Hindawi
Scientific Programming
Volume 2020, Article ID 8837357, 9 pages
https://doi.org/10.1155/2020/8837357

Research Article

AGNES-SMOTE: An Oversampling Algorithm Based on Hierarchical Clustering and Improved SMOTE

Xin Wang,[1,2] Yue Yang,[1] Mingsong Chen,[2,3] Qin Wang,[2] Qin Qin,[2] Hua Jiang,[1] and Huijiao Wang[1]

[1]*School of Computer Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*
[2]*Beihai Campus, Guilin University of Electronic Technology, Beihai, Guangxi 536000, China*
[3]*School of Information and Communication, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*

Correspondence should be addressed to Yue Yang; 463164648@qq.com and Qin Qin; 357715896@qq.com

Received 23 April 2020; Revised 7 August 2020; Accepted 9 September 2020; Published 23 September 2020

Academic Editor: Fabrizio Riguzzi

Aiming at low classification accuracy of imbalanced datasets, an oversampling algorithm—AGNES-SMOTE (Agglomerative Nesting-Synthetic Minority Oversampling Technique) based on hierarchical clustering and improved SMOTE—is proposed. Its key procedures include hierarchically cluster majority samples and minority samples, respectively; divide minority subclusters on the basis of the obtained majority subclusters; select "seed sample" based on the sampling weight and probability distribution of minority subcluster; and restrict the generation of new samples in a certain area by centroid method in the sampling process. The combination of AGNES-SMOTE and SVM (Support Vector Machine) is presented to deal with imbalanced datasets classification. Experiments on UCI datasets are conducted to compare the performance of different algorithms mentioned in the literature. Experimental results indicate AGNES-SMOTE excels in synthesizing new samples and improves SVM classification performance on imbalanced datasets.

1. Introduction

Imbalanced dataset is featured with having fewer instances of some classes than others in a dataset. In the biclass cases, one class with fewer samples is referred to as a minority class, and the other class with more samples is the majority class [1]. In reality, there are many scenarios of imbalanced data classification, such as credit card fraud detection, information retrieval and filtering, and market analysis [2]. Conventional classifiers typically favor the majority class, giving rise to classification errors. The imbalance of sample sizes between two different classes is regarded as between-class imbalance, and the imbalanced data distribution density within one class is within-class imbalance. Within-class imbalance forms multiple subclasses with the same class but different data distribution [3, 4]. Both the two abovementioned imbalances will cause classification errors. In addition, oversampling algorithms often cause problems such as synthetic samples overlap [5] and samples distributed "marginally" [6], which reduce classification performance. Therefore, how to improve conventional algorithms to solve the imbalanced classification of datasets and promote classification performance becomes the research focus of data mining and machine learning.

Researches on imbalanced datasets mainly include data processing and classification algorithm [7, 8]. Cost-sensitive learning [9] and integrated learning [10] are representative classification algorithms. The most frequently used methods to process data are oversampling and undersampling methods, which balance two classes by increasing minority samples and decreasing majority samples, respectively. Sampling methods based on data are usually simple and intuitive. Undersampling method usually causes information loss while oversampling method tends to balance the original dataset. Thus, the latter one is often adopted in data classification.

At present, the most frequently used oversampling method is SMOTE algorithm proposed by Chawla's team

[11] in 2002. It created new synthetic samples by linear interpolation of sample x and sample y, in which x referred to an existing minority sample and another minority sample y was picked up randomly from the nearest neighbors of x. This algorithm neglected uneven data distribution in minority class and the possibility of samples overlap when synthesizing samples. Han Hui's team [12] suggested Borderline-SMOTE algorithm in 2005, which divided minority samples into boundary area, safe area, and dangerous area. This algorithm synthesized samples by selecting samples from the boundary area, which avoided selecting minority samples indiscriminately and produced lots of redundant new samples caused by SMOTE algorithm. ADASYN algorithm, proposed by He's team, [13] indicated that the samples size needed to be generated by each minority sample was automatically determined based on data distribution. Minority samples with more neighboring majority samples generated more samples. Compared with SMOTE, it divided the sample distribution exhaustively. Cluster-SMOTE [14] adopted the K-means algorithm to cluster minority samples, found minority subclusters, and, then, applied SMOTE algorithm, respectively. However, this algorithm did not determine the optimal size of subclusters and did not calculate the sample size generated by each subcluster. K-means-SMOTE [15] combined K-means clustering algorithm with SMOTE algorithm. Compared with Cluster-SMOTE, K-means-SMOTE clustered the entire datasets, found the overlap and avoided oversampling in unsafe areas, restricted the synthetic samples in the target area, and eliminated within-class and between-class imbalances. Meanwhile, it avoided noise samples and attained good results. CBSO [16] combined clustering with the data generation mechanism of the existing oversampling technology to ensure that the generated synthetic samples were always in the minority class area and avoided generating erroneous samples. Although the abovementioned oversampling methods indeed improve classification accuracy to a certain extent, they have the following deficiencies: (1) when oversampling, much attention has been paid to solving between-class imbalance while has been paid less attention to within-class imbalance. (2) Clustering can address the issue of between-class and within-class imbalance, but two classes aliasing are exacerbated, leading to generating new overlapping synthetic samples. The conventional k-means clustering algorithm needs to set k value when clustering, which is more effective for spherical datasets and is more complex. (3) The minority boundary is not maximized, affecting synthetic samples quality. (4) No restrictions on destination area of synthetic samples result in synthetic samples distributed marginally. (5) Noise samples interfere.

