Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(10): 17866–17885.
DOI: 10.3934/mbe.2023794
Received: 04 July 2023
Revised: 21 August 2023
Accepted: 21 August 2023
Published: 18 September 2023

Research article

A new Monte Carlo sampling method based on Gaussian Mixture Model for imbalanced data classification

Gang Chen*, Binjie Hou and Tiangang Lei

Department of Mathematics, Dalian Maritime University, Dalian 116026, China

* **Correspondence:** Email: chengang@dlmu.edu.cn.

Abstract: Imbalanced data classification has been a major topic in the machine learning community. Different approaches can be taken to solve the issue in recent years, and researchers have given a lot of attention to data level techniques and algorithm level. However, existing methods often generate samples in specific regions without considering the complexity of imbalanced distributions. This can lead to learning models overemphasizing certain difficult factors in the minority data. In this paper, a Monte Carlo sampling algorithm based on Gaussian Mixture Model (MCS-GMM) is proposed. In MCS-GMM, we utilize the Gaussian mixed model to fit the distribution of the imbalanced data and apply the Monte Carlo algorithm to generate new data. Then, in order to reduce the impact of data overlap, the three sigma rule is used to divide data into four types, and the weight of each minority class instance based on its neighbor and probability density function. Based on experiments conducted on Knowledge Extraction based on Evolutionary Learning datasets, our method has been proven to be effective and outperforms existing approaches such as Synthetic Minority Over-sampling TEchnique.

Keywords: imbalanced data; Monte Carlo sampling; probability density function; oversampling technique; Gaussian Mixture Model

1. Introduction

In recent years, imbalanced data classification has received extensive attention from scholars and fostered many real-world applications, such as fraud detection [1], medical diagnosis [2], email filtering [3] and so on. Generally speaking, binary classification problems are relatively common in data classification. In an imbalanced dataset, the majority class refers to the class with the highest number of instances, while the minority class refers to the class with the lowest number of instances. In cases where there is an imbalance between the majority class and the minority class, traditional classifiers often exhibit bias towards the majority class, resulting in suboptimal performance in classifying the minority class [4]. Therefore, it is crucial to improve the accuracy of classifying the

minority class.

In order to address this problem, several scholars have proposed various methods, which can be divided into two aspects: data level and algorithm level. At the data level, two main techniques are commonly adopted: oversampling and undersampling. Undersampling involves reducing the size of the majority class, while oversampling aims to increase the size of the minority class. However, these techniques have limitations in preserving the inherent structure of the dataset and may discard informative instances. Innovative techniques, such as Monte Carlo sampling, leverage random sampling to create balanced datasets while retaining distributional characteristics [5]. Recent studies Gaussian Mixture Undersampling (GMUS) selected the maximum of probability density function for the minority class as the cross-edge of two classes and undersampled the majority samples near the cross-edge, enhancing the quality of the undersampled datasets [6]. Notably, Monte Carlo sampling algorithm based on Gaussian Mixture Model (MCS-GMM) is different from GMUS. Because GMUS is a undersampling technique and MCS-GMM is a oversampling technique. Yan et al. [7] proposed a spatial distribution based undersampling (SDUS) method to solve the class imbalance problem. SDUS used a supervised construction process to learn majority-class local patterns based on spherical neighborhood (SPN). In order to preserve the distribution pattern of the original data, two strategies, top-down and bottom-up, were proposed to select the majority class sample subset. Sdus introduced the integration technology to improve the learning performance by utilizing the diversity caused by the randomness of the local pattern learning process. Zhu et al. [8] proposed a new undersampling scheme called Noisy-Sample-Removed Undersampling Scheme (NUS) for imbalanced classification. NUS first clusters the majority class samples and builds a hypersphere around each cluster's center. It then determines whether each minority sample is within the hypersphere or not, and removes noisy samples from both the majority and minority classes. NUS was integrated with three basic classifiers.

Oversampling technique, a method are widely used for imbalanced data classification. Synthetic Minority Over-sampling TEchnique (SMOTE) [9] was a very well-known method based on k-nearest neighbors in which the new minority class instances are generated along the line segments joining any or all of the k-nearest neighbors. However, the SMOTE algorithm does not consider the impact of surrounding data on minority class instances [10]. To this end, some variants of SMOTE were proposed. He et al. [11] proposed a adaptive synthetic sampling approach for imbalanced data (ADASYN). This approach calculates the weight of each minority class instance based on the number of majority class instances in its k-nearest neighbors. Han et al. [12] proposed a Borderline-SMOTE in which the number of majority class instances and minority class instance in its k-nearest neighbors is adopted to determine the sampling boundary. Douzas et al. [13] proposed a Kmeans-SMOTE, its method is to generate new samples by By dividing the minority class region. Yan et al. [14] oversampled the minority class using the local-density (LDAS) of each minority class instance. Xie et al. [15] used the local distribution (GDO) of each minority class instance. In GDO, the Gaussian distribution was determined and the sampling weight takes into account the influence of surrounding data. Bhagwani et al. [16] trained the datasets to generate new samples by the generative adversarial model. Maldonado et al. [17] proposed a feature-weighted oversampling approach for imbalanced classification (FW-SMOTE). Kaya et al. [18] proposed a differential evolution based oversampling approach to solve imbalanced data problems. Xie et al. [19] proposed an improved oversampling algorithm Based on the Samples' Selection Strategy to select samples with high information value, which has an effective candidate individual generation mechanism. Peng et al. [20]

proposed a novel approach for imbalanced data classification using data gravitation, which can strengthen and weaken the gravitational field of the minority and majority classes. Rahmati et al. [21] proposed proposed a method called gravitational density-based mass sharing to improve the data quality. Koziarski et al. [22] proposed a method for oversampling imbalanced data in noisy environments by identifying regions of interest for the minority class using a Radial-Based approach. Bunkhumpornpat et al. [23] proposed a Safe-Level-SMOTE, which utilizes more information about surrounding neighbors. Sun et al. [24] proposed a robust oversampling method called disjuncts-robust oversampling (DROS) method. This novel approach shows that filling minority-class regions in data space with new synthetic samples can be viewed as a search light illuminating real-life constrained regions with light cones. In the first step, DROS computes a sequence of light cone structures, first starting from the inner minority region, then passing through the boundary minority region, and finally stopped by the majority region. In a second step, DROS generates new synthetic samples within these light cone structures.

On the other hand, there are many ways to deal with such a problem at the algorithmic level. Yin et al. [25] proposed a fault detection method that utilizes robust one-class support vector machines. The algorithm's robustness can be enhanced by adjusting the weight of samples, which can also improve the ability of a single-class support vector machine to handle uncorrelated samples. Scholkopf et al. [26] proposed One-Class support vector machines (OCSVM), this method creates a decision boundary between two classes called a hyperplane. UnderBagging [27] was a technique that involves random undersampling of the data set in each Bagging iteration, while retaining all the minority class instances in each iteration. To effectively train a model without synthesizing minority classes of data, it is important to make the most of the majority class data during the training process. Li [28] proposed a method that combines under-sampling with the classical 'Bagging' ensemble method. Hido et al. [29] proposed an alternative approach to the Bagging method for imbalanced datasets.

The above oversampling methods in data level discussed in this study aim to identify specific types of data and generate new examples to improve classifier performance, and each method has its own advantages depending on different scenarios. However, most oversampling methods prioritize the generation of new examples in specific regions, which may lead to possible overfitting [30] and neglect the probability distribution of minority class. Meanwhile, recent studies suggest that class overlap [31] can pose a challenge for traditional classifiers in distinguishing two classes. The method is based on Generative Adversarial Networks (GANs) [16] guarantee the probability distribution of the minority class and generate new data. Although numerous strengths exist, there are also some shortcomings in the study. For example, the new data generated by GANs ignore surrounding majority class samples, which will cause overlap of samples between the minority class and majority class. For MCS-GMM, it will reduce the degree of overlap between the minority and majority class and guarantee the distribution of the minority class to a certain extent. But MCS-GMM corrupts the probabilistic nature of the imbalanced data.

