Information Sciences 585 (2022) 313–343

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Binary imbalanced data classification based on diversity oversampling by generative models

Junhai Zhai *, Jiaxing Qi, Chu Shen

Hebei Key Laboratory of Machine Learning and Computational Intelligence, College of Mathematics and Information Science, Hebei University, Baoding 071002, Hebei, China

ARTICLE INFO

Article history:
Received 19 February 2021
Received in revised form 23 September 2021
Accepted 19 November 2021
Available online 25 November 2021

Keywords:
Imbalanced learning
Binary imbalanced data classification
Diversity oversampling
Generative adversarial network
Extreme learning machine autoencoder

ABSTRACT

In many practical applications, the data are class imbalanced. Accordingly, it is very meaningful and valuable to investigate the classification of imbalanced data. In the framework of binary imbalanced data classification, the synthetic minority oversampling technique (SMOTE) is the best-known oversampling method. However, for each positive sample, SMOTE generates only k synthetic samples on the lines between the positive sample and its k-nearest neighbors, resulting in three drawbacks: (1) SMOTE cannot effectively extend the training field of positive samples; (2) the generated positive samples lack diversity; (3) SMOTE does not accurately approximate the probability distribution of the positive samples. Therefore, two binary imbalanced data classification methods named BIDC1 and BIDC2 based on diversity oversampling by generative models are proposed. The BIDC1 and BIDC2 conduct diversity oversampling using extreme learning machine autoencoder and generative adversarial network, respectively. Extensive experiments on 26 data sets are conducted to compare the two methods with 14 state-of-the-art methods using five metrics: F-measure, G-means, AUC-area, MMD-score, and Silhouette-score. The experimental results demonstrate that the two methods outperform the other 14 methods.
© 2021 Elsevier Inc. All rights reserved.

1. Introduction

A binary classification problem is imbalanced when the majority class (negative class) is significantly larger than the minority class (positive class). Many real-world applications are binary imbalanced classification problems, such as software defect predictions, medical diagnosis, credit card fraud detection, spam filtering, and severe convective weather prediction. Therefore, binary imbalanced classification problems have attracted the attention of researchers during the last decade. In binary imbalanced data classification, misclassifying a positive sample is more costly than misclassifying a negative sample. For example, diagnosing a sick patient as a healthy person has more serious consequences than diagnosing a healthy person as a sick person. Since imbalance introduces a bias favoring the majority class, it is challenging for traditional classification algorithms to discriminate between the minority and majority classes. Balancing the distributions of the minority and majority classes by augmenting positive class samples is a predominant strategy for handling imbalanced data classifications. The most notable data augmentation method is the synthetic minority oversampling technique (SMOTE) [1], which generates synthetic samples on the lines between a positive sample and its k-nearest neighbors. Since SMOTE was proposed

* Corresponding author.
E-mail address: mczjh@126.com (J. Zhai).

https://doi.org/10.1016/j.ins.2021.11.058
0020-0255/© 2021 Elsevier Inc. All rights reserved.

in 2002, many variants of SMOTE have been developed, such as borderline SMOTE (B-SMOTE) [2], adaptive synthetic sampling algorithm (ADASYN) [3], geometric SMOTE (GSMOTE) [4], multiple imputation-based minority oversampling (MI-MOTE) [5], and robust SMOTE (RSMOTE) [6]. However, SMOTE family algorithms have three drawbacks: (1) they do not effectively extend the training field of positive samples; (2) the generated positive samples lack diversity; (3) SMOTE does not accurately approximate the probability distribution of the positive samples.

Therefore, motivated by generative models, we propose two binary imbalanced data classification methods BIDC1 and BIDC2 based on diversity oversampling. The two methods carry out diversity oversampling using extreme learning machine autoencoder (ELMAE) and generative adversarial network (GAN), respectively. The main contributions of this paper are threefold.

1. We propose a novel framework to generate new positive samples using generative models iteratively. Unlike SMOTE and its variants, the proposed framework generates new positive samples in each iteration based on the distribution of the positive samples generated in the previous iteration rather than generating new positive samples on the lines between a positive sample to its k-nearest neighbors.
2. We use two generative models, i.e., ELMAE and GAN, to generate new positive samples successively. We use the maximum mean discrepancy score (MMD-score) [7] and the Silhouette-score [8] to ensure the high quality of the generated positive samples. The MMD-score measures the diversity of the generated positive samples, and the Silhouette-score evaluates the separability between the generated positive samples and the original negative samples.
3. We conduct extensive experiments on 26 data sets to compare the BIDC1 and BIDC2 with 14 state-of-the-art methods. The experimental results on the F-measure, G-means, AUC-area, MMD-score, and Silhouette-score show that our methods outperform the 14 approaches.

The rest of this paper is organized as follows. In Section 2, we review the related works of binary imbalanced data classification. In Section 3, we describe the details of the proposed methods. In Section 4, Extensive experiments compared the BIDC1 and BIDC2 with 14 state-of-the-art methods are carried out to verify the effectiveness of the two methods. At last, we conclude our work in the Section 5.

2. Related works

Many methods have been proposed for addressing binary imbalanced data classification problem. The state-of-the-art methods were reviewed in two excellent papers [9,10]. Generally, binary imbalanced data classification methods can be roughly classified into three categories: data-level methods, algorithm-level methods, and hybrid methods.

Let $S = S^+ \cup S^-$, where S^+ and S^- denote the positive and negative class respectively. The goal of data-level methods is to balance the distribution of the samples in the two classes. The basic strategy is sampling, including random undersampling (RUS) and random oversampling (ROS) [9]. Undersampling refers to randomly selecting a subset S_u^- with the same size as S^+ from S^-. The union $S_u^- \cup S^+$ is a balanced subset of S. Subsequently, a classifier is trained on $S_u^- \cup S^+$ using a standard classification algorithm, such as a neural network, decision tree, or support vector machine. The drawback of undersampling is that only a very small subset S_u^- of S^- with S^+ is used for classification. An improvement is to repeat undersampling p times, where p is a hyperparameter; p classifiers are trained and integrated using an ensemble method for classification. Oversampling refers to randomly duplicating positive samples or generating synthetic positive samples using specific strategies. SMOTE is the predominant method for augmenting synthetic positive samples. Fernández et al. [11] presented a very comprehensive review of SMOTE-based approaches and its progress and challenges in fifteen years (2003 to 2018). The representative methods include B-SMOTE [2], ADASYN [3], GSMOTE [4], MI-MOTE [5], and RSMOTE [6]. B-SMOTE [2] is based on the assumption that the samples far from the boundary contribute little to the classification results; thus only the minority examples near the boundary are over sampled. ADASYN [3] is an adaptive synthetic sampling approach for imbalanced learning. It uses a weighted distribution for different positive samples according to their level of difficulty in learning. More synthetic data are generated for positive samples with a higher level of difficulty in learning than for positive samples with a lower level. GSMOTE [4] generates synthetic positive samples in a geometric region around a selected positive sample in the input space. MI-MOTE [5] is a method for imbalanced and incomplete data classification. RSMOTE [6] improves the robustness of SMOTE by incorporating label noise. In recent years, some new data-level methods have been proposed by different researchers, such as K-means-SMOTE (K-SMOTE) [12], adaptive neighbor synthetic-oversampling (ANS) [13], combined cleaning and resampling (CCR) [14], noise reduction a priori synthetic over-sampling (NRPSOS) [15], clustering using representatives SMOTE (C-SMOTE) [16], self-organizing map oversampling (SOMO) [17], Gaussian-SMOTE (G-SMOTE) [18], and over-sampling using propensity scores (OUPS) [19]. Since generative models [20] and their variants [21–24] can generate realistic samples, several researchers used generative models to handle imbalanced data classification problem. For example, inspired by the auxiliary classifier-GAN (AC-GAN), Ali-Gombe and Elyan proposed an improved model named multiple fake class-GAN (MFC-GAN) [25] for imbalanced data classification. The MFC-GAN differs from the AC-GAN by using a classifier, a discriminator, and multiple fake classes rather than the single fake class in AC-GAN. Furthermore, the MFC-GAN preserves the structure of the minority classes by learning the correct data distribution, which is a desirable property. Zhang et al. [26] used the conditional Wasserstein GAN with a gradient penalty for imbalanced data classification and proposed a novel and

efficient synthetic oversampling approach. Unlike existing methods, the novelty of the proposed methods is that the MMD and Silhouette score ensure the diversity of the synthetic samples. The diversity significantly improves the performance of imbalanced data classification, as was demonstrated by Wang and Yao [27].