Based on the above discussion, this paper offers an improved oversampling method—AGNES-SMOTE. Its procedures are listed as follows: filter noise samples, adopt the AGNES algorithm to cluster minority samples, form the minority subclusters iteratively, and consider the majority samples distribution during the merging process to avoid generating overlapping synthetic samples. Repeat this operation until the distance between the two closest minority subclusters exceeds the set valve-value. Then, determine sampling weights according to sample size in minority subcluster, calculate the probability distribution of each minority subcluster according to the distance between the minority samples and their neighbor majority samples, and combine the two to select "seed sample" for oversampling. Restrict the generation of new samples in a certain area by centroid method in the synthesizing process. Select a sample from all "seed samples," randomly select two neighboring minority samples from the subcluster where the selected "seed sample" is located, form a triangle with the three selected samples, and synthesize new samples on the line from the three samples to the centroid. Compared with other algorithms, AGNES-SMOTE attains a better result in the experiment.

2. Preliminary Theory

2.1. SMOTE Algorithm. SMOTE algorithm alleviates the problem of data imbalance by artificially synthesizing new minority samples and calculates the distance between one sample $X = \{x_1, x_2, \ldots, x_n\}$ and the other sample $Y = \{y_1, y_2, \ldots, y_n\}$ by the Euclidean distance. The subscript numbers $1, 2, \ldots, n$ are sample dimension values. The Euclidean distance D between sample X and sample Y is

$$D = \sqrt{\sum_{k=1}^{n} (X_k - Y_k)^2}. \tag{1}$$

For each sample X in minority class, search for its nearest neighbor samples K and randomly select N samples from these nearest neighbor datasets. For each original sample, select N samples from K-nearest neighbor samples, and then perform interpolation between the original samples and their nearest neighbor samples. The formula is described as follows:

$$X_{\text{new}} = X + \text{rand}(0, 1) \times (Y_i - X), \tag{2}$$

where i is 1, 2, . . ., N; X_{new} is the new synthetic samples; X is the selected original sample; rand $(0, 1)$ is a random number between 0 and 1; and Y_i is N samples selected from the nearest K samples of the original sample X.

2.2. AGNES Algorithm. The conventional AGNES algorithm is all about hierarchical clustering. It treats every data as a cluster and gradually merges those clusters according to some certain criteria. For example, if the distance between the two data objects in different clusters is the smallest, the two clusters may be merged. The merging is repeated until a certain termination condition is met. In AGNES, the distance between clusters is attained by calculating the distance between the closest data objects in two clusters, so a cluster can be represented by all objects in the cluster.

Compared with aggregating samples with the conventional centroid method, the AGNES algorithm is more accessible, independent of the selected initial values, and free from the samples' distribution shape. It also can aggregate all samples together. Considering the influence of between-

class and within-class samples imbalance on model performance, the AGNES algorithm is more applicable to deal with unbalanced data distribution of within-class imbalance.

3. Improved SMOTE Algorithm

3.1. Divide Minority Clusters. The AGNES-SMOTE algorithm is proposed in this paper to refine SMOTE and its improved algorithm. The newly proposed algorithm filters noise samples first, uses AGNES to cluster samples, and divides datasets into subclusters. In the clustering process, this paper uses the average distance method to calculate the distance between two subclusters. Merge the two closest subclusters to form a new subcluster. Reduce the size of the subclusters by one. Then, continue to merge the two closest subclusters. Stop clustering until the distance between the subclusters exceeds the set valve-value. To avoid generating overlapping samples, majority samples distribution needs to be considered.

Before clustering minority samples with AGNES, cluster majority samples first to get majority subclusters set. The subclusters in the set represent the majority class. Then, judge the distance between the majority class and minority class. If the distances between majority samples and any two minority subclusters are less than the minimum distances between two minority subclusters, the merged minority subclusters will produce overlapping samples and the two minority subclusters should not be merged. The specific steps to classify clusters are listed as follows:

Step 1. Given the original dataset I, use K-nearest neighbor to filter noise samples in dataset I. Set $K = 5$ to traverse samples in dataset I. If more than 4/5 sample classes of K-nearest neighbors in dataset I are opposite to the selected sample class, judge the selected sample as noise sample and eliminate it. The remaining samples constitute the sample set $I\prime$.