To mitigate these problems, we propose a new oversampling algorithm (MCS-GMM) for imbalanced data, which pay more attention to the impact of data distribution and the overlap of minority class data. Besides, to a certain extent, MCS-GMM can consider the probability distribution of minority class and take full advantage of probability information.

This paper is divided into three parts:

1) In order to obtain statistical information, finding the optimal probability density function to describe majority class and minority class by Gaussian Mixture Model (GMM). Besides, the probability density is used when determining the sampling weight to set the number of samples that more suitable for minority class.

2) Specifically, the overlap between majority class and minority class may directly affect classification performance. So, the three sigma rule [32] is adopted to divide the degree of overlap. In this way, the impact of class overlap on data classification will be further reduced.

3) Data generated by oversampling techniques often gathers together. To ensure the diversity of the generated data, the variant of Monte Carlo Sampling is adopted to oversample the minority class.

The structure of this paper is as follows: Section 2 describes a detailed description of the proposed MCS-GMM algorithm, Section 3 presents the experimental results, Section 4 presents the Time complexity of the proposed algorithm and Section 5 concludes the paper and offers future research directions.

2. Proposed method

The proposed MCS-GMM's basic idea may be divided into four steps: 1) Probability Density Estimation for imbalanced data, 2) Monte Carlo sampling, 3) The oversampling technique of minority class, 4) An adaptive weight of minority class. The subsections that follow will describe each component's specific method in detail.

2.1. GMM

With the continuous development of research, GMM can approximate arbitrary probability distribution, thus it is considered as a dominant tool for classification in such domains [33]. Consequently, more and more researchers are interested in this method. In this paper, we use $k(k = 2)$ Gaussian distributions $f(x|\ \mu_{maj}, Cov_{maj})$ and $f(x|\ \mu_{min}, Cov_{min})$ to fit the distribution of T_{maj} and T_{min}. As shown in Figure 1, blue represents the distribution of the majority class, green represents the distribution of the minority class, we use two Gaussian distributions to describe the imbalanced data distribution. According to GMM, the distinct Gaussian distributions with various weights. However, the majority class tend to have larger Gaussian weight and the Gaussian weight of minority class is smaller.

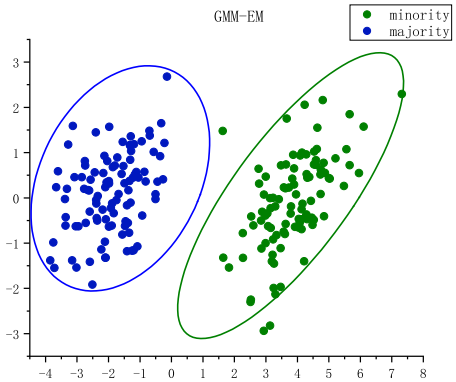

Figure 1. GMM-Expectation-maximization (EM).

To describe the data for a class in a given datasets, we use the formula below in binary classification problems.

$$p(x|\,\alpha,\mu,Cov) = \sum_{i=1}^{h} \alpha_i f(x|\,\mu_i, Cov_i) \tag{1}$$

where x is an instance in T, μ_i is the mean vector of the i-th Gaussian distribution model, Cov_i is the covariance vector of the i-th Gaussian distribution model, and $f(x|\,\mu_i, Cov_i)$ is the i-th component in the mixed model, which is defined as follows:

$$f(x|\,\mu_i, Cov_i) = \frac{1}{\sqrt{2\pi}|Cov_i|} exp\left[\frac{1}{2}(x-\mu_i)^T Cov_i (x-\mu_i)\right] \tag{2}$$

α_i is the weight of each Gaussian distribution and satisfies the following equation:

$$\sum_{i}^{h} \alpha_i = 1 \tag{3}$$

The GMM [34] needs to be estimated for three unknown parameters, namely the variance, mean and the weight of each Gaussian distribution. In our study, we intend to use the EM algorithm to optimize these parameters, which includes two steps. First, the parameters are initialized. The initial values of the mean (μ) and variance (Cov) in the GMM are calculated from the mean and variance of minority set, and α is initialized to $1/k$. Then, the probability function is maximized using the values calculated in the first step. Therefore, the maximum likelihood function becomes very important. To estimate the parameters in GMM, the maximum likelihood functions should be taked the expectation.

$$Q(\theta) = \sum_{k=1}^{h}\sum_{i=1}^{N} \gamma(Z_i^k) log(\alpha_k) + \sum_{k=1}^{h}\sum_{i=1}^{N} \gamma(Z_i^k) log(f(x_i|\,\theta_k)) \tag{4}$$

$$\gamma(Z_i^k) = \frac{\alpha_i f(x_i|\,\mu_k, Cov_k)}{\sum_{k=1}^{h} \alpha_k f(x_i|\,\mu_k, Cov_k)} \tag{5}$$

where $\gamma(Z_i^k)$ is the posterior probability of x_i belonging to the k-th Gaussian distribution. It is presented to simplify the EM algorithm's use in estimating GMM parameters. θ is the three unknown parameters for GMM. $f(x_i|\,\theta_k)$ is the probability density value of x_i belonging to the k-th Gaussian distribution.

By performing logarithmic transformation and derivation on Eq (4), the solution formulas of μ, Cov and α for the derivation are as follows:

$$\alpha_k^{i+1} = \frac{1}{N}\sum_{i=1}^{N} \gamma(Z_i^k) \tag{6}$$

$$\mu_k^{i+1} = \frac{1}{\sum_{i=1}^{N} \gamma(Z_i^k)}\sum_{i=1}^{N} x_i \gamma(Z_i^k) \tag{7}$$

$$Cov_k^{i+1} = \frac{1}{\sum_{i=1}^{N} \gamma(Z_i^k)}\sum_{i=1}^{N} \gamma(Z_i^k)(x_i - \mu_k^{i+1})(x_i - \mu_k^{i+1})^T \tag{8}$$

2.2. Monte Carlo Sampling

Conventional Monte Carlo simulations are stochastic and sample based on the probability distribution. The method [35] samples from a uniform distribution and the random values are independently drawn from a uniform distribution on $[0, 1]$. At the same time, cumulative distribution

function plays a very important role [36] in Monte Carlo simulations. Combining the cumulative distribution function, the final value of the sample value can be obtained.

$$f(x|\,\mu,\sigma) = \frac{1}{(2\pi)^{1/2}\sigma} exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}) \tag{9}$$

$$F(y) = p(Y \le y) = \int_{-\infty}^{y} f(x|\,\mu,\sigma)dx \tag{10}$$

$$rand = u^i \sim uniform(0,1) \tag{11}$$

$$s = F^{-1}(rand) \tag{12}$$

where $f(x|\,\mu,\sigma)$ is the probability density function of minority set in one-dimensional space, $F^{-1}(y)$ is the inverse cumulative distribution function of minority set in one-dimensional space and s is the sampling sample. The pseudocode of Monte Carlo Sampling algorithm is given in algorithm 1.