Algorithm-level methods modify existing classification algorithms to adapt them to imbalanced classification scenarios. Khan et al. [28] proposed a cost-sensitive deep neural network model that automatically learned useful features from imbalanced data by simultaneously optimizing the class correlation losses and network parameters. In many practical applications, the losses of misclassification are unknown and are difficult to determine. Zhang et al. [29] modified a traditional deep belief network model and proposed an evolutionary cost-sensitive deep belief network model that effectively handled imbalanced data classification problem. Dong et al. [30] used a deep learning approach for imbalanced data classification and formulated an imbalanced deep learning model based on batch-wise incremental minority class rectification. In the context of deep learning, Mateusz et al. [31] systematically investigated the impact of class imbalance on the classification performance of convolutional neural networks. They experimentally compared frequently used imbalanced data classification algorithms using three benchmark data sets (MNIST, CIFAR-10, and ImageNet).

Hybrid methods combine data-level and algorithm-level methods and typically employ ensemble learning to integrate both approaches. Because the hybrid approaches have the advantage of the two methods, they outperform the two individual methods in generalization performance. Hybrid methods can be broadly classified into sampling-based methods and cost-sensitive based methods. In the first category, oversampling or undersampling is used. SMOTEBoost [32] is the most well-known early oversampling-based method that integrates SMOTE and boosting. The innovation of SMOTEBoost is the weighting mechanism, which differs from the original method. Specifically, the original boosting method gives all misclassified samples equal weights, whereas SMOTEBoost generates synthetic positive samples, changing the updating weights and compensating for skewed distributions. Chen et al. [33] proposed an oversampling-based ensemble method based on the localized generalization error model. The generated synthetic positive samples are located in a local area of the training samples, and the basic classifiers are trained with the combined original negative training samples and the synthetic neighborhood samples. Raghuwanshi and Shukla [34] proposed an undersampling-based ensemble method that creates several balanced training subsets by random undersampling of the majority class samples. The number of training subsets is determined by the degree of the class imbalance. The generated balanced training subsets are used for training the basic classifiers, and bagging is used as the ensemble method. The drawback of this method is that the number of training subsets is very large if the original data set has a high imbalance ratio. Based on the generalized imbalance ratio, which is a core concept in [35], Tang and He proposed two sampling-based ensemble methods, i.e., undersampling-based and oversampling-based ensemble approaches. Both approaches adaptively split the imbalanced training set into multiple balanced training subsets using a probabilistic method. Multiple weak classifiers are trained using a boosting method. Chen et al. [36] proposed a hybrid data-level ensemble method for imbalanced data classification that integrates ensemble learning with combined margin-based undersampling and diversity-enhancing oversampling. In the second category, pioneering cost-sensitive-based methods were proposed by Zhou et al. [37] and by Sun et al. [38]. The method in [37] is tailored for training cost-sensitive neural networks to address the class imbalance, and both oversampling and undersampling are considered. In [38], different weighting mechanisms were used in the conventional AdaBoost algorithm. Tao et al. [39] proposed a cost-sensitive-based ensemble method for imbalanced data classification. The novelty of the proposed method is a self-adaptive cost-sensitive-based support vector machine used for ensemble classification. Alaba et al. [40] reviewed state-of-the-art cost-sensitive ELM methods and presented recent trends in imbalanced data classification.

3. The two methods

This section presents the two methods BIDC1 and BIDC2 for binary imbalanced data classification. The first method is based on diversity oversampling and uses ELMAE, and the second method is based on diversity oversampling and uses GAN.

3.1. Binary imbalanced data classification based on diversity oversampling using extreme learning machine autoencoder

An ELM [41] is a random algorithm for training a single hidden layer feedforward neural network (SLFNN) (Fig. 1). The activation functions of the hidden layer nodes are sigmoid functions, whereas the activation functions of the input and output layer nodes are linear functions. The randomness of ELM is ensured by randomly generating the weights between the input layer and hidden layer and the biases of the hidden layer nodes. The SLFNN trained by the ELM can be viewed as a random weight network [42], and its generalization performance is closely related to uncertainty [43,44]. Since it is unnecessary for an ELM to adjust the network parameters iteratively, it has a very fast learning speed with excellent generalization performance and has found many successful applications [45–48].

Given a training set $S = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in R^d, \mathbf{y}_i \in R^k, i = 1, 2, \cdots, n\}$, ELM only needs to solve the following linear Eq. (1). In other words, it only needs to calculate the Moore–Penrose generalized inverse of hidden output matrix $\mathbf{H}$.

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y} \tag{1}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_m \cdot \mathbf{x}_1 + b_m) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_n + b_1) & \cdots & g(\mathbf{w}_m \cdot \mathbf{x}_n + b_m) \end{bmatrix} \tag{2}$$

$$\boldsymbol{\beta} = (\boldsymbol{\beta}_1^{\mathrm{T}}, \cdots, \boldsymbol{\beta}_m^{\mathrm{T}})^{\mathrm{T}} \tag{3}$$

and

$$\mathbf{Y} = (\mathbf{y}_1^{\mathrm{T}}, \cdots, \mathbf{y}_n^{\mathrm{T}})^{\mathrm{T}} \tag{4}$$

In the (2), $g(\cdot)$ is the activation function of hidden layer nodes, $\mathbf{w}_j = (w_{j1}, w_{j2}, \cdots, w_{jd})^{\mathrm{T}}$ is the weight vector connecting the j^{th} hidden node with the input nodes, b_j is the bias of the j^{th} hidden node. In the (3), $\boldsymbol{\beta}_j = (\beta_{j1}, \beta_{j2}, \cdots, \beta_{jm})^{\mathrm{T}}$ is the weight vector connecting the j^{th} hidden node with the output nodes, T stands for transpose.

Generally, the number of hidden nodes is much less than the number of training samples. Accordingly, $\mathbf{H}$ is not a square matrix, and the Eq. (1) hasn't exact solution, one can find its smallest norm least square solution by solving the following optimization problem.

$$\min_{\boldsymbol{\beta}} ||\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}|| \tag{5}$$

The smallest norm least-squares solution of (5) can be obtained by (6).

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{6}$$

where $\mathbf{H}^{\dagger} = (\mathbf{H}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{H}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$. The pseudocode of ELM is given in Algorithm 1.

