4.     Initialize all splitting points $S$ on $Leaf_i$
5.     calculate $TRIM$ on $Leaf_i$
6.     For each $S$ {
7.       accept the highest $T - Gain_{max}$ that does not split any minority data at $S_{max}$
    splitting point
8.     }
9.     if ($T - Gain_{max} > TRIM$) {
10.      split $Leaf_i$ into $Leaf_{left}$ and $Leaf_{right}$ using $S_{max}$
11.      if ($subset_{left}$ contains any minority data)
12.       add $Leaf_{left}$ to $Leaf$
13.      if ($subset_{right}$ contains any minority data)
14.       add $Leaf_{right}$ to $Leaf$
15.      remove $Leaf_i$ from $Leaf$
16.    } else {
17.      add $Leaf_i$ to $Candidate$
18.      remove $Leaf_i$ from $Leaf$
19.    }
20.  }
21.  For each $Candidate$ {
22.    Initialize all splitting points $S$ on $Candidate_i$
23.    calculate $TRIM$ on $Candidate_i$
24.   For each $S$ {
25.     accept highest $T - Gain_{max}$ at $S_{max}$ splitting point
26.   }
27.   if ($T - Gain_{max} > TRIM$) {
28.     split $Candidate_i$ into $Candidate_{left}$ and $Candidate_{right}$ using $S_{max}$
29.     if ($Candidate_{left}$ contains any minority data)
30.       add $Candidate_{left}$ to $Leaf$
31.     if ($Candidate_{right}$ contains any minority data)
32.       add $Candidate_{right}$ to $Leaf$
33.     remove $Candidate_i$ from $Candidate$
34.   } else {
35.     add $Candidate_i$ to $Seed$
36.     remove $Candidate_i$ from $Candidate$
37.   }
38. }
39. }
40. return $Seed$
```

In line 1, the algorithm starts with a single set containing all data. The algorithm consists of two main steps which are irrelevant data filtering and candidate splitting. The first step, irrelevant data filtering, is shown in lines 3-20. The main objective of this step is to filter out irrelevant majority data. The splitting point will be used with a constraint that is no minority data will be split into different set. This constraint ensures only irrelevant majority data is split in this step and the relevant majority data is used to splits minority data in the next step. Candidate splitting is performed in lines 21-39. This step focuses on splitting some minority data to get more precise seed data. Both split minority data and new seed data will be used in the 1st step of the next iteration. This process iterates until no more better splitting point is found.

In lines 4-5 of the irrelevant data filtering step, the algorithm starts by initializing available splitting points and $TRIM$ value of the whole dataset. In lines 6-8, $T - Gain$ is evaluated on every splitting point that splits all minority data into one set. In lines 10-15, the data is split into two sets (left and right). All majority data will be filtered out if they do not contain any minority examples. If no irrelevant data is found, the data is sent to the second step at line 17.

In candidate splitting, the algorithm is similar to the filtering step. However, this step focuses only on minority data, as shown in line 25. Notice that each seed data will be processed in both steps at least once. In order to maintain the generalization, the algorithm does not split minority data from majority data that are located outside the convex hull. To ensure this condition, all the irrelevant majority data are filtered out in the filtering step. Thus, the second step will split minority data based only on relevant majority and minority data.

The higher the $T - Gain$ in Equation (2), the more precise and generalized data is obtained. In line 35, seed data will be generated when $T - Gain \leq TRIM$. This can be interpreted as meaning that no more precise and generalized data can be found. Although seed data is well-approximated, some seed data can be considered as noise. To filter out noise, a threshold minimum precision ($minPrecision$) is used to select seed data. Define $|minority_i|$ as the number of minority data in $Seed_i$; N_i be the total number of data in $Seed_i$. The precision of $Seed_i$ ($precision_i$) can be expressed mathematically as shown in Equation (3).

$$precision_i = \frac{|minority_i|}{N_i} \tag{3}$$

The precision value of a seed is compared against $minPrecision$. The seed set will be filtered out if $precision < minPrecision$. Intuitively, if we set $minPrecision$ too high, we lose some information. However, if we set $minPrecision$ too low, noise can happen in the seed data. We experimentally selected $minPrecision = 0.3$ as suitable to preserve information while filtering out noise.