Algorithm 1: Monte Carlo sampling

Input: input parameters: μ, σ, r = [], s = []
Output: Sample set s

```
Initialize f(x|μ,σ), F(y) ;
for u^i ~ uniform(0,1) do
    r ← u^i;
end
 for rand in r do
    s ← F^{-1}(rand);
end
```

2.3. Dividing minority data and generating new minority data

The imbalanced data distribution have an impact on the performance of detection and classification, leading to classifier results that are biased toward the majority class. The performance of single classifier is poor when there are count overlaps between minority class and majority class. For example, for many variants of SMOTE, it will be less effective for the datasets with high degree of overlap. Therefore, it becomes extremely important for the newly generated dataset to prevent overlap of samples between the minority class and the majority class.

For a binary classification problem, let T be a given dataset composed of T_{maj} and T_{min}, that is $T = T_{maj} \cup T_{min}$, where T_{maj} and T_{min} represent the majority class and minority class. Respectively, N is the number of examples in T, N_{maj} is the number of examples in T_{maj} and N_{min} is the number of examples in T_{min}. At the same time, μ_{maj} is the mean of majority class and calculated by GMM, σ_{maj} is the standard deviation of majority class and calculated by GMM. μ_{min} and σ_{min} are the relevant parameters for minority class.

In order to divide the degree of overlap, we introduce three sigma rule to divide overlap area. The probability density function of majority class are presented on the left, the probability density function

of minority set are presented on the right. For the majority class, the boundaries in majority clss data, denoted by B_{maj}, is defined as follows:

$$B_{maj} = \mu_{maj} + 2\sigma_{maj} \tag{13}$$

For the minority set, the boundaries in minority class data, denoted by B^l_{min} and B^r_{min}, is defined as follows:

$$B^l_{min} = \mu_{min} - 2\sigma_{min} \tag{14}$$

$$B^r_{min} = \mu_{min} + 2\sigma_{min} \tag{15}$$

We classify the minority class data into four categories: A, B, C and D, according to their degree of overlap. Figure 2 is a representation of the probability density function for majority and minority class. Among them, blue is probability density function of the majority class, green is probability density function of the minority class.

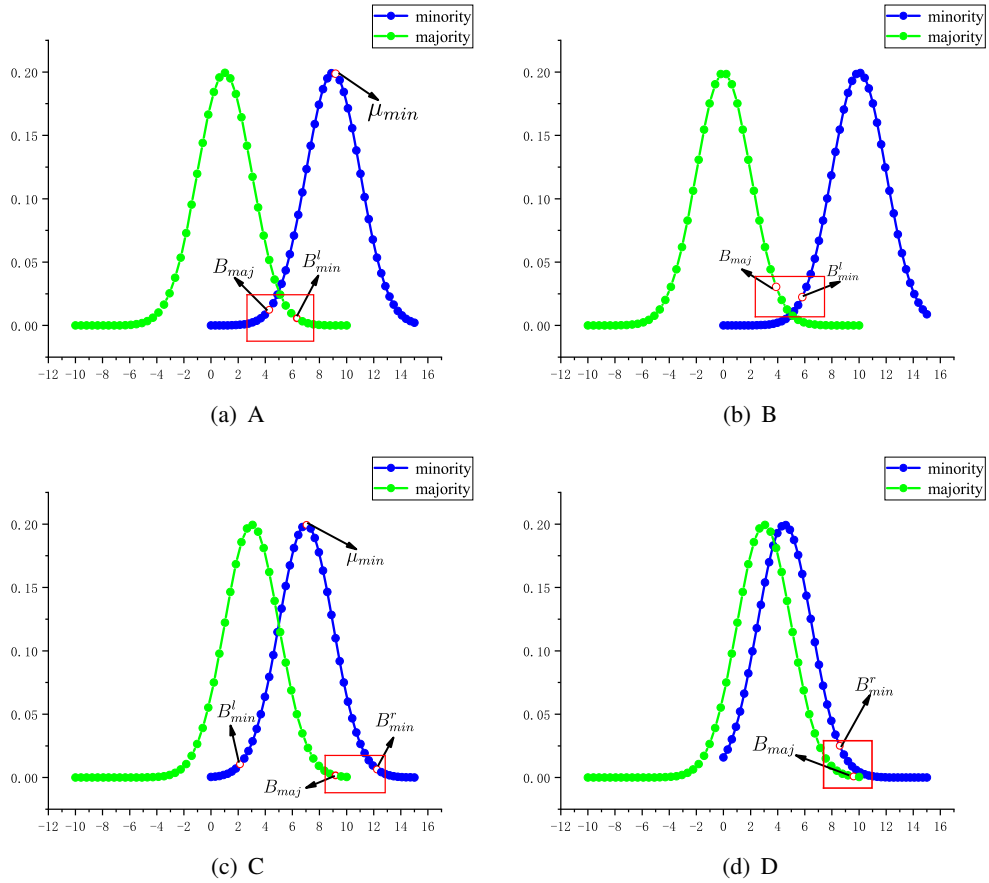

Figure 2. Different types of data.

1) Minority class data of category A must satisfie the condition $B_{maj} \leq B^l_{min}$, and shown in Eq (16). For category A data, we apply the SMOTE algorithm to oversample the minority class data and balance the skewed class distribution.

$$A = \left\{ B_{maj} \leq B^l_{min} \ \ \& \ \ x_i \in \mathrm{T}_{min} \right\} \tag{16}$$

2) Minority class data of category B must satisfie the conditions $B_{maj}>B^{l}_{min}$ and $B_{maj} \leq \mu_{min}$, and are shown in Eq (17). To improve the quality of generated data, we limit the range of sampled data in minority class data. In this regard, three significant situations are to be addressed, namely B1, B2 and B3, which are defined as follows:

$$\mathrm{B} = \left\{ B_{maj}>B^{l}_{min} \;\; \& \;\; B_{maj} \leq \mu_{min} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{17}$$

$$\mathrm{B1} = \left\{ x_i |\; x_i \leq B_{maj} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{18}$$

$$\mathrm{B2} = \left\{ x_i |\; x_i > B_{maj} \;\; \& \;\; x_i \leq \mu_{min} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{19}$$

$$\mathrm{B3} = \{ x_i |\; x_i > \mu_{min} \;\; \& \;\; x_i \in \mathrm{T}_{min} \} \tag{20}$$

where B1 is a data set, which contains the data that not exceed B_{maj} in minority class, B2 contains the data that not exceed μ_{min} and exceed B_{maj}, B3 contains the data that exceed μ_{min}.

For any data x_i in B1, we aim to find a symmetry point of it in a Gaussian distribution which is defined as x^*_i. Symmetry is utlized to reduce overlap between majority data when utilizing Monte Carlo Sampling to oversample the data. Subsequently, a random value r_1 is generated within the sampling interval $\left[0.5, F(x^*_i)\right]$ and the newly generated data is denoted as x_{new}.

$$x^*_i = 2\,\mu_{min} - x_i \tag{21}$$

$$r_1 = [0.5, F(x^*_i)] \tag{22}$$

$$x_n = F^{-1}(r_1) \tag{23}$$

It should be noted that the number of B1 should be chosen neither too small nor too large. To achieve this, we set the range of the number of B1 to be 0 to $N_{min}/5$. If the number falls outside this range, Eq (27) is applied to generate new data.

For any data x_i in B2, we adopt a different sampling interval for improved quality and enhanced diversity of generated data. To this end, we generate a random value r_2 within in the sampling interval $[F(x_i), 0.5]$. Different of those of B1 and B2, the sampling interval for x_i in B3 is set as $[0.5, F(x_i)]$ with random value r_3.