Algorithm 1: The ELM Algorithm

Input: Training data set $S = \{(\mathbf{x}_i, \mathbf{y}_i)|\mathbf{x}_i \in R^d, \mathbf{y}_i \in R^k, i = 1, 2, \cdots, n\}$, an activation function $g(\cdot)$, and the number of hidden nodes m

Output: weights matrix β.

```
for (j = 1; j ≤ m; j = j + 1) do
    Randomly assign input weights w_j and biases b_j;
end
Calculate the hidden layer output matrix H;
Calculate output weights matrix β̂ = H†Y.
```

We can introduce a regularization item into (5), the corresponding optimization problem becomes (7).

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \{\tfrac{1}{2}||\beta||_2^2 + \tfrac{C}{2}\sum_{i=1}^{n}||\xi||_2^2\} \\ &s.t.\ \ \boldsymbol{\beta}^T\mathbf{h}_i = \mathbf{y}_i - \boldsymbol{\xi}_i, 1 \leqslant i \leqslant n. \end{aligned} \tag{7}$$

where $\boldsymbol{\xi}_i$ is the error vector corresponding to $\mathbf{x}_i$ and C is a positive parameter.

The solution of optimization problem (7) is given by

$$\hat{\boldsymbol{\beta}} = (\frac{1}{C}\mathbf{I} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}\mathbf{Y}^T \tag{8}$$

where $\mathbf{I}$ is the identity matrix.

In Fig. 1, if we let the number of nodes in the output layer equal to the number of nodes in the input layer, and let the input equal to the output, i.e. $\mathbf{y}_i = \mathbf{x}_i$, then the SLFNN given in Fig. 1 becomes an ELMAE (Fig. 2). Obviously, we can use ELM algorithm to train ELMAE.

The ELMAE can be regarded as a generative model. We propose a binary imbalanced data classification algorithm (BIDC1) that uses ELMAE to generate positive samples to balance the data sets. It should be noted that the generated positive samples should be similar but distinct from the original positive samples. Furthermore, the generated positive samples should not overlap with the negative samples, a feature we refer to as separability. The separability between the generated positive samples and the original negative samples is determined by the Silhouette coefficient [8], which is named Silhouette-

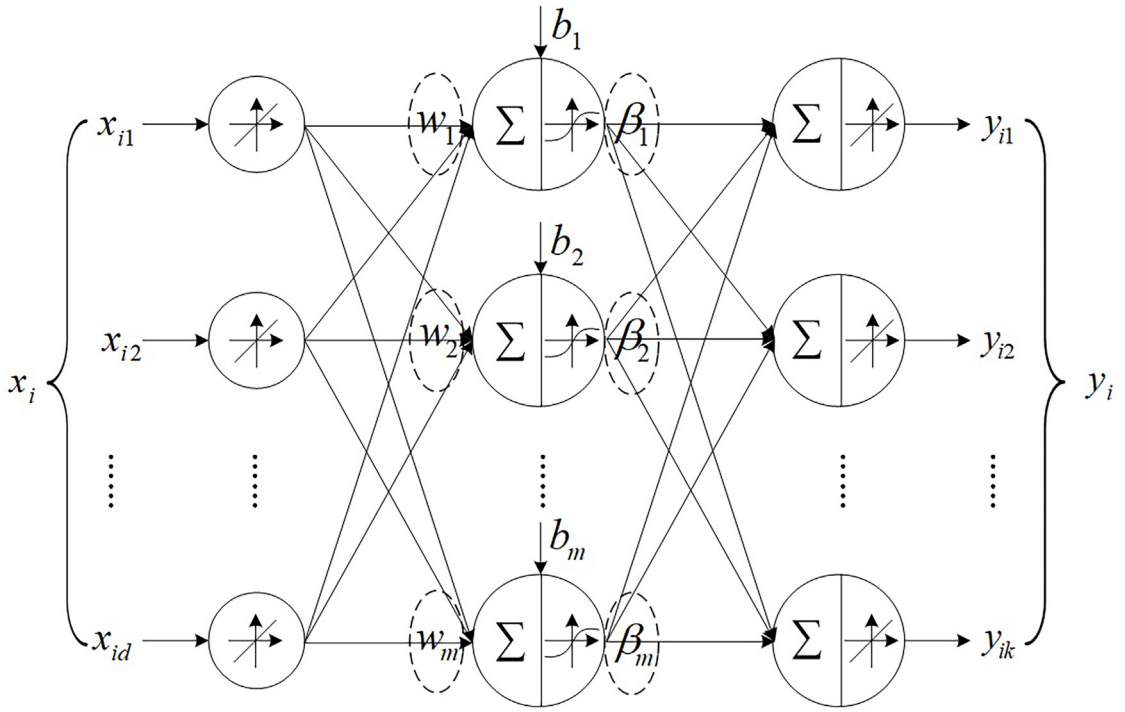

Fig. 1. The single hidden layer feedforward neural network with special architecture.

score in this paper. We use the MMD [7] to characterize the similarity between the generated positive samples and the original positive samples.

The Silhouette-score [8] is an evaluation index of clustering algorithms. Given a sample $\mathbf{x}$ which belongs to cluster A, the Silhouette-score of $\mathbf{x}$ is defined as Eq. (9).

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}} \tag{9}$$

where $a(\mathbf{x})$ is the average dissimilarity of sample $\mathbf{x}$ to all other samples of A, $b(\mathbf{x}) = \min_{C \neq A} d(\mathbf{x}, C)$, while $d(\mathbf{x}, C)$ is the average dissimilarity of sample $\mathbf{x}$ to all samples of cluster C. With respect to a cluster (or a set) A, the Silhouette-score of A is $s(A) = \frac{1}{|A|}\sum_{\mathbf{x}\in A} s(\mathbf{x})$. From Eq. (9), it is easy to find that the value of $s(\mathbf{x})$ is between [-1,1], and the closer the value of $s(\mathbf{x})$ to 1, the better the separability is.

The MMD is a statistics for measuring the mean squared difference of two sets of samples. Given two sets of samples $\mathbf{X} = \{\mathbf{x}_i\}, 1 \leqslant i \leqslant n$ and $\mathbf{Y} = \{\mathbf{y}_i\}, 1 \leqslant i \leqslant m$, the MMD of $\mathbf{X}$ and $\mathbf{Y}$ is defined as Eq. (10).

$$\begin{aligned} \mathrm{MMD} &= \left\| \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(\mathbf{y}_i) \right\|^2 \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n}\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_{i'}) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(\mathbf{x}_i)^T\phi(\mathbf{y}_j) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m}\phi(\mathbf{y}_j)^T\phi(\mathbf{y}_{j'}) \end{aligned} \tag{10}$$

In Eq. (10), $\phi(\cdot)$ is a kernel mapping, using kernel trick, Eq. (10) can be written as Eq. (11).

$$\mathrm{MMD} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n}k(\mathbf{x}_i, \mathbf{x}_{i'}) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}k(\mathbf{x}_i, \mathbf{y}_j) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m}k(\mathbf{y}_j, \mathbf{y}_{j'}) \tag{11}$$

The Gaussian kernel is typically used to calculate the value of MMD (i.e., the MMD-score) between the distribution of the original positive samples and the generated positive samples and evaluate the divergence of the two distributions. The larger the MMD-score, the greater the divergence between the two distributions is, and the higher the diversity of the generated samples is.

The BIDC1 consists of three stages: (1) training the ELMAE on the original data set; (2) generating synthetic positive samples with the trained ELMAE model to balance the original data set; (3) training a classifier model on the balanced data set and classifying the testing samples. The pseudocode of the BIDC1 is given in Algorithm 2.