The TRIM algorithm calculates $T - Gain$ on every splitting point. Define M as the number of attributes and N be number of data values. The maximum number of splitting points is $N - 1$, and the maximum number of iterations is $N - 1$ in the case where only one example data is split at each iteration. Hence, the time complexity of the algorithm is $O(MN^2)$. However, in experiments the critical step was found to be data sorting, having complexity $O(MNlog(N))$.

4 Experimental Results

In the following, the results of TRIM-SMOTE, SMOTE, and MSYN are compared using 11 continuous datasets. All experiments were conducted using 10 random seeds on 10-fold cross validation with three levels of sampling, i.e., 100%, 300%, 500%. Two measurements, AUC [1] and F-measure [12], are used to evaluate performance. Each algorithm was used in at least 3,300 experiments. The experiments use WEKA's C4.5 [6] as a classification model with default configuration, i.e., weka.classifiers.trees.J48 -C 0.25 -M 2. For TRIM, $minPrecision$ was set to 0.3; parameters for the other algorithms were set exactly same as in their papers.

Table 1. Experimental datasets

#	Dataset	#Attributes	%Minority	#Minority	Data size
1	letter-A	16	3.94	789	20000
2	arrhythmia-6	280	5.53	25	452
3	libras-10	90	6.67	24	360
4	glass-3	10	7.9	17	214
5	mfeat-fourier1	76	10.0	200	2000
6	yeast-ME3	9	10.9	162	1484
7	breastTissue-fad	10	14.15	15	106
8	segment-path	19	14.28	330	2310
9	bloodTransfer	4	23.7	178	748
10	haberman	3	26.4	81	306
11	ionoshpere	35	35.8	126	351

In Table 1 the 11 UCI datasets [5] are sorted by percentage of minority class. The percentages vary from 3.9% to 35.8%. Each dataset has five attributes, i.e., dataset's name, number of attributes (#Attributes), percentage of minority data (%Minority), number of minority data (#Minority), and size of dataset (Data size). All datasets have binary class, i.e, minority class and majority class. However, the 1st - 8th datasets have more than two classes. Therefore, we used one class as the minority class, and grouped the others as a majority class. For example, yeast-ME3 contains two classes, i.e., ME3 and others. The ME3 and others are considered as minority and majority classes, respectively.

To illustrate the impact of over-generalization, we examine results for the the segment-path and ionosphere datasets in term of AUC and F-Measure.

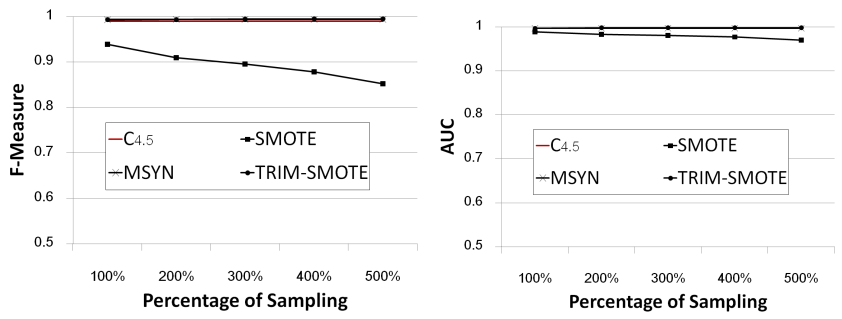

Fig. 2. The negative impact of over-generalization in term of F-Measure and AUC on segment-path dataset

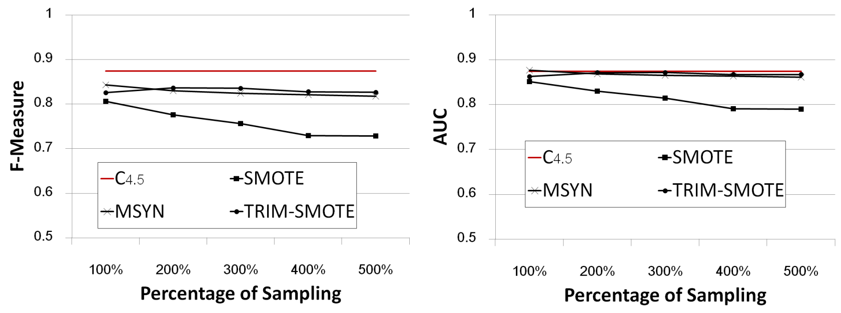

Fig. 3. The negative impact of over-generalization in term of F-Measure and AUC on ionoshpere dataset

As shown in Fig. 2, the three algorithms (decision tree C4.5, TRIM-SMOTE, and MSYN) exhibit comparable performance with respect to both AUC and F-measure. This can be interpretted to mean that performance improvement is not guaranteed by synthetic over-sampling methods. However, SMOTE is degraded on every percentage of sampling and its performance is worse when the sampling percentage increases. The reason is that MSYN and TRIM are designed to handle over-generalization while SMOTE suffers from this problem.