3) Minority class data of category C shall satisfy conditions $B_{maj}>\mu_{min}$ and $B_{maj} \leq B^{r}_{min}$, shown in Eq (24). In this case, the amount of majority class data exceeds half of the minority class data, which readily causes the generated data to overalp with the majority class. Hence, it is important to consider how to mitigate the extent of overlapping. Given the above considerations, we divide minority class data into two classes: C1 and C2. C1 contains data that do not exceed B_{maj} and the rest are in C2.

$$\mathrm{C} = \left\{ B_{maj}>\mu_{min} \;\; \& \;\; B_{maj} \leq B^{r}_{min} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{24}$$

$$\mathrm{C1} = \left\{ x_i |\; x_i \leq B_{maj} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{25}$$

$$\mathrm{C2} = \left\{ x_i |\; x_i > B_{maj} \;\; \& \;\; x_i \in \mathrm{T}_{min} \right\} \tag{26}$$

For x_i in C1, we employ a new method for data generation. For each minority class instance, its k-nearest (k=2) neighbors are identified, and interpolation is conducted between the minority class instance and each identified neighbor x^k_i to alleviate the class imbalance. Figure 3 describes new data x_n and is defined as follows:

$$x_n = x_i + rand(0, 1) \times (x_k - x_i) \tag{27}$$

Figure 3 visualizes the neighbors of a minority class instance and illustrates the sampled neighbor x_i^k of our method. In Figure 3(a), we can see that the minority class instance has two different neighbors: the minority neighbor and the majority neighbour. In this case, we generate new minority class data by interpolation with the minority class neighbor among the two neighbors via the k-nearest neighbors algorithm. In Figure 3(b), the neighbor that is closer to the minority class instance contains more essential information when both neighbors of the minority class instance are minority class neighbors. In Figure 3(c), when both neighbors are majority neighbors, our method selects the closer neighbor for the minority class instance to avoid new data that overlaps in the feature space with the majority class.

For x_i in C2, the sampling interval is set as $\left[F(B_{maj}), F(x_i)\right]$ for the generation of the random value as well as data using Eq (23).

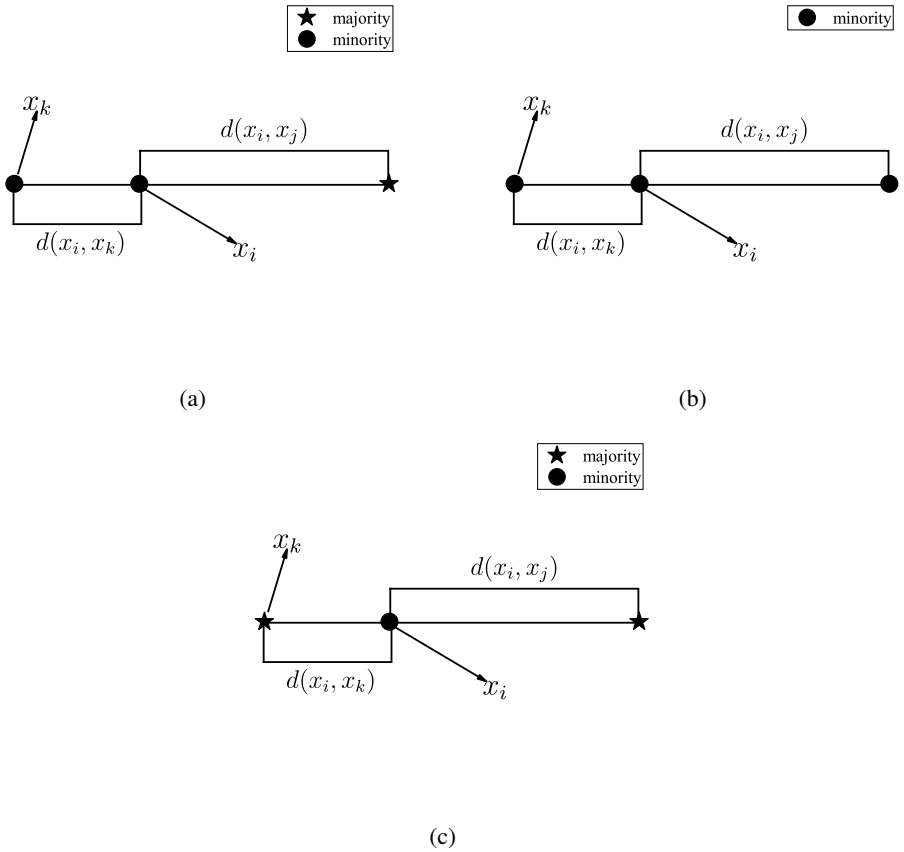

Figure 3. Different sample instances for each minority class instance.

4) For minority class data of category D, they must satisfy the condition $B_{maj} > B^r_{min}$, shown in Eq (28). Nevertheless, there exists a significant overlap between minority and majority classes. In this case, we generate data using Eq (27).

$$\mathrm{D} = \left\{ B_{maj} > B^r_{min} \ \ \& \ \ x_i \in \mathrm{T_{min}} \right\} \tag{28}$$

2.4. Weighting sampling instance

In previous sections, we divided minority class data into several types according to B_{maj} and generated minority class instances. More important, not all of the minority class instances will be equally important, and some might offer more insight into how to synthetic minority class instances than others. Hence, it is necessary to assign greater weights to instances that carry more important information. Existing weighting schemes in the literature are primarily based on the k-nearest neighbors of the majority and minority classes, which may be less efficient if the probability of certain minority class data is large. As such, we adopt a novel way of assigning geometric weights for the minority class distribution, exploiting the information contained in the minority class. Concretely, we assign the weights via:

$$p_i = f(x_i|\, \mu_{min}, \sum_{min}) \tag{29}$$

$$D_i = \frac{D_i^{min}}{D_i^{maj}} \tag{30}$$

$$W_i = p_i \cdot D_i \tag{31}$$

where p_i is the i-th value that satisfy the Gaussian distribution of minority class calculated by GMM, D_i is the proportion of average Euclidean distance, W_i is the weight of sampling instance of the i-th minority class instance.

$$D_i^{min} = \frac{d_i^{min}}{|k_i^{min}|} \tag{32}$$

$$d_i^{min} = \sum_{j \in k_i^{min}} d(x_i, x_j) \tag{33}$$

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \tag{34}$$

where D_i^{min} is the average Euclidean distance of minority class neighbors on the k-nearest (k=5) neighbors, $|k_i^{min}|$ is the number of minority class neighbors, $d(x_i, x_j)$ is the euclidean distance between x_i and x_j and m is the number of features of the given data T.

The pseudocode of MCS-GMM algorithm is given in algorithm 2.

Algorithm 2: MCS-GMM

Input: training set T, μ, σ, α.

Output: new training set T_{new}.

Input parameters: Calculate μ_{min}, μ_{maj}, σ_{min}, σ_{maj} by GMM, T_g = [] ;
Calculate B_{maj}, B^r_{min}, B^l_{min} and normalized sampling weights using Eq (31);
if $B_{maj} \le B^l_{min}$ **then**
    Synthesize new data using by SMOTE;
    $T_g \leftarrow x_n$;
end
if $B_{maj} \le \mu_{min}$ & $B_{maj} > B^r_{min}$ **then**
    for x_i *in* T_{min} **do**
        if $x_i \le B_{maj}$ & $|B1| \le N_{min}/5$ **then**
            Synthesize new data using Eq (23);
        end
        if $x_i \le B_{maj}$ & $|B1| > N_{min}/5$ **then**
            Synthesize new data using Eq (27);
        end
        if $x_i \le \mu_{min}$ & $x_i > B_{maj}$ **then**
            Get a random value from the sampling interval $[F(x_i), 0.5]$;
        end
        if $x_i > \mu_{min}$ **then**
            Get a random value from the sampling interval $[0.5, F(x_i)]$;
        end
        $T_g \leftarrow x_n$;
    end
end
if $B_{maj} > \mu_{min}$ **then**
    for x_i *in* T_{min} **do**
        if $x_i \le B_{maj}$ **then**
            Synthesize new data using Eq (27);
        end
        if $x_i > B_{maj}$ **then**
            Get a random value from the sampling interval $\left[F(B_{maj}), F(x_i)\right]$;
        end
        $T_g \leftarrow x_n$;
    end
end
if $B_{maj} > B^r_{min}$ **then**
    Synthesize new data using Eq (27);
    $T_g \leftarrow x_n$;
end
$T_{new} \leftarrow T_{maj} \cup T_{min} \cup T_g$