Algorithm 2: The BIDC1 algorithm

Input: Imbalanced training set $S_{tr} = S_{tr}^{+} + S_{tr}^{-}$, where the S_{tr}^{+} is the set of positive training examples, and S_{tr}^{-} is the set of negative training examples; Imbalanced testing set $S_{te} = S_{te}^{+} + S_{te}^{-}$, where the S_{te}^{+} is the set of positive test examples, and S_{te}^{-} is the set of negative test examples; The activation function $g(\cdot)$, the number of hidden nodes m, and the iterative number t.

Output: The classification results of $\mathbf{x} \in S_{te}$.

1 // Stage 1: training the ELMAE on S_{tr};
2 **for** $(j = 1; j \leq m; j = j + 1)$ **do**
3 | Randomly assign input weights $\mathbf{w}_j$ and b_j;
4 **end**
5 Calculate the hidden layer output matrix $\mathbf{H}$;
6 Calculate output weights matrix $\hat{\beta} = (\frac{1}{C}\mathbf{I} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}\mathbf{X}^T$;
7 // Stage 2: generating synthetic positive samples with the trained ELMAE model;
8 $S_1^{+} = S_{tr}^{+}$;
9 **for** $(i = 1; i \leq t; i = i + 1)$ **do**
10 | Input S_i^{+} into ELMAE, and compressed vectors can be obtained by the encoder;
11 | Take these vectors added Gaussian noise with normal distribution as input of decoder, then get the generate synthetic positive samples;
12 | Select informative positive samples from the synthetic ones by Silhouette-score and MMD-score, the set of selected positive samples is denoted by S_{gen}^{+};
13 | $S_{i+1}^{+} = S_i^{+} + S_{gen}^{+}$;
14 **end**
15 // Stage 3: training a classifier model on balanced data set and classifying testing samples;
16 $S_{tr}^{+} = S_{t+1}^{+}$;
17 $S_{tr} = S_{tr}^{+} + S_{tr}^{-}$;
18 Train a classifier on S_{tr}, and use the trained classifier to classify $\mathbf{x} \in S_{te}$;

Note: To execute the statement 11 in the Algorithm 2, we first use the existing positive samples and the generated positive samples to calculate MMD-score and use the generated positive samples and negative samples to calculate the Silhouette-score. Specifically, S_i^{+} are used as the input of ELMAE to obtain the generated positive samples in i^{th} iteration, and we use the random sampling method to get the generated positive samples subset, then calculate the MMD-score between S_i^{+} and the subset, calculate the Silhouette-score between S_{tr}^{-} and the subset. Finally, we select a subset S_{gen}^{+} with the highest scores, the score is the sum of MMD-score add Silhouette-score.

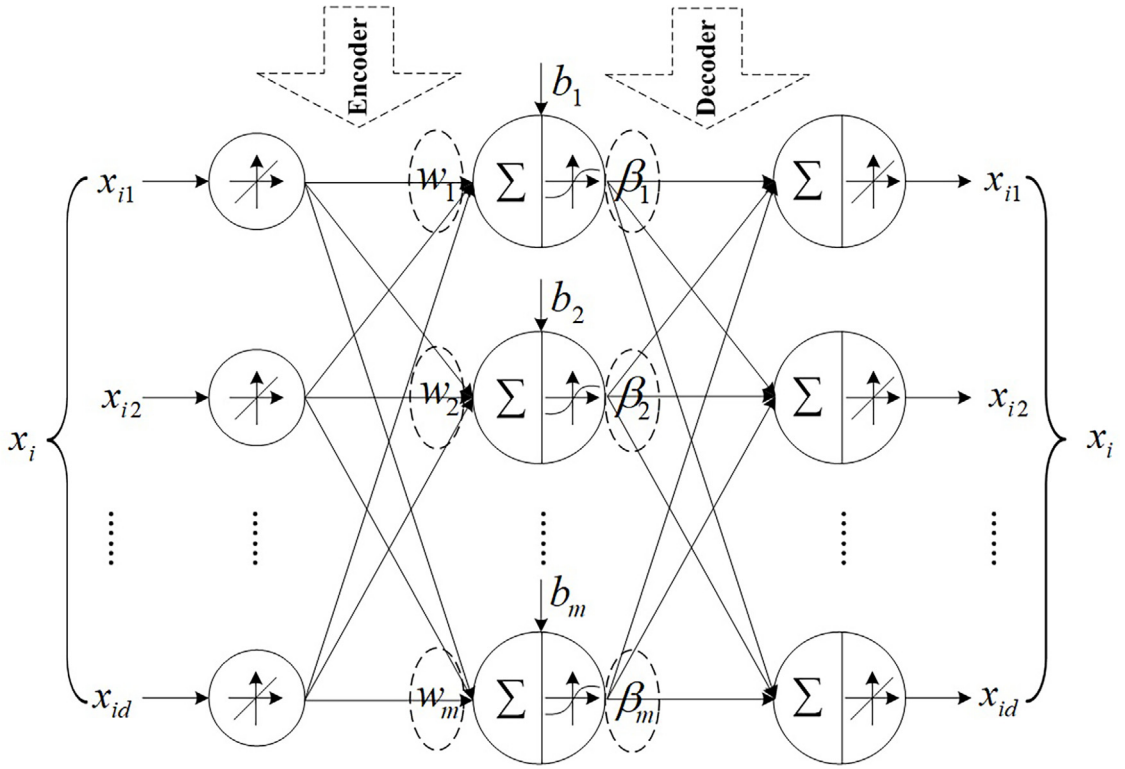

Fig. 2. The extreme learning machine autoencoder.

3.2. Binary imbalanced data classification based on diversity oversampling by generative adversarial network

A GAN [20] is an implicit probabilistic generation model that consists of two neural networks (Fig. 3), a generator G, and a discriminator D. The inputs **z** of the generator are samples obtained from a prior distribution P_{noise}, which is usually a

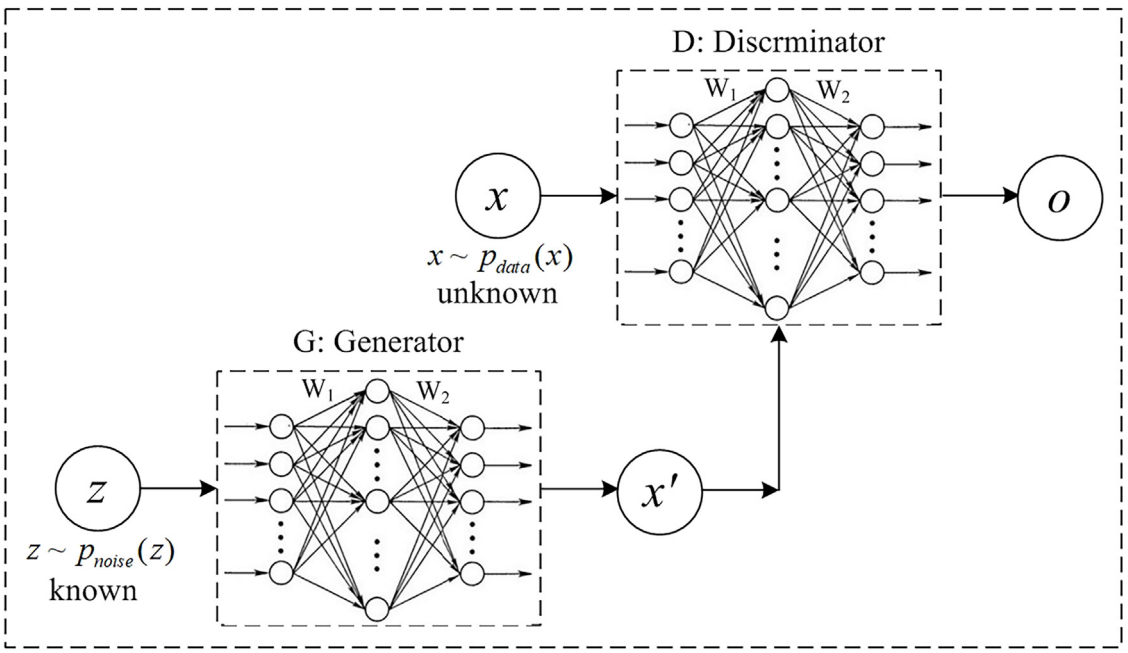

Fig. 3. The architecture of generative adversarial network.