Step 2. Cluster majority samples in I', treat each sample as an independent subcluster, use formula (3) to calculate the distance between the subclusters, merge the two closest subclusters, repeat the above procedures until the distance reaches the preset valve-value T_h, and obtain some majority subcluster sets $C^{\text{maj}} = \{C_1^{\text{maj}}, C_2^{\text{maj}}, \ldots, C_n^{\text{maj}}\}$:

$$d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{p \in C_i} \sum_{q \in C_j} |p - q|, \tag{3}$$

In this formula, p and q are samples in subclusters C_i and C_j, respectively. $|C_i|$ and $|C_j|$ represent their sample sizes.

Step 3. Divide minority samples according to the obtained majority subcluster set C^{maj}; treat each minority sample as a separate subcluster, and obtain minority subcluster set: $C^{\text{min}} = \{C_1^{\text{min}}, C_2^{\text{min}}, \ldots, C_m^{\text{min}}\}$.

Step 4. Calculate the distance between two minority subclusters with formula (3), make $D_{\text{min}} = d_{\text{avg}}(C_i, C_j)$, and record the minimum distance D_{min} and its corresponding subcluster numbers i and j.

Step 5. Traverse the majority subclusters in the set C^{maj}. If there is a majority subcluster C_k^{maj} and the distances from it to minority subclusters C_i^{min} and C_j^{min} are both less than the distance between the two minority subclusters, the minority subclusters C_i^{min} and C_j^{min} will not be merged, and the minimum distance D_{min} will be set large to avoid being considered when remerging. Otherwise, if the minority subclusters C_i^{min} and C_j^{min} are merged into a new minority subcluster C_m^{min}, the size of minority subclusters will be reduced by one.

Step 6. When new minority subcluster C_n^{maj} is merged, recalculate the distance between C_n^{maj} in minority subcluster set C^{min} and the remaining minority subclusters with formula (3). Repeat Step 3 to Step 5 until the distance between the nearest minority subclusters exceeds the set valve-value T_h; then, stop merging minority subclusters. Get the final minority subcluster set $C_{\text{new}}^{\text{min}} = \{C_1^{\text{min}}, C_2^{\text{min}}, \ldots, C_K^{\text{min}}\}$.

The valve-value is the key condition for merging subcluster. For better estimating valve-value T_h, define a value d_{avg} first:

$$d_{\text{avg}} = \frac{1}{|C_i^{\text{min}}|} \sum_{x_a \in C_i^{\text{min}}} \underset{a \neq b, x_b \in C_i^{\text{min}}}{\text{medium}} \text{dist}(x_a, x_b). \tag{4}$$

In the formula, x_a and x_b are samples in minority subcluster C_i^{min}, and $|C_i^{\text{min}}|$ is the sample size of this subcluster. Suppose d represents the median distance between a sample in minority subcluster and the rest of the samples. d_{avg} represents the average of these median distances. Taking the average of the median distance as the reference value can avoid noise samples interference. Redefine the valve-value T_h as follows:

$$T_h = d_{\text{avg}} \times f. \tag{5}$$

Parameter f is the distance adjustment factor, which can adjust valve-value T_h. The value range of parameter f will be discussed later.

3.2. Determine Sampling Weight and Probability Distribution. In classification tasks, the imbalances of within-class samples and between-class samples will affect model performance. The density of each subcluster varies with its sample size. The sampling weight of each minority subcluster is determined by its denseness. Set small weight for dense subcluster and large weight for sparse subcluster to avoid overfitting. Thus, sampling weights assigned to minority subclusters vary with their sizes, denoted as W_i:

$$W_i = 1 - \frac{\text{num}_i}{\sum_{i=1}^{N} \text{num}_i}. \tag{6}$$

N represents the size of minority subclusters and num_i represents the sample size of $i^{\text{-th}}$ minority subcluster. From formula (6), it is known that the larger the sample size in a

certain minority subcluster is, the larger the proportion of the sample sizes in the total minority subclusters is and the smaller W_i will be; namely, the assigned weight and synthetic samples size both become smaller, eventually achieving balanced sample distribution in the same class.