In Fig. 3, every synthetic over-sampling method, i.e., SMOTE, MSYN, and TRIM-SMOTE yields lower performance with respect to F-measure when compared to C4.5. However, we obtained stable performance with MSYN and TRIM-SMOTE, i.e., 82.8% F-measure with 0.7% standardization. In contrast, SMOTE always yields lower performance on every increasing sampling percentage. In terms of AUC, decision tree C4.5, TRIM-SMOTE, and MSYN show comparable performance while SMOTE is always lower. These four graphs illustrate the negative impact of over-generalization.

Table 2. Result in terms of F-measure in the experiments performs on eleven UCI datasets. A value will be underlined when the particular algorithm yield the highest F-measure among all four algorithms.

Dataset	%Minority	#Minority	%Sampling	C4.5	SMOTE	MSYN	TRIM-SMOTE
letter-A	3.94	16	100%	94.7%	93.9%	94.3%	$\underline{95.2\%}$
			300%	94.7%	92.3%	93.7%	$\underline{95.2\%}$
			500%	94.7%	91.8%	93.7%	$\underline{94.8\%}$
arrhythmia-6	5.53	25	100%	62.7%	58.1%	58.1%	$\underline{64.3\%}$
			300%	62.7%	40.8%	40.9%	$\underline{64.1\%}$
			500%	62.7%	36.2%	36.5%	$\underline{69.6\%}$
libras-10	6.67	24	100%	$\underline{57.1\%}$	56.6%	56.2%	57.0%
			300%	57.1%	52.7%	59.0%	$\underline{60.3\%}$
			500%	57.1%	50.8%	56.8%	$\underline{65.6\%}$
glass-3	7.9	17	100%	$\underline{51.6\%}$	35.0%	42.5%	47.7%
			300%	$\underline{51.6\%}$	30.3%	39.8%	47.9%
			500%	$\underline{51.6\%}$	29.6%	37.4%	43.8%
mfeat-fourier1	10.0	200	100%	97.4%	89.3%	$\underline{98.1\%}$	96.7%
			300%	97.4%	79.7%	$\underline{98.3\%}$	96.9%
			500%	97.4%	75.7%	$\underline{98.3\%}$	96.7%
yeast-ME3	10.9	162	100%	76.3%	76.3%	77.3%	$\underline{77.4\%}$
			300%	$\underline{76.3\%}$	74.9%	76.0%	$\underline{76.3\%}$
			500%	76.3%	75.0%	76.5%	$\underline{78.4\%}$
breastTissue-fad	14.15	15	100%	$\underline{41.6\%}$	26.7%	38.2%	36.9%
			300%	$\underline{41.6\%}$	34.8%	36.9%	39.4%
			500%	41.6%	33.2%	34.7%	$\underline{42.0\%}$
segment-path	14.28	330	100%	$\underline{99.2\%}$	93.8%	$\underline{99.2\%}$	$\underline{99.2\%}$
			300%	99.2%	89.5%	99.2%	$\underline{99.3\%}$
			500%	99.2%	85.1%	99.2%	$\underline{99.3\%}$
bloodTransfer	23.7	178	100%	46.9%	47.9%	48.3%	$\underline{48.5\%}$
			300%	46.9%	47.4%	47.6%	$\underline{48.6\%}$
			500%	46.9%	47.7%	47.0%	$\underline{47.8\%}$
haberman	26.4	81	100%	41.1%	49.5%	48.1%	$\underline{51.1\%}$
			300%	41.1%	47.9%	$\underline{50.1\%}$	48.2%
			500%	41.1%	45.6%	$\underline{49.3\%}$	48.1%
ionoshpere	35.8	126	100%	83.3%	80.5%	$\underline{84.2\%}$	82.5%
			300%	83.3%	75.5%	82.3%	$\underline{83.4\%}$