Gaussian distribution. The outputs $\mathbf{x}' = \mathrm{G}(\mathbf{z})$ of the generator are fake samples used to mimic real samples to mislead the discriminator. The inputs of discriminator D include the fake data $\mathbf{x}'$ and real data $\mathbf{x}$, where $\mathbf{x}$ is sampled from an unknown real distribution P_d. The outputs of discriminator D are probability distributions, which indicate the probability that the inputs originated from the real distribution P_{data} or from the generated distribution P_{gen}. In other words, the goal of the discriminator is to distinguish between real and fake samples. GANs have excellent performance in several fields, such as computer vision, natural language processing, and audio recognition.

The appealing property of GAN is the adversarial training of the two networks, G and D. The goal of adversarial training is to find a Nash equilibrium. Let $J^{(\mathrm{D})}(\boldsymbol{\theta}^{(\mathrm{D})}, \boldsymbol{\theta}^{(\mathrm{G})})$ is the loss function for the discriminator D, and $J^{(\mathrm{G})}(\boldsymbol{\theta}^{(\mathrm{D})}, \boldsymbol{\theta}^{(\mathrm{G})})$ is the loss function for the generator G. A Nash equilibrium is a point $(\boldsymbol{\theta}^{(\mathrm{D})}, \boldsymbol{\theta}^{(\mathrm{G})})$, such that $J^{(\mathrm{D})}(\cdot,\cdot)$ obtains its maximum with respect to parameter $\boldsymbol{\theta}^{(\mathrm{D})}$, and $J^{(\mathrm{G})}(\cdot,\cdot)$ obtains its minimum with respect to parameter $\boldsymbol{\theta}^{(\mathrm{G})}$. The corresponding optimal model can be summarized as the following min–max model:

$$\begin{aligned} \min_{\mathrm{G}}\max_{\mathrm{D}} J(\mathrm{G},\mathrm{D}) &= \mathrm{E}_{\mathbf{x}\sim P_{data}}[\log(\mathrm{D}(\mathbf{x}))] + \mathrm{E}_{\mathbf{x}'\sim P_{gen}}[\log(1-\mathrm{D}(\mathbf{x}'))] \\ &= \mathrm{E}_{\mathbf{x}\sim P_{data}}[\log(\mathrm{D}(\mathbf{x}))] + \mathrm{E}_{\mathbf{z}\sim P_{noise}}[\log(1-\mathrm{D}(\mathrm{G}(\mathbf{z})))] \end{aligned} \tag{12}$$

With respect to the training of GAN, Goodfellow et al. [20] suggest to training GAN by alternating between k steps of optimizing D and one step of optimizing G. In addition, Goodfellow et al. also pointed out that since Eq. (12) may not provide sufficient gradient for G to learn well. Early in learning, when the performance of G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1-\mathrm{D}(\mathrm{G}(\mathbf{z})))$ saturates. Rather than training G to minimize $\log(1-\mathrm{D}(\mathrm{G}(\mathbf{z})))$, one can train G to maximize $\log(\mathrm{D}(\mathrm{G}(\mathbf{z})))$. The pseudo-code of the algorithm based on minibatch stochastic gradient descent for training GAN is given in Algorithm 3.

Algorithm 3: Minibatch stochastic gradient descent training of generative adversarial nets

Input: The training set $S_{tr} = \{\mathbf{x_i}, 1 \le i \le n\}$, the known noise prior distribution P_{noise}, the number of steps to apply to the discriminator k, and the iterative number t.

Output: The model parameters $(\theta^{(\mathrm{D})}, \theta^{(\mathrm{G})})$.

for $(i = 1; i \le t; i = i + 1)$ **do**
  for $(j = 1; j \le k; j = j + 1)$ **do**
    Sample minibatch of m noise samples $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m\}$ from noise prior P_{noise};
    Sample minibatch of m samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ from the training set S_{tr};
    Update the discriminator by ascending its stochastic gradient:

$$\bigtriangledown_{\theta^{(\mathrm{D})}} \frac{1}{m}\sum_{i=1}^{m}[\log \mathrm{D}(\mathbf{x}_i) + \log(1-\mathrm{D}(\mathrm{G}(\mathbf{z}_i))]$$

  end
  Sample minibatch of m noise samples $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m\}$ from noise prior P_{noise};
  Update the generator by descending its stochastic gradient:

$$\bigtriangledown_{\theta^{(\mathrm{G})}} \frac{1}{m}\sum_{i=1}^{m}\log(1-\mathrm{D}(\mathrm{G}(\mathbf{z}_i))$$

end
Return $(\theta^{(\mathrm{D})}, \theta^{(\mathrm{G})})$.

After training of GAN, the obtained G can be used to implicitly approximate the data distribution P_{data}, i.e., using the generated samples $\mathbf{x}'$ to approximate real samples $\mathbf{x} \sim P_{data}$. Motivated by this idea, we borrow technical route of GAN to address binary imbalanced data classification problem, and proposed another binary imbalanced data classification algorithm

(BIDC2). Similar to the BIDC1, the BIDC2 uses GAN to generate positive samples to balance the data sets and it also consists of three stages: (1) firstly, training the GAN on S_{tr}^{+}; (2) generating synthetic positive samples with the trained GAN to balance original data set; (3) training a classifier model on balanced data set and classifying testing samples. The pseudo-code of the BIDC2 is given in Algorithm 4.

Algorithm 4: The BIDC2 algorithm

Input: Imbalanced training set $S_{tr} = S_{tr}^{+} + S_{tr}^{-}$, imbalanced testing set $S_{te} = S_{te}^{+} + S_{te}^{-}$, the iterative number t.
Output: The classification results of $\mathbf{x} \in S_{te}$.

// Stage 1: training the GAN on S_{tr}^{+};
Call Algorithm 3 to train GAN model on S_{tr}^{+};
// Stage 2: generating synthetic positive samples with the trained GAN model;
$S_1^{+} = S_{tr}^{+}$;
for $(i = 1; i \leq t; i = i + 1)$ **do**
  Sample m' minibatch noises with size m from noise prior P_{noise};
  Input the m' minibatch noises into the generator of the trained GAN, and generate synthetic positive samples;
  Select informative positive samples from the synthetic ones by Silhouette-score and MMD-score, the set of selected positive samples is denoted by S_{gen}^{+};
  $S_{i+1}^{+} = S_i^{+} + S_{gen}^{+}$;
end
// Stage 3: training a classifier model on balanced data set and classifying testing samples;
$S_{tr}^{+} = S_{t+1}^{+}$;
$S_{tr} = S_{tr}^{+} + S_{tr}^{-}$;
Train a classifier on S_{tr}, and use the trained classifier to classify $\mathbf{x} \in S_{te}$;

Notes: (1) In the experiments, we set $m' = m$; (2) We calculate the MMD-score between S_i^{+} and each minibatch generated positive samples and calculate the Silhouette-score between each minibatch generated positive samples and the negative samples. Based on the calculation results, we select the minibatch S_{gen}^{+} with the highest score, the score is the sum of MMD-score add Silhouette-score.