3. Experimental study

3.1. Basic settings

Datasets: Table 1 in the paper presents the comprehensive characteristics of the nine Knowledge Extraction based on Evolutionary Learning (KEEL) datasets. It includes the total number of instances (Size), the number of attributes (Attribute), and the Imbalanced Ratio (IR) for each dataset. IR is the most commonly used measurement for the imbalance degree of imbalanced data, which is defined as follows:

$$IR = \frac{N_{maj}}{N_{min}} \tag{35}$$

where N_{maj} and N_{min} are the number of majority and minority class instances. As can be seen from Table 1, we chose different kinds of datasets. It is important to note that preprocessing was done on every datasets in this investigation. Nominal or categorical attributes were encoded as numerical attributes after the attributes were standardized to a range of [0,1].

Table 1. Description of KEEL Datasets.

Datasets	Size	Attribute	IR
glass1	214	9	1.82
glass6	214	9	6.38
yeast1	1484	8	2.46
yeast4	1484	8	28.1
yeast6	1484	8	39.15
yeast-1_vs_7	459	7	14.3
yeast-2_vs_4	514	8	9.08
vehicle0	846	18	3.23
wisconsin	683	9	1.86

Evaluation metric: accuracy. An evaluation metric is based on balanced datasets. In imbalanced data classification, we have selected G-mean, F-measure and AUC as the evaluation metrics. These metrics are frequently employed to evaluate how well algorithms work with imbalanced data. To calculate them, the confusion matrix is used to determine evaluation metrics for classification models, as illustrated in Table 2.

Table 2. Confusion matrix.

Actual value	Predicted value	
	Minority class	Majority class
Minority class	True Positive (TP)	False Negative (FN)
Majority class	False Positive (FP)	True Negative (TN)

In order to evaluate the performance of a classification model, we use a confusion matrix. This matrix helps us calculate the number of TP, FN, FP and TN. Using these values, we can calculate various metrics to assess the model's performance.

G-mean consists of two parts: Recall and Specificity, which measures the classification accuracy of the classifier for the minority class and the majority class. They are defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{36}$$

$$Specificity = \frac{TN}{TN + FP} \tag{37}$$

$$G - mean = \sqrt{Recall \times Specificity} \tag{38}$$

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{39}$$

The area under curve (AUC) is also an important evaluation metric of predictive ability. It means that the predicted probability of positive samples exceeds the number of predicted probabilities for negative samples. Therefore, they are often used to measure the performance of imbalanced learning models.

3.2. Experiment on KEEL Datasets

In this section, we demonstrate the effectiveness of the proposed MCS-GMM on KEEL data. To conveniently show the performance, we only display the data corresponding to two dimensions, which includes the minority class, majority class and generated data.

The first experiment compares BL1SMOTE and ADASYN using KEEL data to demonstrate the importance of MCS-GMM. For full appreciation, two aspects of first experiment need specific attention: firstly, the instances of the informative minority class are suitably chosen; secondly, the overlapping area around the minority class instances should be avoided as much as possible. In this experiment, the datasets contains 13 minority class instances and 201 majority class instances, and the parameters for the minority class data are $\mu_{min} = (1.52, 13.6)$, $\sigma_{min} = (0.004, 1.09)$.

As shown in Figure 4, comparing with BL1SMOTE and ADASYN, the proposed MCS-GMM selects informative minority class instances and generates more samples in the region where there is little or no overlap between the majority class instances. While both BL1SMOTE and ADASYN fail to find out overlapping area and generates noisy data in subsequent instances generation, which will reduce the discriminative ability of the data and cause overfitting of data. Based on MCS-GMM, BL1SMOTE and ADASYN, we used a decision tree classifier and kept the default parameters of the classifier. The G-mean and AUC is 0.930, 0.935, which shows the superiority of proposed MCS-GMM.

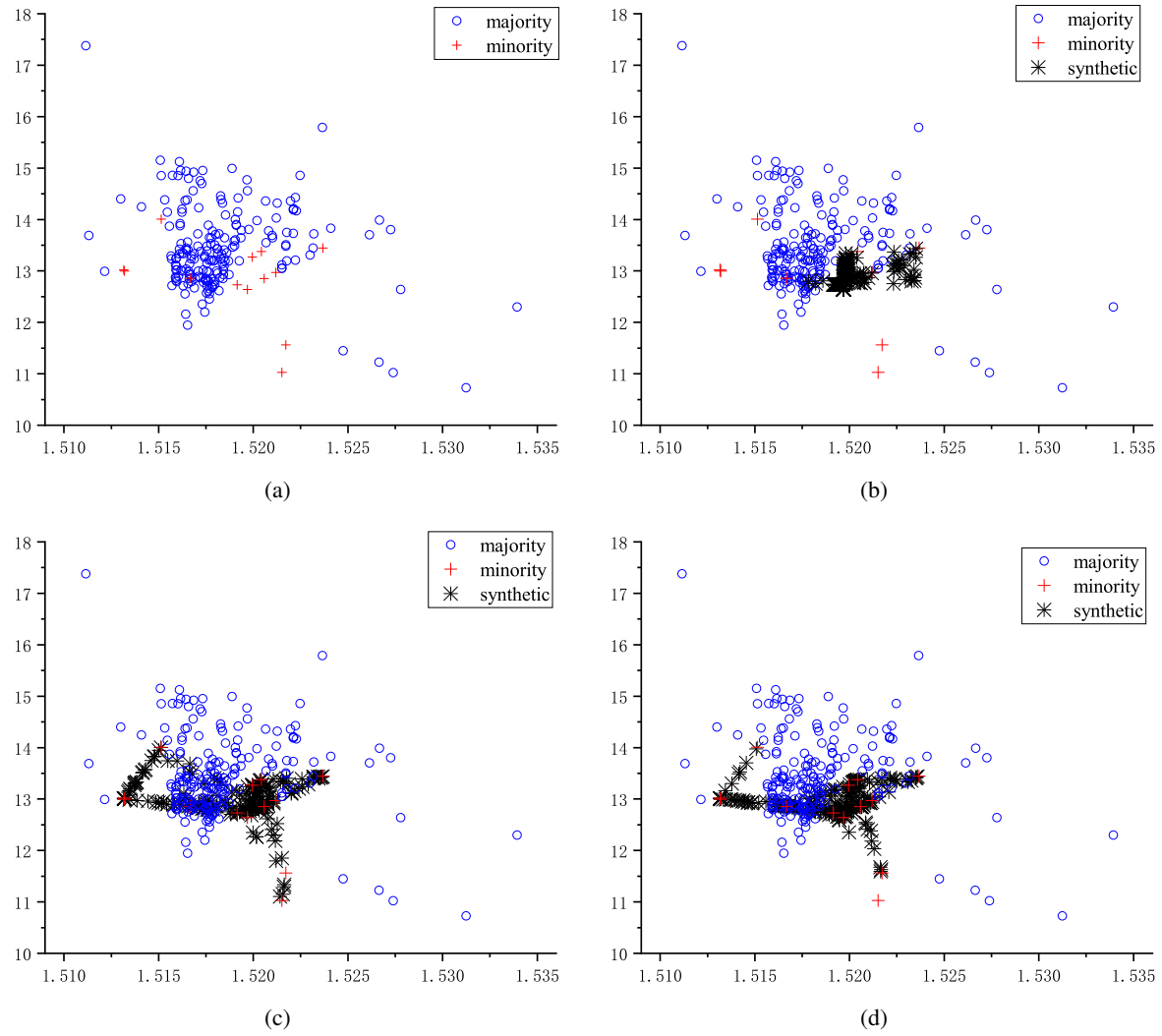

Figure 4. Data visualization of different sampling methods.