As shown in formula (7), the sampling size num_i of each minority subcluster can be determined by W_i (sampling weights of each subcluster) and $N_{\text{maj}} - N_{\text{min}}$ (the difference between the sizes of majority sample and minority sample after excluding noise samples):

$$\text{num}_i = \left(N_{\text{maj}} - N_{\text{min}}\right) * W_i. \tag{7}$$

In addition, when classifying samples, the minority samples closer to the decision boundary are more prone to be misclassified, increasing the learning difficulty of minority samples. Therefore, it is necessary to select samples for oversampling. To ensure the quality of synthetic samples, the probability distribution of minority subclusters is introduced to select "seed samples" from minority samples with important but difficult information. The probability of each sample being selected is set as D_i:

$$D_i = \frac{1}{\sum_{b=1}^{k} d_{xy_b}}. \tag{8}$$

The probability distribution of minority subclusters is

$$P_i = \frac{D_i}{\sum_{i=1}^{n} D_i}. \tag{9}$$

In this equation, y_b is x's b^{th} majority sample's neighbor. $1 \le b \le k$. d_{xy_b} denotes the Euclidean distance between sample x in minority subcluster and majority sample y_b. i represents one sample in minority subcluster, n is the sample size of a certain minority subcluster, and k signifies neighbors' size. It can be reckoned from the formula that the probability of each sample being selected is determined by the distance between this sample and majority class boundary; the probability of minority samples closer to the majority class boundary being selected is higher than that of samples far away; the probability of each sample being selected constitutes the probability distribution of minority subclusters. In this way, the distribution characteristics of samples are considered and the minority class decision boundaries are extended effectively.

3.3. Restrict the Generation of New Samples in a Certain Area. Determine the synthetic samples size of each minority subcluster, and select the "seed sample" according to the probability distribution of each minority subcluster. Consider the generation of new samples in a certain area to improve classifier performance and prevent synthetic samples from being distributed marginally. Therefore, when synthesizing samples, the new generated samples distribution needs to be taken into account. Select a sample from "seed samples," randomly select two neighboring minority samples from the subcluster where the selected "seed sample" is located, and form a triangle with the three selected samples as vertexes. Synthesize new samples on the line from the three vertexes to the centroid, respectively. One triangle generates three new synthetic samples. The centroid method is adopted to restrict the generation of new samples in a certain area. Set three samples distribution as X_1, X_2, and X_3; their centroid X_T can be calculated by the following formula:

$$X_T = \left(\frac{1}{3}\sum_{i=1}^{3} X_i, \frac{1}{3}\sum_{i=1}^{3} Y_i\right), \tag{10}$$

where X_i represents the horizontal coordinates of three vertexes and Y_i represents the vertical coordinates of three vertexes. This method makes the new samples move closer to the centroid, which addresses the issue of the marginal distribution of new samples caused by SMOTE. When synthesizing new samples, restrict the generation of new samples in a certain area. Those synthetic samples will move closer to the centroid.

3.4. AGNES-SMOTE Algorithm. The procedures of the AGNES-SMOTE algorithm are depicted below. Use K-nearest neighbor to filter noise samples in the original dataset. Adopt the AGNES algorithm to cluster majority samples and divide them into several majority subclusters. Cluster minority samples and merge the two closest subclusters on the basis of the obtained majority subclusters and keep clustering until the distance between two minority subclusters exceeds the set valve-value; then obtain minority subclusters. Assign weight to each minority subcluster and calculate the probability distribution of each minority subcluster, and combine the two to oversample samples in minority subcluster. Restrict the generation area of synthetic samples by the centroid method. The detailed Algorithm 1 is as follows:

(1) Delete noise samples from original datasets to obtain ClearData datasets, and then split ClearData into majority sample group and minority sample group. Use AGNES to cluster majority sample group to obtain majority subclusters. Then, cluster minority sample group. When clustering, determine whether there exist majority samples between the two nearest minority subclusters. If no majority samples exist, merge minority subclusters (line 1 to line 10).

(2) Calculate sample size of the obtained minority subcluster, assign sampling weight to minority subcluster, calculate the size of samples needed to be synthesized, and then calculate the probability distribution of each minority subcluster (lines 15 to 23).

(3) Finally, in each minority subcluster, select "seed samples" based on the size and the probability distribution of samples needed to be synthesized. Select a sample from all "seed samples," randomly select two neighboring minority samples from the subcluster where the selected "seed sample" is located, form a triangle with the three selected samples as vertexes, and synthesize new samples on the line