4. Experimental results and analysis

To verify the effectiveness of the two algorithms BIDC1 and BIDC2, we conducted extensive experiments on 26 data sets to compare the two algorithms with 14 state-of-the-art algorithms, ROS [9], SMOTE [1], B-SMOTE [2], ADASYN [3], K-SMOTE [12], ANS [13], CCR [14], NRPSOS [15], C-SMOTE [16], SOMO [17], G-SMOTE [18], OUPS [19], AC-GAN [21], and MFC-GAN [25]. The 26 data sets include 1 artificial data set, 15 public testing data sets, and 10 application-oriented data sets. The artificial data set is a two-dimensional data set with two classes following Gaussian distributions. The mean vectors and covariance matrices of the two Gaussian distributions are listed in Table 1. The artificial data set is used to demonstrate the feasibility of the proposed approach and visualize the generated synthetic examples. The distribution of the data points of the artificial data set for the class-balanced scenario is shown in Fig. 4. The 15 public testing data sets include 10 UCI data sets[1] and 5 KEEL data sets[2]. The 10 application-oriented data sets include 7 software defect prediction data sets[3], and 3 liver data sets[4]. The basic information of the artificial data set and the 15 public testing data sets is listed in Table 2, and that of

[1] http://archive.ics.uci.edu/ml/index.php
[2] https://sci2s.ugr.es/keel/datasets.php
[3] http://promise.site.uottawa.ca/SERepository
[4] https://github.com/ShenData/data

Table 1
The mean vectors and covariance matrices of two Gaussian distributions.

i	μ_i	Σ_i
1	$(1.0, 1.0)^{\mathrm{T}}$	$\begin{bmatrix} 0.6 & -0.2 \\ -0.2 & 0.6 \end{bmatrix}$
2	$(2.5, 2.5)^{\mathrm{T}}$	$\begin{bmatrix} 0.2 & -0.1 \\ -0.1 & 0.2 \end{bmatrix}$

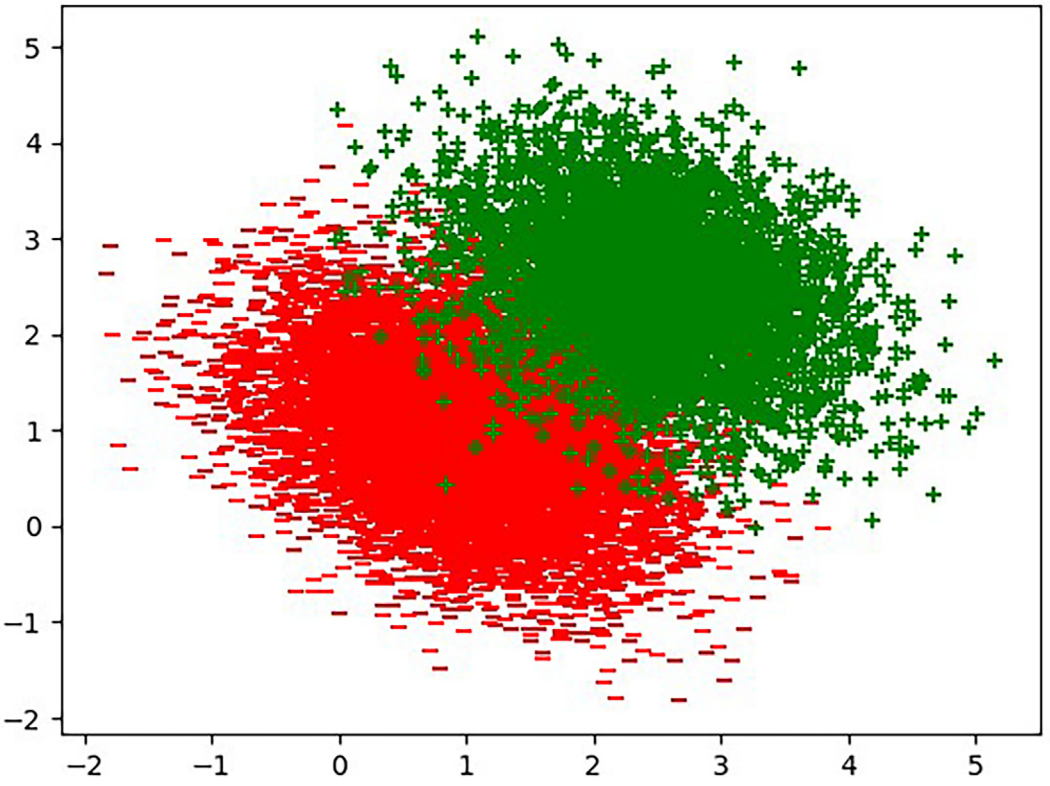

Fig. 4. The visualization of distribution of the artificial data set.

Table 2
The basic information of the artificial data set and the 15 public testing data sets.

Data sets	#Sample	#Attribute	#Minority	#Majority	**IR**
Artificial	10100	2	100	10000	100
Ecoli1	336	7	52	284	5.46
Ecoli2	310	7	26	284	10.92
Glass1	214	9	70	144	2.06
Glass2	179	9	35	144	4.11
Iris1	150	4	50	100	2.00
Iris2	125	4	25	100	4.00
ILPD1	345	6	145	200	1.38
ILPD2	272	6	72	200	2.78
Wine1	178	13	71	107	1.51
Wine2	142	13	35	107	3.06
Segment	2308	18	329	1979	6.02
Yeast3	1484	8	163	1321	8.10
Yeast4	1484	8	51	1430	28.04
Yeast6	1484	8	35	1449	41.40
Vowel0	988	13	90	898	9.98

Table 3
The basic information of the 10 data sets for two application scenarios.

Data sets	#Sample	#Attribute	#Minority	#Majority	IR
CM1	327	37	42	285	6.79
JM1	7782	21	1672	6110	3.65
MC1	1591	38	37	1554	42.00
MC2	125	39	44	81	1.84
PC1	1109	21	77	1032	13.40
KC2	522	21	107	415	3.88
KC3	194	39	36	158	4.39
Liver1	12400	5	200	12200	61
Liver2	14000	5	1000	13000	13
Liver3	13000	5	500	12500	25

the 10 application-oriented data sets is listed in Table 3. In these tables, #Sample is the number of samples in the data set, #Attribute is the number of attributes, #Minority is the number of minority samples, #Majority is the number of majority samples, and the IR = #Majority/#Minority.

All experiments are conducted on a PC platform with an Intel i5-6600 k CPU, 16G memory, and 1 TB hard disk. The operating system is 64-bit Windows 10. The experimental comparison metrics are the F-measure, G-mean, AUC-area [9], MMD-score [7], and Silhouette-score [8]. Let Positive and Negative represent the true labels of the positive and negative classes respectively, and Yes and No represent the predicted labels of positive and negative classes, then the confusion matrix of the binary imbalanced classification problem can be defined in Table 4.

Based on the confusion matrix, the Precision, Recall, and Specificity can be defined by Eq. (13), Eq. (14), and Eq. (15), respectively.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{15}$$

Based on the Precision and Recall, the F-measure can be defined by Eq. (16).

$$\text{F} - \text{measure} = \frac{(1+\beta)^2 \times \text{Recall} \times \text{Precision}}{\beta^2 \times \text{Recall} + \text{Precision}} \tag{16}$$

where β is a coefficient to adjust the relative importance of precision versus recall, in our experiments, we let $\beta = 1$.