3.3. Comparison results by Decision Tree classifier

In addition, we compared our proposal with nine sampling algorithms, namely, SMOTE, Borderline1-SMOTE (BL1SMOTE), Borderline2-SMOTE (BL2SMOTE), Safe-level-SMOTE (SafeSMOTE), ADASYN (ADA), LoRAS [37] and Bagging. Table 3 displays the algorithms that were compared and their specific parameter settings.

In this section, the performance of MCS-GMM is compared with seven other data-level approaches. At the same time, we evaluate classification models using 10-fold cross-validation techniques. Tables 4–6 display the AUC, F-measure and G-mean results. The best results for the datasets are highlighted, and the histogram is used to demonstrate the superiority of the proposed method. To evaluate the algorithm's performance, we count the number of datasets on which it obtains the highest AUC, F-measure and G-mean in comparison to other algorithms. The algorithm's 'winning times' are referred to as this total.

Table 3. Parameter settings of the related algorithms.

Programming language	python 3.8.8
Types of decision trees	criterion="gini", splitter="best"
MCS-GMM	number of Gaussian components=2
SMOTE	neighbors=5, sampling_strategy="auto"
Borderline1-SMOTE	n_neighbors=5, m_neighbors=10
Borderline2-SMOTE	n_neighbors=5, m_neighbors=10
Safe-level-SMOTE	n_neighbors=5, n_jobs=1
ADASYN	n_neighbors=5, sampling_strategy="auto"
LoRAS	sigma=0.005, sampling_strategy='auto'
Bagging	base_estimator=DecisionTreeClassifier()
	n_estimators=10, sampling_strategy='auto'

Table 4. Decision Tree results for G-mean on KEEL datasets.

	MCS-GMM	SMOTE	BL1SMOTE	BL2SMOTE	SafeSMOTE	ADA	LoRAS	Bagging
glass1	0.664 ± 0.132	0.711 ± 0.058	0.742 ± 0.058	**0.758 ± 0.029**	0.550 ± 0.094	0.743 ± 0.072	0.642 ± 0.100	0.743 ± 0.109
glass6	**0.938 ± 0.070**	0.903 ± 0.045	0.937 ± 0.072	0.876 ± 0.105	0.740 ± 0.146	0.806 ± 0.098	0.813 ± 0.082	0.792 ± 0.290
yeast1	**0.699 ± 0.196**	0.686 ± 0.089	0.678 ± 0.012	0.648 ± 0.110	0.534 ± 0.067	0.655 ± 0.102	0.643 ± 0.159	0.554 ± 0.057
yeast4	**0.968 ± 0.056**	0.930 ± 0.024	0.947 ± 0.026	0.931 ± 0.022	0.905 ± 0.122	0.868 ± 0.043	0.912 ± 0.040	0.786 ± 0.086
yeast6	0.972 ± 0.037	0.940 ± 0.024	**0.975 ± 0.017**	0.948 ± 0.025	0.849 ± 0.256	0.893 ± 0.069	0.930 ± 0.027	0.860 ± 0.122
yeast-1_vs_7	**0.928 ± 0.128**	0.870 ± 0.069	0.906 ± 0.050	0.825 ± 0.103	0.795 ± 0.089	0.822 ± 0.078	0.807 ± 0.046	0.515 ± 0.272
yeast-2_vs_4	**0.937 ± 0.110**	0.899 ± 0.055	0.915 ± 0.051	0.846 ± 0.087	0.865 ± 0.110	0.857 ± 0.077	0.878 ± 0.038	0.845 ± 0.154
vehicle0	0.795 ± 0.170	0.789 ± 0.032	**0.810 ± 0.030**	0.728 ± 0.061	0.656 ± 0.046	0.770 ± 0.043	0.779 ± 0.161	0.680 ± 0.029
wisconsin	0.947 ± 0.047	0.946 ± 0.022	0.914 ± 0.031	0.870 ± 0.059	0.936 ± 0.033	0.884 ± 0.079	**0.948 ± 0.047**	0.938 ± 0.039

Table 5. Decision Tree results for F-measure on KEEL datasets.

	MCS-GMM	SMOTE	BL1SMOTE	BL2SMOTE	SafeSMOTE	ADA	LoRAS	Bagging
glass1	0.662 ± 0.153	0.722 ± 0.058	0.754 ± 0.052	**0.766 ± 0.034**	0.562 ± 0.127	0.761 ± 0.065	0.619 ± 0.099	0.679 ± 0.130
glass6	**0.937 ± 0.079**	0.906 ± 0.044	0.936 ± 0.078	0.875 ± 0.109	0.636 ± 0.178	0.800 ± 0.118	0.821 ± 0.074	0.633 ± 0.254
yeast1	**0.674 ± 0.255**	0.671 ± 0.120	0.661 ± 0.138	0.624 ± 0.149	0.491 ± 0.076	0.622 ± 0.133	0.610 ± 0.211	0.407 ± 0.065
yeast4	**0.966 ± 0.062**	0.929 ± 0.025	0.946 ± 0.028	0.930 ± 0.023	0.832 ± 0.142	0.860 ± 0.049	0.910 ± 0.043	0.262 ± 0.046
yeast6	**0.971 ± 0.039**	0.944 ± 0.022	0.975 ± 0.017	0.948 ± 0.026	0.760 ± 0.285	0.887 ± 0.078	0.929 ± 0.029	0.324 ± 0.094
yeast-1_vs_7	**0.921 ± 0.153**	0.866 ± 0.075	0.907 ± 0.053	0.815 ± 0.114	0.707 ± 0.106	0.813 ± 0.087	0.800 ± 0.055	0.188 ± 0.109
yeast-2_vs_4	**0.931 ± 0.131**	0.900 ± 0.055	0.913 ± 0.057	0.836 ± 0.101	0.775 ± 0.138	0.850 ± 0.089	0.872 ± 0.043	0.625 ± 0.166
vehicle0	0.733 ± 0.213	0.805 ± 0.037	**0.817 ± 0.035**	0.726 ± 0.082	0.592 ± 0.057	0.784 ± 0.045	0.754 ± 0.202	0.503 ± 0.030
wisconsin	0.946 ± 0.050	**0.948 ± 0.022**	0.917 ± 0.027	0.870 ± 0.067	0.928 ± 0.036	0.881 ± 0.089	0.946 ± 0.050	0.912 ± 0.044

According to Tables 4–6, our proposed method exhibits slightly higher AUC, F-measure and G-mean values compared to the other methods. Table 4 provides the G-mean results of the experimental study. Our method outperforms in five out of nine datasets in terms of G-mean, is not significantly different in two datasets, and performs significantly worse in two datasets, when considering only 2D information. In addition, our method demonstrates superior performance in certain high IR datasets, specifically 'glass6' and 'yeast4'. Furthermore, MCS-GMM outperforms its strongest competitor,

BL1SMOTE, in two out of the nine datasets. This is because BL1SMOTE tends to select boundary data for generating minority examples. It is important to highlight that BL2SMOTE, LoRAS, also yield favorable outcomes.