Input: dataset Data, distance threshold T_h
Output: dataset sample
(1) ClearData = Noise_Delete (Data);
(2) [majority, minority] = splite (ClearData);
(3) Cmaj = agg_cluster (majority, T_h);
(4) While pp_min < T_h
(5) p_dist = pdist (Cmin);
(6) [pp_min, p1, p2] = min_pp (p_dist);
(7) if (pp_min < minds (p1, Cmaj) and pp_min < minds (p2, Cmaj))
(8) merge(p1, p2);
(9) end if
(10) end while
(11) for i = 1:size (Cmin)
(12) num[i] = unique (Cmin[i])
(13) $W_i = 1 - (\text{num}_i / \sum_{i=1}^{\text{size}(C\,\text{min})} \text{num}_i)$
(14) end for
(15) for i = 1: size (Cmin)
(16) nsample = dist_NB (Cmin[i], 5)
(17) for $a_j \in C\,\text{min}[i]$
(18) d_j = sum (a_j, nsample))
(19) $d = d \cup d_j$
(20) $D_j = 1/d$
(21) end for
(22) $P_i = D_j / \sum_{i=1}^{\text{size}(C\,\text{min})} D_j$
(23) num[i] = (majority–minority) ∗ W[i]
(24) seed = seed ∪ sample (P_i, num[i], Cmin[i])
(25) while j < seed
(26) s = sample (seed, 1)
(27) [ns1, ns2] = sample (s, 5, 2, Cmin[i])
(28) $X_T = ((1/3)\sum_{i=1}^{3} X_i, (1/3)\sum_{i=1}^{3} Y_i)$
(29) snew = s + rand (0, 1) ∗ (X_T–s)
(30) ns1new = ns1 + rand (0, 1) ∗ (X_T – ns1)
(31) ns2new = ns2 + rand (0, 1) ∗ (X_T – ns2)
(32) $j = j + 1$
(33) end while
(34) sample = sample ∪ snew ∪ ns1new ∪ ns2new
(35) end for
(36) return sample

Algorithm 1: AGNES-SMOTE.

from the three samples to the centroid, respectively. Then, add new synthetic samples to the synthetic samples group (lines 24 to 36).

4. Experimental Design and Result Analysis

4.1. Evaluation Index. The conventional classification algorithms use the confusion matrix to perform the evaluation, as shown in Table 1 [17]. In this paper, the minority class is defined as a positive class, and the majority class is a negative class. In the confusion matrix, TN (True Negatives) is the number of negative examples rightly classified, FP (False Positives) is the number of negative examples wrongly classified as positive, FN (False Negatives) is the number of positive examples wrongly classified as negative, and TP (True Positives) is the number of positive examples rightly classified [11].

The classifier uses precision and recall [18] as two basic indicators for classification, defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{11}$$

In processing imbalanced data, three commonly used indicators, F-measure, G-mean, and AUC, are generally used to evaluate the performance of classification algorithms. F-measure is the harmonic mean of accuracy and recall, and β is set to be1 in the experiment. G-mean combines the accuracy of the classifier on majority

sample and minority sample. AUC represents the sum of the areas under the ROC curve. N and M, respectively, represent the size of minority samples and majority samples in datasets. F-measure, G-mean, and AUC are defined as follows [19]:

$$F - \text{measure} = \frac{\left(1 + \beta^2\right) * \text{Recall} * \text{Precision}}{\beta^2 * \text{Recall} + \text{Precision}},$$

$$G - \text{mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TN}}{\text{TN} + \text{FP}}}, \tag{12}$$

$$\text{AUC} = 1 - \frac{(\text{FN}/N) + (\text{FN}/M)}{2}.$$

4.2. Experimental Analysis

4.2.1. Datasets. In this paper, nine UCI datasets groups [20] are selected for the experiment, whose structures are listed in Table 2.

The hierarchical random division is adopted in this paper to ensure the consistent imbalance ratio of samples in the training set and test set. 50% cross-validation is used as an evaluation method. Each dataset is divided into 10 parts. Select one part as verification set in turn and the remaining nine parts as the training set. Obtain the average of 10 results. The parameters of the SVM classifier are set as follows: the kernel function is a Gaussian radial basis and the penalty factor C is 10.

4.2.2. Determine Parameters f. The performance of the AGNES-SMOTE algorithm is affected by the parameters to some extent. The distance adjustment factor f is used to control subcluster merging when clustering. If f value is too small, the size of minority subclusters will be too large while the size of samples in each subcluster will be too small, which reduce the diversity of synthetic samples and cause overfitting. If f value is too large, merged clusters will contain majority samples, resulting in overlapping when synthesizing.

As shown in Table 3, five datasets are used as test datasets to determine the range of parameter f. $f = 1.0$ indicates there is no need to adjust the valve-value. Then, $f = 1.0$ is used as the axis to select f values. After testing, the results show that when $f = 1.0$, 3 datasets obtain maximum F-measure value; when $f = 0.6$, 1 dataset obtains maximum F-measure value; and when $f = 1.5$, 1 dataset obtains maximum F-measure value. Therefore, the reference range of parameter f should be between 0.3 and 1.5. When $f > 2.5$, F-measure values will be similar because when parameter f becomes larger, all subclusters will eventually merge into one.