Based on the Recall and Specificity, the G-mean can be defined by Eq. (17).

$$\text{G} - \text{mean} = \sqrt{\text{Recall} \times \text{Specificity}} \tag{17}$$

The AUC refers to the area under the curve of receiver operating characteristics (ROC) [9]. The ROC curve is a graph in 2-dimension space of TP-rate and FP-rate, which is formed by plotting TP-rate over FP-rate, the TP-rate and FP-rate are the vertical and horizontal axis of the 2-dimension space respectively.

4.1. Comparison of the 16 algorithms on the artificial data set and 15 public testing data sets

In this experiment, we compared the BIDC1 and BIDC2 with the 14 state-of-the-art algorithms. In the comparison of the BIDC1 and the 14 state-of-the-art algorithms, the programming environment consists of PyCharm Community Edition 2017.1.1, and Weka 3.9. The support vector machine is selected as the classifier. The activation function of ELMAE is the sigmoid function $s(x) = \frac{1}{1+e^{-x}}$. The selected hyperparameters for the different data sets are listed in Table 5, where #Hidden

Table 4
Confusion matrix of binary imbalanced classification problem.

True labels	Prediction labels	
	Yes	No
Positive	TP(True Positive)	FN(False Negative)
Negative	FP(False Positive)	TN(True Negative)

Table 5
The number of hidden layer neurons and the number of iterations.

Data sets	#Hidden nodes	#Iterations
Artificial	150	5
Ecoli1	10	2
Ecoli2	10	3
Glass1	10	1
Glass2	10	2
Iris1	5	1
Iris2	5	2
ILPD1	15	1
ILPD2	15	1
Wine1	20	1
Wine2	20	2
Segment	30	2
Yeast3	20	3
Yrast4	10	4
Yeast6	10	5
Vowel0	20	3

nodes and #Iterations are the number of hidden layer neurons and number of iterations respectively. In the comparison of the BIDC2 and the 14 state-of-the-art algorithms, the programming environment consists of PyCharm Community Edition 2017.1.1 and TensorFlow. The generator and discriminator of the GAN are single hidden layer feedforward neural networks, and the activation functions of hidden and output layers are $s(x) = \frac{1}{1+e^{-x}}$ and $t(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, respectively. The dimension $\mathrm{d}_{\mathbf{z}}$ of the noise variable $\mathbf{z}$ and the number of hidden nodes of generator G and discriminator D are listed in Table 6. We use the ADAM algorithm to train the two networks with the following parameters: $\alpha = 0.001, \beta_1 = 0.900, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

We use a grid search strategy to select the appropriate hyperparameters based on the highest performance. For example, the numbers of hidden nodes of the encoder and decoder networks in the BIDC1 are listed in Table 5, and the numbers of hidden nodes of the generator and discriminator networks in the BIDC2 are listed in Table 6. For each data set, we determine the suitable numbers of hidden nodes of the neural networks using a grid search strategy with the same interval [50,150]. The suitable number of iterations in BIDC1 for each data are selected in the same manner in [1,15], and the two intervals are determined empirically.

We also analyze the influence of the hyperparameters on the results. The number of hidden nodes determines the size of the network and affects overfitting or underfitting of the model. If the data set is large or complex, and there are few hidden nodes, underfitting can occur. Overfitting is likely if the data set is very small with many hidden nodes. Larger or more complex data sets require more iterations to ensure high performance, whereas smaller data sets can learn the distribution well, resulting in less training time with fewer iterations.

The experimental results on the artificial data set and 15 public data sets are summarized in Tables 7–11. It is observed that the proposed methods BIDC1 and BIDC2 outperform most state-of-the-art methods on all metrics. Especially in Table 10, our methods outperform all state-of-the-art methods on the MMD-score. It is also found that BIDC2 is most effective and achieves most maximum values. Its superior performance against BIDC1 suggest that GAN is a stronger generative model

Table 6
The dimension of noise variable $\mathbf{z}$ and the number of hidden nodes of generator G and discriminator D.

Data sets	$\mathbf{d_z}$	#Hidden nodes of G	#Hidden nodes of D
Artificial	100	100	100
Ecoli1	55	70	35
Ecoli2	35	50	20
Glass1	35	90	45
Glass2	25	70	35
Iris1	20	25	15
Iris2	20	25	15
ILPD1	50	50	20
ILPD2	25	35	20
Wine1	130	65	40
Wine2	130	65	40
Segment	150	75	50
Yeast3	100	50	30
Yeast4	100	50	30
Yeast6	100	50	30
Vowel0	120	50	40

Table 7
The experimental results compared with 14 state-of-the-art methods on the 1 artificial data set and 15 public testing data sets on F-measure.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
Artificial	0.243	0.433	0.332	0.623	0.144	0.584	0.234	0.561	0.664	0.804	0.581	0.550	0.621	0.683	0.714	0.783
Ecoli1	0.621	0.625	0.674	0.718	0.756	0.797	0.616	0.800	0.796	0.000	0.710	0.788	0.652	0.688	0.812	**0.833**
Ecoli2	0.476	0.417	0.500	0.556	0.825	0.821	0.700	**0.852**	0.819	0.000	0.722	0.741	0.774	0.000	0.485	0.572
Glass1	0.437	0.505	0.609	0.547	0.505	0.530	0.552	0.630	0.556	0.129	0.569	0.551	0.610	0.619	0.633	**0.658**
Glass2	0.430	0.483	0.572	0.538	0.751	0.639	0.455	0.501	0.511	0.000	0.065	0.671	0.734	0.000	**0.769**	0.690
Iris1	0.643	0.658	0.286	0.712	0.501	0.000	0.492	0.501	0.501	0.505	0.505	0.501	0.752	0.764	0.720	**0.774**
Iris2	0.458	0.471	0.502	0.536	0.000	0.528	0.581	**0.901**	0.476	0.000	0.418	0.649	0.663	0.240	0.625	0.548
ILPD1	0.617	0.602	0.532	0.633	0.285	0.000	0.000	0.668	0.393	0.322	0.415	0.285	0.359	0.586	0.635	**0.705**
ILPD2	0.524	0.509	0.488	0.554	0.669	0.669	0.132	**0.672**	0.105	0.000	0.299	0.669	0.075	0.099	0.600	0.644
Wine1	0.880	0.846	0.905	0.899	0.764	0.766	0.726	0.771	0.761	0.764	0.764	0.766	0.923	0.933	0.923	**0.938**
Wine2	0.872	0.938	0.991	0.984	0.442	0.891	0.671	0.891	0.119	0.891	0.427	0.365	0.921	0.891	**0.997**	0.993
Segment	0.982	0.991	0.993	0.993	0.741	0.722	0.714	0.716	0.725	0.825	0.767	0.724	0.743	0.523	0.995	**0.998**
Yeast3	0.665	0.669	0.732	0.708	0.767	0.739	0.728	0.780	0.743	0.000	0.717	0.744	0.571	0.764	0.717	**0.784**
Yeast4	0.170	0.467	0.504	0.500	0.000	**0.942**	0.000	0.000	0.000	0.000	0.000	0.739	0.031	0.031	0.514	0.530
Yeast6	0.133	0.510	0.458	0.469	0.000	0.052	0.283	0.113	0.000	0.000	0.454	0.000	0.029	0.000	0.534	**0.551**
Vowel0	0.878	0.809	0.920	0.923	0.918	0.000	0.837	0.879	0.893	0.000	0.845	0.867	0.540	0.733	0.939	**0.955**