			500%	$\underline{83.3\%}$	72.8%	81.7%	82.5%
Win/Draw/Lose Significant				17/7/9	26/0/7	17/6/10	NA
1st rank				9	0	7	20

Performance of C4.5 (without sampling), SMOTE, MSYN, and TRIM-SMOTE on the eleven datasets in terms of F-measure and AUC are shown in Table 2 and 3. C4.5 is used as a baseline to evaluate improvement or decreased performance of the three algorithms. The bottom rows provide a comparison of each of the three algorithms with TRIM-SMOTE. The row labeled (Win/Draw/Lose Significant) summarizes the number of cases where the algorithm significant outperforms, equals, or performs worse than TRIM-SMOTE. To evaluate statistical significant, Wilcoxon signed rank test [11]

Table 3. Result in terms of AUC in the experiments performs on eleven UCI datasets. A value will be underlined when the particular algorithm yield the highest AUC among all four algorithms.

Dataset	%Minority	#Minority	%Sampling	C4.5	SMOTE	MSYN	TRIM-SMOTE
			100%	96.5%	96.8%	96.2%	97.3%
letter-A	3.94	16	300%	96.5%	96.3%	96.1%	97.5%
			500%	96.5%	96.2%	96.3%	97.3%
			100%	80.8%	82.7%	82.7%	81.8%
arrhythmia-6	5.53	25	300%	80.8%	73.7%	73.6%	82.2%
			500%	80.8%	71.8%	72.0%	86.5%
			100%	77.5%	79.8%	77.2%	78.0%
libras-10	6.67	24	300%	77.5%	80.5%	80.5%	78.0%
			500%	77.5%	83.0%	79.3%	78.3%
			100%	72.0%	64.2%	68.5%	72.4%
glass-3	7.9	17	300%	72.0%	64.3%	68.5%	71.8%
			500%	72.0%	67.9%	66.9%	70.4%
			100%	98.3%	98.0%	98.8%	98.0%
mfeat-fourier1	10.0	200	300%	98.3%	96.7%	99.1%	98.2%
			500%	98.3%	96.1%	99.2%	98.2%
			100%	87.0%	89.4%	88.4%	89.0%
yeast-ME3	10.9	162	300%	87.0%	90.3%	87.8%	89.5%
			500%	87.0%	91.4%	88.1%	90.8%
			100%	64.4%	57.8%	65.9%	64.5%
breastTissue-fad	14.15	15	300%	64.4%	65.8%	64.3%	65.7%
			500%	64.4%	65.4%	63.2%	67.7%
			100%	99.6%	98.8%	99.6%	99.6%
segment-path	14.28	330	300%	99.6%	97.9%	99.6%	99.7%
			500%	99.6%	96.9%	99.6%	99.7%
			100%	65.2%	65.8%	66.1%	66.2%
bloodTransfer	23.7	178	300%	65.2%	65.4%	65.6%	66.3%
			500%	65.2%	65.7%	65.2%	65.7%
			100%	61.0%	65.4%	64.7%	66.6%
haberman	26.4	81	300%	61.0%	63.8%	66.0%	64.2%
			500%	61.0%	61.6%	65.3%	64.0%
			100%	86.5%	85.1%	87.6%	86.2%
ionoshpere	35.8	126	300%	86.5%	81.3%	86.4%	87.1%
			500%	86.5%	78.9%	86.0%	86.6%
Win/Draw/Lose Significant				18/2/13	22/3/8	13/5/15	NA
1st rank				8	4	8	15

with p-value greater than or equal to 95% evaluates on every cross validation result. The second row shows number of cases where the algorithm obtains the highest performance among all four algorithms. The result is underlined when it obtains the highest performance.