4.2.3. Experimental Results and Analysis

(1) Analysis of synthetic data distribution results: this paper uses artificial datasets to verify and compare synthetic samples distribution of the new proposedly algorithm and SMOTE. The results are shown in the following figures, in which the red dots represent majority samples and the black crosses represent minority samples and their synthetic samples. Compared with Figure 1, the synthetic samples sampled by the SMOTE algorithm are more distributed in the edge area and even mixed into majority samples which cause overlapping. As the new synthetic samples are highly similar and repeated, within-class imbalance in original dataset has not been improved. In view of the shortcomings in Figure 2, AGNES-SMOTE effectively filters noise samples; when clustering, divide minority subclusters in consideration of majority samples distribution to avoid new synthetic samples mixing into majority sample area and reduce noise impact. Assign sampling weights to minority subclusters to achieve within-class balance of minority subcluster. Sample more marginal samples susceptible to be misclassified on the basis of the probability distribution to form an obvious boundary between two sample classes. For samples distributed marginally, the centroid method is used to restrict the generation of new samples in a certain area, which further guarantees the quality and diversity of synthetic samples. The data distribution is shown in Figure 3.

(2) Analysis of actual dataset results: compare AGNES−SMOTE with SMOTE, K-means-SMOTE, and Cluster-SMOTE in the experiments. The AUC values of the above sampling algorithms on datasets are shown in Table 4.

Table 1: Confusion matrix.

Classification	Predicted negative	Predicted positive
Actual negative	TN	FP
Actual positive	FN	TP

The experimental results in Table 4 indicate that AGNES-SMOTE has better AUC values on datasets Ecoli, Libra, Yeast1, Optical_digits, and Abalone than other sampling algorithms. Besides, AGNES-SMOTE has large AUC values on datasets Libra and Optical_digits because of their large imbalance ratios and rich features; thus more samples are needed to be synthesized. AGNES-SMOTE considers within-class imbalance, selects samples, restricts the generation area of synthetic samples, and reduces the overlap of synthetic samples to ensure synthetic samples quality and provide various information for the classifier. AGNES-SMOTE has low AUC values on datasets Haberman, Yeast1, and Liver due to their smaller imbalance ratio and fewer features.

The F-measure values and G-mean values with SMOTE, K-means-SMOTE, Cluster-Smote, and AGNES-SMOTE on each dataset are listed in Tables 5 and 6.

Tables 5 and 6 indicate that the AGNES-SMOTE algorithm attains good F-measure values and G-mean values on most datasets. It greatly improves F-measure values and G-mean values on datasets Ecoli, Yeast1,

Table 2: Imbalanced data table.

Dataset	Instances	Features	Minority instances	Majority instances	Ratio
Ecoli	336	7	35	301	1 : 8.6
Libra	360	90	24	336	1 : 14
Yeast1	1484	10	429	1055	1 : 2.46
Haberman	306	3	81	225	1 : 2.78
Satimage	6435	36	626	5809	1 : 9.28
Optical digits	5620	64	554	5066	1 : 9.14
Abalone	4177	10	391	3786	1 : 9.68
Liver	345	6	145	200	1 : 1.38
LEV	1000	4	93	907	1 : 9.75

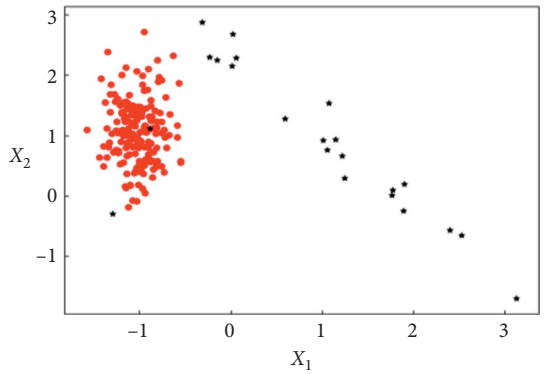

Figure 1: Original data distribution.

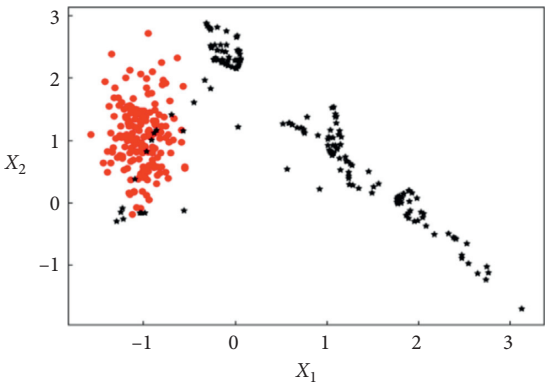

Figure 2: Data distribution after SMOTE sampling.