Table 5 shows the F-measure results of different algorithms using Decision Tree classifier on the KEEL datasets. From the results in Table 5, the F-measure results of the eight algorithms on the nine datasets are all different. Among them, MCS-GMM performs best on six datasets, and the performance of SMOTE, BL1SMOTE and BL2SMOTE is not as good as MCS-GMM. It performs best on only one dataset of nine datasets. The rest of the algorithms perform even worse on KEEL datasets.

The AUC results are presented in Table 6. For AUC, MCS-GMM performs best for six of the nine datasets. In this case, BL1SMOTE and BL2SMOTE still obtained an excellent result. The winning times of BL1SMOTE and BL2SMOTE has reached one times. Meanwhile, we note that the AUC results for BL1SMOTE does not much different from the values of MCS-GMM. Besides, the performance of SMOTE is not as good as BL1SMOTE, this may be caused by the sampling area. According to the description above, MCS-GMM is better than the six oversampling techniques and generate minority examples in low overlap area and high density function area.

Table 6. Decision Tree results for AUC on KEEL datasets.

	MCS-GMM	SMOTE	BL1SMOTE	BL2SMOTE	SafeSMOTE	ADA	LoRAS	Bagging
glass1	0.677 ± 0.127	0.715 ± 0.073	0.748 ± 0.059	**0.766 ± 0.030**	0.592 ± 0.090	0.752 ± 0.068	0.655 ± 0.100	0.792 ± 0.093
glass6	**0.942 ± 0.061**	0.905 ± 0.043	0.937 ± 0.071	0.878 ± 0.104	0.755 ± 0.015	0.823 ± 0.082	0.818 ± 0.078	0.927 ± 0.108
yeast1	**0.751 ± 0.164**	0.714 ± 0.091	0.714 ± 0.103	0.693 ± 0.099	0.571 ± 0.073	0.699 ± 0.101	0.688 ± 0.127	0.587 ± 0.053
yeast4	**0.970 ± 0.050**	0.931 ± 0.023	0.949 ± 0.025	0.932 ± 0.021	0.915 ± 0.098	0.875 ± 0.036	0.913 ± 0.040	0.861 ± 0.073
yeast6	0.974 ± 0.034	0.945 ± 0.021	**0.975 ± 0.016**	0.949 ± 0.025	0.890 ± 0.163	0.900 ± 0.062	0.933 ± 0.026	0.885 ± 0.129
yeast-1_vs_7	**0.938 ± 0.098**	0.873 ± 0.066	0.908 ± 0.048	0.833 ± 0.099	0.839 ± 0.083	0.826 ± 0.079	0.809 ± 0.044	0.627 ± 0.182
yeast-2_vs_4	**0.946 ± 0.088**	0.899 ± 0.056	0.918 ± 0.048	0.853 ± 0.082	0.861 ± 0.092	0.864 ± 0.070	0.880 ± 0.037	0.900 ± 0.125
vehicle0	**0.873 ± 0.103**	0.852 ± 0.031	0.855 ± 0.028	0.800 ± 0.048	0.753 ± 0.048	0.824 ± 0.028	0.867 ± 0.102	0.761 ± 0.045
wisconsin	0.974 ± 0.031	**0.980 ± 0.014**	0.964 ± 0.019	0.943 ± 0.034	0.975 ± 0.020	0.958 ± 0.032	0.973 ± 0.030	0.977 ± 0.019

In addition, MCS-GMM considered two effects: spatial and statistical features. Based on surrounding neighbors of minority example and the distribution of minority class, a new way of weighting is determined. Thus, our method can generate higher quality data and shows high performance on the nine imbalanced datasets, especially for the high imbalanced datasets.

As shown in Figure 5, it describes the winning times of seven different oversampling methods and orange represents the number of winning times for AUC, yellow represents the number of winning times for F-measure, green represents the number of winning times for G-mean. It can be seen from Figure 5, MCS-GMM has more winning times than other oversampling techniques in AUC, F-measure and G-mean. The winning times of MCS-GMM has reached 17 times. The second place is BL1SMOTE, which also has good performance and its winning times has reached 4 times. The rest of the oversampling algorithm achieved a lower number of winning times compared to the new algorithm. These results indicate that our new method is more effective in imbalanced data classification.

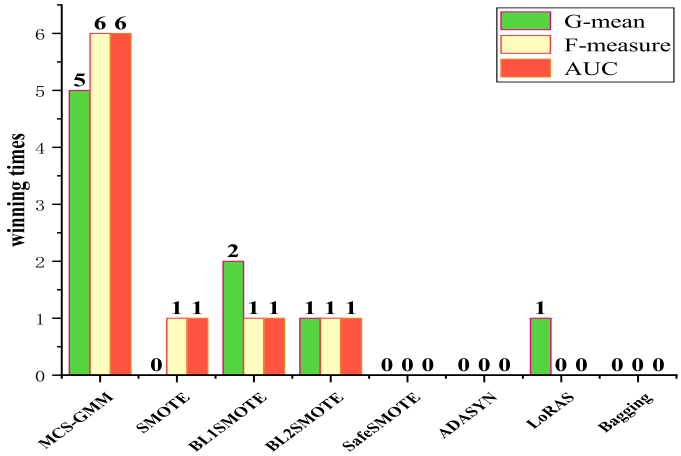

Figure 5. Comparison of winning times.

4. Time complexity of the proposed algorithm

Time complexity of the proposed algorithm: In MCS-GMM, the time complexity of the EM algorithm is $O(n^2 + n)$, the time complexity of normalization and calculation of weights is $O(m)$. Besides, the time complexity of k-nearest neighbor algorithm is $O(n * M * log(k))$. Finally, the time complexity of the proposed algorithm is $O(n^2 + n + m + n * M * log(k))$, where n is the number of dataset, m is the number of minority class, k is the number of nearest neighbors used in the k-nearest neighbors algorithm, M is the number of features.

5. Conclusions

This paper proposes a new sampling method based on Gaussian distribution which can extract better features by combining statistical and spatial features, and the distribution of imbalanced data is guaranteed. In addition, MCS-GMM generates less samples in overlapping regions by designing new weights and tries to make use of as many instances that contain further information as possible. tries to make use of as many instances that contain further information as possible. A sample is selected to generate new instances and prevent any overfitting of the following model learning. Besides, our proposed method is beneficial for small datasets with high degree of overlap and can be applied to protein classification.

The results of the experiment show that the proposed method is effective in completing classification tasks, while also effectively preventing overfitting. Furthermore, the method is versatile and can be applied to different tasks. Particularly, in terms of overlapping area, our method can generate higher quality data to improve classification performance. Compared to other oversampling techniques, MCS-GMM has a higher value in AUC, but BL1SMOTE outperforms MCS-GMM on some datasets with lower overlapping ratios in terms of AUC. However, it should be pointed that, BL1SMOTE has also shown good performance in our experiments as it generates new minority in border regions.

Although the method proposed in this paper performed well, it still has some limitations. First, the probability density function of the imbalanced data is not accurate enough, and in the future, a multilayer neural network approach can be chosen get a more accurate probability density function.