Table 2 shows the experimental results evaluated using F-measure. The table shows that TRIM-SMOTE provided the best classification in 20 of 33 cases, and was almost similar to C4.5. The table shows that SMOTE generally underperformed both C4.5 and TRIM-SMOTE. For datasets having a very small number of minority examples, namely, arrhythmia-6, libras-10, glass-3, and breastTissue-fad, SMOTE and MSYN show lower performance than C4.5, whereas TRIM-SMOTE is most similar to C4.5, showing more stable performance.

Table 3 shows experimental results evaluated using AUC. The table shows that SMOTE generally underperformed both C4.5 and TRIM-SMOTE. For the datasets having a very small number of minority examples, namely, arrhythmia-6, glass-3, and breastTissue-fad, SMOTE and MSYN show lower performance than C4.5, whereas TRIM-SMOTE is most similar to C4.5, showing more stable performance. For libras-10 and yeast-ME3, SMOTE shows highest performance in terms of AUC, while TRIM-SMOTE yielded the highest F-measure. This can be explained by the fact that SMOTE randomly generates synthetic data on every minority data without regarding to majority data, whereas TRIM-SMOTE tries to maintain a precise and generalized set of seed data. As a result, TRIM-SMOTE produces comparable recall but higher precision while SMOTE gain higher recall but lower precision.

These experimental results indicate that, of the four algorithms, SMOTE underperforms C4.5 the most, and its performance declines as the sampling percentage increases. We observed stable performance for MSYN on large datasets but less than C4.5 for small datasets.

5 Conclusions

Synthetic minority over-sampling is a method that generates new minority examples to balance an imbalanced class distribution. The advantage is that synthetic data does not duplicate the original minority data. Therefore, the classification model is not overfitted to synthetic data. Synthetic over-sampling has its own drawback: over-generalization. The problem is that the majority class region is confounded with synthetic minority examples. To overcome over-generalization, we propose an algorithm called TRIM as a preprocessing for synthetic over-sampling. TRIM searches for a set of precise minority examples while maintaining their generalization. The precise seed set is used as an input to a synthetic minority over-sampling method. Thus, TRIM can be used as a preprocessing algorithm for many available synthetic over-sampling techniques such as SMOTE, BSMOTE, and MSYN. Prior to sampling, evaluation or interpretation of the seed data can also be conducted. Empirical results also show encouraging improvement over SMOTE and MSYN. TRIM-SMOTE is able to cope with the over-generalization problem more than MSYN. Its performance is stable on large dataset, and increased on small datasets. Experiments on multi-class and multivariate datasets are to be explored in the future study.

References

[1] Bradley, A.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)

[2] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 475–482. Springer, Heidelberg (2009), `http://dblp.uni-trier.de/db/conf/pakdd/pakdd2009.html`

[3] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)

[4] Fan, X., Tang, K., Weise, T.: Margin-Based Over-Sampling Method for Learning from Imbalanced Datasets. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 309–320. Springer, Heidelberg (2011)

[5] Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`

[6] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explorations 11(1), 10–18 (2009)

[7] Han, H., Wang, W.-Y., Mao, B.-H.: 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, Part I. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)

[8] He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IJCNN, pp. 1322–1328. IEEE (2008), `http://dblp.uni-trier.de/db/conf/ijcnn/ijcnn2008.html`

[9] He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21, 1263–1284 (2009)

[10] Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)

[11] Ramsey, P.H., Hodges, J.L., Shaffer, J.P.: Significance probabilities of the wilcoxon signed-rank test. Journal of Nonparametric Statistics 2(2), 133–153 (1993), `http://www.informaworld.com/10.1080/10485259308832548`

[12] van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Butterworths, London (1979)

[13] Weiss, G.M.: Mining with rarity: a unifying framework. SIGKDD Explorations 6(1), 7–19 (2004), `http://dblp.uni-trier.de/db/journals/sigkdd/sigkdd6.html`

[14] Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology and Decision Making 5(4), 597–604 (2006), `http://dblp.uni-trier.de/db/journals/ijitdm/ijitdm5.html`