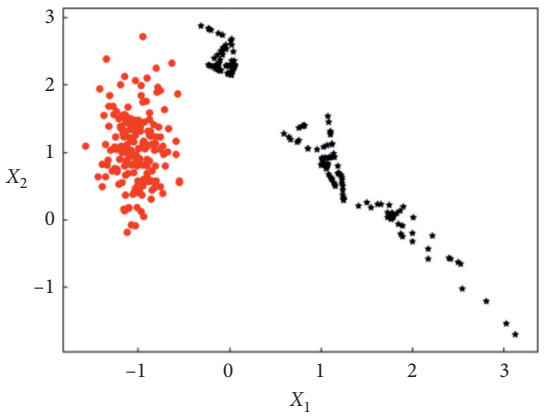

Figure 3: Data distribution after AGNES–SMOTE sampling.

Table 3: F-measure values with different f values.

F	Ecoli	Libra	Yeast1	Optical_digits	Liver
0.3	0.6444	0.5882	0.5813	0.9581	0.6082
0.6	0.6444	0.6250	0.5886	0.9641	0.6147
1.0	0.6517	0.6531	0.5935	0.9661	0.6044
1.5	0.6534	0.6517	0.5892	0.9645	0.6044
2.0	0.6547	0.6517	0.5892	0.9645	0.6044
2.5	0.6547	0.6403	0.5892	0.9627	0.6044
3.0	0.6547	0.6403	0.5892	0.9627	0.6044
8.0	0.6547	0.6403	0.5892	0.9627	0.6044

Haberman, Optical_digits, Abalone, and LEV, among which F-measure highest value reaches 96.70% and G-mean highest value reaches 97.53%. On dataset Libra,

Table 4: AUC values on each dataset with different algorithms.

Dataset	SMOTE	K-means-SMOTE	Cluster-SMOTE	AGNES-SMOTE
Ecoli	0.9408	0.9432	0.9416	0.9444
Libra	0.9167	0.9330	0.9142	0.9395
Yeast1	0.7715	0.7722	0.7775	0.7795
Haberman	0.6905	0.7219	0.6706	0.6984
Satimage	0.9302	0.9203	0.9204	0.9209
Optical_digits	0.9979	0.9959	0.9961	0.9980
Abalone	0.8544	0.7838	0.8463	0.8549
Liver	0.7274	0.6976	0.6719	0.7192
LEV	0.7503	0.8698	0.8841	0.8703

Table 5: *F*-measure values on each dataset with different algorithms.

Dataset	SMOTE	*K*-means-SMOTE	Cluster-SMOTE	AGNES-SMOTE
Ecoli	0.6237	0.6374	0.6154	0.6444
Libra	0.5926	0.5106	0.6429	0.6531
Yeast1	0.5820	0.5492	0.5828	0.5888
Haberman	0.4719	0.4361	0.4327	0.4746
Satimage	0.5812	0.5661	0.5458	0.5654
Optical_digits	0.9613	0.9661	0.9601	0.9670
Abalone	0.3945	0.0150	0.3924	0.4013
Liver	0.6075	0.5967	0.5920	0.6044
LEV	0.5103	0.4153	0.4593	0.5534

Table 6: G-mean values on each dataset with different algorithms.

Dataset	SMOTE	K-means-SMOTE	Cluster-SMOTE	AGNES-SMOTE
Ecoli	0.8653	0.8685	0.8518	0.8701
Libra	0.7993	0.6954	0.8478	0.8055
Yeast1	0.7065	0.6596	0.7058	0.7121
Haberman	0.6259	0.5669	0.5909	0.6278
Satimage	0.8519	0.7412	0.8288	0.8396
Optical_digits	0.9689	0.9752	0.9655	0.9753
Abalone	0.7827	0.0875	0.7929	0.7955
Liver	0.6551	0.6407	0.5461	0.6593
LEV	0.7469	0.7920	0.7736	0.8046

G-mean value by AGNES-SMOTE improves greatly but is still slightly lower than that by Cluster-SMOTE; however, F-measure value by AGNES-SMOTE on dataset Libra increases by 14.25%. On dataset Satimage, F-measure value and G-mean value by AGNES-SMOTE are slightly lower than those by SMOTE, since this dataset has many overlapping data and interference samples affect classification performance. On dataset Liver, F-measure value and G-mean value by AGNES-SMOTE are similar to those by SMOTE algorithm because the data distribution in the original dataset is also relatively concentrated. Generally speaking, in dealing with imbalanced data, the AGNES-SMOTE algorithm improves classification performance through reducing noise interference, reducing synthetic samples overlap, selecting marginal samples susceptible to be misclassified, and considering within-class imbalance and generated samples distribution.