Second, the information of neighbors for each minority sample not get enough. In future research, we will pay more attention to these limitations.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. C. Phua, D. Alahakoon, V. Lee, Minority report in fraud detection: classification of skewed data, *ACM SIGKDD Explor. Newsl.*, **6** (2004), 50–59. https://doi.org/10.1145/1007730.1007738
2. B. Krawczyk, M. Galar, L. Jelen, F. Herrera, Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy, *Appl. Soft Comput.*, **38** (2016), 714–726. https://doi.org/10.1016/j.asoc.2015.08.060
3. J. Alqatawna, H. Faris, K. Jaradat, M. Al-Zewairi, O. Adwan, Improving knowledge based spam detection methods: The effect of malicious related features in imbalance data distribution, *Int. J. Commun. Network Syst. Sci.*, **8** (2015), 118–129. https://doi.org/10.4236/ijcns.2015.85014
4. N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, *Intell. Data Anal.*, **6** (2002), 429–449. https://doi.org/10.3233/IDA-2002-6504
5. X. Fan, H. Yu, GAMC: An oversampling method based on genetic algorithm and monte carlo method to solve the class imbalance issue in industry, in *2022 International Conference on Industrial IoT, Big Data and Supply Chain (IIoTBDSC)*, (2022), 127–132. https://doi.org/10.1109/IIoTBDSC57192.2022.00033
6. F. Zhang, G. Liu, Z. Li, C. Yan, C. Jang, GMM-based undersampling and its application for credit card fraud detection, in *2019 International Joint Conference on Neural Networks (IJCNN)*, (2019), 1–8. https://doi.org/10.1109/IJCNN.2019.8852415
7. Y. Yan, Y. Zhu, R. Liu, Y. Zhang, Y. Zhang, L. Zhang, Spatial distribution-based imbalanced undersampling, *IEEE Trans. Knowl. Data Eng.*, **35** (2023), 6376–6391. https://doi.org/10.1109/TKDE.2022.3161537
8. H. Zhu, M. Zhou, G. Liu, Y. Xie, S. Liu, C. Guo, NUS: Noisy-sample-removed undersampling scheme for imbalanced classification and application to credit card fraud detection, *IEEE Trans. Comput. Soc. Syst.*, (2023), 1–12. https://doi.org/10.1109/TCSS.2023.3243925
9. N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority oversampling technique, *J. Artif. Intell. Res.*, **16** (2002), 321–357. https://doi.org/10.1613/jair.953
10. A. Fernández, S. Garcia, F. Herrera, N. V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, *J. Artif. Intell. Res.*, **61** (2018), 863–905. https://doi.org/10.1613/jair.1.11192

11. H. He, Y. Bai, E. A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, (2008), 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969
12. H. Han, W. Wang, B. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in *International Conference on Intelligent Computing*, **3644** (2005), 878–887. https://doi.org/10.1007/11538059_91
13. G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, *Inf. Sci.*, **465** (2018), 1–20. https://doi.org/10.1016/j.ins.2018.06.056
14. Y. Yan, Y. Jiang, Z. Zheng, C. Yu, Y. Zhang, Y. Zhang, LDAS: Local density-based adaptive sampling for imbalanced data classification, *Expert Syst. Appl.*, **191** (2022), 116213. https://doi.org/10.1016/j.eswa.2021.116213
15. Y. Xie, M. Qiu, H. Zhang, L. Peng, Z. Chen, Gaussian distribution based oversampling for imbalanced data classification, *IEEE Trans. Knowl. Data Eng.*, **34** (2022), 667–669. https://doi.org/10.1109/TKDE.2020.2985965
16. H. Bhagwani, S. Agarwal, A. Kodipalli, R. J. Martis, Targeting class imbalance problem using GAN, in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, (2021), 318–322. https://doi.org/10.1109/ICEECCOT52851.2021.9708011
17. S. Maldonado, C. Vairetti, A. Fernandez, F. Herrera, FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification, *Pattern Recognit.*, **124** (2022), 108511. https://doi.org/10.1016/j.patcog.2021.108511
18. E. Kaya, S. Korkmaz, M. A. Sahman, A. C. Cinar, DEBOHID: A differential evolution based oversampling approach for highly imbalanced datasets, *Expert Syst. Appl.*, **169** (2021), 794–801. https://doi.org/10.1016/j.eswa.2020.114482
19. W. Xie, G. Liang, Z. Dong, B. Tan, B. Zhang, An improved oversampling algorithm based on the samples' selection strategy for classifying imbalanced data, *Math. Probl. Eng.*, **2019** (2019), 3526539. https://doi.org/10.1155/2019/3526539
20. L. Peng, H. Zhang, B. Yang, Y. Chen, A new approach for imbalanced data classification based on data gravitation, *Inf. Sci.*, **288** (2014), 347–373. https://doi.org/10.1016/j.ins.2014.04.046
21. F. Rahmati, H. Nezamabadi-Pour, B. Nikpour, A gravitational density-based mass sharing method for imbalanced data classification, *SN Appl. Sci.*, **2** (2020). https://doi.org/10.1007/s42452-020-2039-2
22. M. Koziarski, B. Krawczyk, M. Wozniak, Radial-Based oversampling for noisy imbalanced data classification, *Neurocomputing*, **343** (2019), 19–33. https://doi.org/10.1016/j.neucom.2018.04.089
23. C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for handling the class imbalanced problem, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, **5476** (2009), 475–482. https://doi.org/10.1007/978-3-642-01307-2_43

24. Y. Sun, L. Cai, B. Liao, W. Zhu, J. Xu, A robust oversampling approach for class imbalance problem with small disjuncts, *IEEE Trans. Knowl. Data Eng.*, **35** (2023), 5550–5562. https://doi.org/10.1109/TKDE.2022.3161291
25. S. Yin, X. Zhu, C. Jing, Fault detection based on a robust one class support vector machine, *Neurocomputing*, **145** (2014), 263–268. https://doi.org/10.1016/j.neucom.2014.05.035
26. B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, *Neural Comput.*, **13** (2001), 1443–1471. https://doi.org/10.1162/089976601750264965
27. R. Barandela, R. M. Valdovinos, J. S. Sánchez, New applications of ensembles of classifiers, *Pattern Anal. Appl.*, **6** (2003), 245–256. https://doi.org/10.1007/s10044-003-0192-z
28. C. Li, Classifying imbalanced data using a bagging ensemble variation (BEV), in *Proceedings of the 45th annual southeast regional conference*, (2007), 203–208. https://doi.org/10.1145/1233341.1233378
29. S. Hido, H. Kashima, Y. Takahashi, Roughly balanced bagging for imbalanced data, *Stat. Anal. Data Min.*, **2** (2009), 412–426. https://doi.org/10.1002/sam.10061
30. B. Chen, S. Xia, Z. Chen, B. Wang, G. Wang, RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise, *Inf. Sci.*, **553** (2021), 397–428. https://doi.org/10.1016/j.ins.2020.10.013
31. H. K. Lee, S. B. Kim, An overlap-sensitive margin classifier for imbalanced and overlapping data, *Expert Syst. Appl.*, **98** (2018), 72–83. https://doi.org/10.1016/j.eswa.2018.01.008
32. J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, *IEEE Trans. Knowl. Data Eng.*, **31** (2019), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857
33. M. K. Paul, B. Pal, Gaussian mixture based semi supervised boosting for imbalanced data classification, in *2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, (2016).
34. Y. Xie, L. Peng, Z. Chen, B. Yang, H. Zhang, H. Zhang, Generative learning for imbalanced data using the Gaussian mixed model, *Appl. Soft Comput.*, **79** (2019), 439–451. https://doi.org/10.1016/j.asoc.2019.03.056
35. A. Shapiro, Monte carlo sampling methods, *Handb. Oper. Res. Manage. Sci.*, **10** (2003), 353–425. https://doi.org/10.1016/S0927-0507(03)10006-0
36. D. P. Kroese, T. Brereton, T. Taimre, Z. I. Botev, Why the Monte Carlo method is so important today, *WIREs Comput. Stat.*, **6** (2014), 386–392. https://doi.org/10.1002/wics.1314
37. S. Bej, N. Davtyan, M. Wolfien, M. Nassar, O. Wolkenhauer, LoRAS: An oversampling approach for imbalanced datasets, *Mach. Learn.*, **110** (2021), 279–301. https://doi.org/10.1007/s10994-020-05913-4

AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)