5. Conclusion

Regarding imbalanced datasets classification, the existing oversampling algorithms mainly deal with between-class imbalance and neglect within-class imbalance. Some problems are ignored, such as samples being oversampled are not selected, noise is not removed, synthetic samples will overlap, and samples will be distributed "marginally." To solve the abovementioned problems, an oversampling algorithm—AGNES-SMOTE—is presented in this paper, which is based on the hierarchical clustering and improved SMOTE. This algorithm follows the following procedures: filter noise samples in the dataset; cluster majority samples and minority samples through the AGNES algorithm, respectively; divide minority subclusters in the light of the obtained majority subclusters; select samples for oversampling based on sampling weight and the probability

distribution of minority subclusters; restrict the generation of new samples in a certain area by the centroid method. Comparative experiments on data processing with different algorithms have been conducted. Experimental results indicate that AGNES-SMOTE improves the classification accuracy of minority samples and the overall classification performance. However, the new oversampling algorithm proposed in this paper is only available for biclass cases. In practice, most data fall into multiple categories, so optimized oversampling algorithms for multiclass data classification will be expected in the future.

Data Availability

The data used to support the results of this study are available on the website: https://archive.ics.uci.edu/ml/index.php.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by the Natural Science Foundation of Guangxi (2019GXNSFAA245053), the Guangxi Science and Technology Major Project (AA19254016), and the Major Research Plan Integration Project of NSFC (91836301).

References

[1] M. O. Zan, Y. Gai, and G. Fan, "Credit card fraud classification based on GAN-AdaBoost-DT imbalanced classification algorithm," *Journal of Computer Applications*, vol. 39, no. 2, pp. 618–622, 2019.

[2] Y. Li, L. I. U. Zhan-dong, and H.-J Zhang, "Review on ensemble algorithms for imbalanced data classification," *Journal of Computer Applications*, vol. 5, no. 31, pp. 1287–1291, 2014.

[3] Q. Li and Y. Mao, "A review of boosting methods for imbalanced data classification," *Pattern Analysis and Applications*, vol. 17, no. 4, pp. 679–693, 2014.

[4] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[5] I. Nekooeimehr and S. K. Lai-Yuen, "Adaptive semi-unsupervised weighted over-sampling (A-SUWO) for imbalanced datasets," *Expert Systems with Applications*, vol. 46, pp. 405–416, 2016.

[6] Q. Zhao, Y. Zhang, J. Ma et al., "Research on classification algorithm of imbalanced datasets based on improved SMOTE," *Computer Engineering and Applications*, vol. 54, no. 18, pp. 168–173, 2018.

[7] J. V. Hulse and T. Khoshgoftaar, "Knowledge discovery from imbalanced and noisy data," *Data & Knowledge Engineering*, vol. 68, no. 12, pp. 1513–1542, 2009.

[8] F. Cheng, J. Zhang, and C. Wen, "Cost-Sensitive Large margin Distribution Machine for classification of imbalanced data," *Pattern Recognition Letters*, vol. 80, pp. 107–112, 2016.

[9] J. Bian, X.-g. Peng, Y. Wang, and H. Zhang, "An efficient cost-sensitive feature selection using chaos genetic algorithm for class imbalance problem," *Mathematical Problems in Engineering*, vol. 2016, no. 6, 9 pages, Article ID 8752181, 2016.

[10] B. Tang and H. He, "GIR-based ensemble sampling approaches for imbalanced learning," *Pattern Recognition*, vol. 71, pp. 306–319, 2017.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," *Lecture Notes in Computer Science*, pp. 878–887, 2005.

[13] H. He, B. Yang, E. A. Garcia et al., "ADASYN: adaptive synthesis sampling approach for imbalanced learning," in *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, Hong Kong, China, June 2008.

[14] D. A. Cieslak, N. V. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets," in *Proceedings of the 2006 IEEE International Conference on Granular Computing*, pp. 732–737, IEEE, Atlanta, GA, USA, May 2006.

[15] D. Georgios, B. Fernando, and L. Felix, "Improving imbalanced learning through a heuristic over-sampling method based on k-means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, 2018.

[16] S. Barua, M. M. Islam, and K. Murase, "A novel synthetic minority oversampling Technique for imbalanced data set learning," *Neural Information Processing*, Springer, Berlin, Germany, pp. 735–744, 2011.

[17] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[18] S. Naganjaneyulu and M. R. Kuppa, "A novel framework for class imbalance learning using intelligent under-sampling," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 73–84, 2013.

[19] Y. Xu, Z. Yang, Y. Zhang, X. Pan, and L. Wang, "A maximum margin and minimum volume hyper-spheres machine with pinball loss for imbalanced data classification," *Knowledge-Based Systems*, vol. 95, pp. 75–85, 2016.

[20] UCI Machine Learning Repository [EB/OL]. http://archive.ics.uci.edu/ml/index.php.