Table 8
The experimental results compared with 14 state-of-the-art methods on the 1 artificial data set and 15 public testing data sets on G-mean.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
Artificial	0.391	0.454	0.462	0.343	0.541	0.590	0.313	0.790	0.733	0.914	0.642	0.761	0.633	0.704	0.922	0.854
Ecoli1	0.862	0.927	0.940	0.971	0.591	0.636	0.679	0.640	0.367	0.000	0.727	0.681	0.802	0.781	0.905	**0.974**
Ecoli2	0.896	0.898	0.908	0.927	0.664	0.636	0.516	0.743	0.458	0.000	0.773	0.607	0.820	0.000	0.912	**0.946**
Glass1	0.616	0.619	0.667	0.683	0.721	0.624	0.565	0.619	0.609	0.216	0.536	0.669	0.689	0.635	0.700	**0.733**
Glass2	0.695	0.670	0.717	0.754	0.663	0.250	0.333	0.288	0.591	0.751	0.226	0.107	0.725	0.000	**0.928**	0.861
Iris1	0.704	0.704	0.722	0.749	0.509	0.000	0.632	0.509	0.509	0.509	0.516	0.509	0.819	0.829	0.794	**0.834**
Iris2	0.664	0.693	0.706	0.690	0.000	0.143	0.114	**0.833**	0.210	0.000	0.287	0.178	0.716	0.451	0.721	0.715
ILPD1	0.316	0.311	0.397	0.445	0.286	0.000	0.000	0.153	0.295	0.000	0.163	0.044	0.434	0.560	0.549	**0.563**
ILPD2	0.672	0.641	0.677	0.692	0.685	0.108	0.045	0.140	0.131	0.000	0.205	0.154	0.233	0.000	0.706	**0.711**
Wine1	0.887	0.865	0.894	0.906	0.479	0.484	0.000	0.484	0.479	0.492	0.498	0.488	0.914	0.921	0.926	**0.935**
Wine2	0.939	0.962	0.980	0.971	0.737	0.000	0.275	0.000	0.370	0.000	0.341	0.547	0.000	0.000	**0.998**	0.984
Segment	0.982	0.990	0.994	0.992	0.545	0.473	0.447	0.448	0.458	0.000	0.758	0.935	0.949	0.490	0.996	**0.999**
Yeast3	0.657	0.662	0.695	0.686	0.620	0.526	0.534	0.627	0.437	0.000	0.560	0.548	0.652	0.000	0.703	**0.775**
Yeast4	0.314	0.620	0.577	0.641	0.000	**0.933**	0.000	0.000	0.000	0.000	0.000	0.773	0.139	0.139	0.685	0.691
Yeast6	0.266	0.584	0.579	0.615	0.000	0.148	0.366	0.218	0.000	0.000	0.545	0.035	0.230	0.000	0.632	**0.668**
Vowel0	0.890	0.865	0.934	0.946	0.914	0.000	0.843	0.857	0.876	0.000	0.838	0.835	0.556	0.780	0.952	**0.967**

Table 9
The experimental results compared with 14 state-of-the-art methods on the 1 artificial data set and 15 public testing data sets on AUC-area.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
Artificial	0.414	0.600	0.614	0.562	0.274	0.671	0.552	0.701	0.762	0.911	0.701	0.773	0.672	0.702	0.833	0.924
Ecoli1	0.862	0.840	0.887	0.891	0.709	0.738	0.703	0.743	0.709	0.500	0.772	0.729	0.802	0.798	0.893	**0.920**
Ecoli2	0.904	0.912	0.926	0.918	0.785	0.750	0.658	0.787	0.680	0.500	0.709	0.653	0.883	0.431	0.887	**0.952**
Glass1	0.690	0.705	0.683	0.731	0.800	0.724	0.703	0.731	0.721	0.515	0.643	0.745	0.699	0.701	0.814	**0.846**
Glass2	0.741	0.764	0.876	0.890	0.811	0.508	0.447	0.535	0.610	0.815	0.528	0.512	0.803	0.500	0.931	**0.945**
Iris1	0.725	0.733	0.767	0.758	0.830	0.500	0.810	0.820	0.820	**0.840**	0.820	0.820	0.835	0.839	0.800	0.827
Iris2	0.700	0.693	0.706	0.718	0.500	0.400	0.477	**0.868**	0.348	0.500	0.460	0.492	0.784	0.544	0.725	0.720
ILPD1	0.550	0.533	0.596	0.666	0.565	0.500	0.500	0.508	0.585	0.500	0.490	0.513	0.552	0.599	0.670	**0.691**
ILPD2	0.680	0.645	0.682	0.701	0.510	0.508	0.505	0.513	0.508	**0.815**	0.498	0.488	0.518	0.515	0.713	0.750
Wine1	0.893	0.869	0.908	0.894	0.613	0.671	0.500	0.671	0.661	0.675	0.707	0.552	0.937	0.962	0.929	**0.966**
Wine2	0.895	0.915	0.982	0.933	0.609	0.500	0.500	0.500	0.403	0.500	0.402	0.500	0.500	0.500	**0.997**	0.989
Segment	0.987	0.991	0.991	0.993	0.652	0.610	0.604	0.599	0.592	0.500	0.746	0.619	0.936	0.949	0.996	**0.998**
Yeast3	0.753	0.757	0.772	0.769	0.693	0.645	0.651	0.707	0.624	0.500	0.625	0.638	0.710	0.494	0.784	**0.825**
Yeast4	0.564	0.612	0.678	0.651	0.500	**0.938**	0.500	0.500	0.500	0.500	0.500	0.819	0.505	0.505	0.693	0.736
Yeast6	0.558	0.644	0.619	0.658	0.500	0.509	0.571	0.518	0.500	0.500	0.657	0.500	0.493	0.496	0.682	**0.704**
Vowel0	0.834	0.854	0.886	0.906	0.925	0.500	0.846	0.872	0.851	0.500	0.841	0.848	0.760	0.803	0.915	**0.958**

Table 10
The experimental results compared with 14 state-of-the-art methods on the 1 artificial data set and 15 public testing data sets on MMD-score.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
Artificial	0.025	0.026	0.394	0.397	1.096	0.197	0.260	0.251	0.970	0.761	0.030	0.126	1.181	1.436	0.837	1.449
Ecoli1	0.034	0.094	0.063	0.056	0.071	0.054	0.033	0.070	0.042	0.025	0.060	0.035	0.080	0.033	0.109	**6.233**
Ecoli2	0.064	0.090	0.070	0.142	0.052	0.068	0.082	0.309	0.150	0.114	0.111	0.137	0.070	0.170	0.103	**6.124**
Glass1	0.046	0.026	0.020	0.032	0.019	0.016	0.031	0.042	0.064	0.029	0.033	0.061	0.057	0.019	**7.729**	7.720
Glass2	0.031	0.032	0.080	0.106	0.087	0.044	0.092	0.333	0.116	0.059	0.060	0.126	0.070	0.031	0.186	**5.243**
Iris1	0.019	0.041	0.028	0.017	0.060	0.088	0.059	0.030	0.043	0.063	0.034	0.027	0.037	0.067	0.233	**7.383**
Iris2	0.047	0.040	0.019	0.048	0.057	0.045	0.046	2.573	0.059	0.188	0.039	0.100	0.019	0.042	0.142	**5.778**
ILPD1	0.041	0.020	0.020	0.015	0.012	0.023	0.012	0.040	0.020	0.022	0.015	0.019	0.008	0.021	**1.460**	0.829
ILPD2	0.028	0.044	0.088	0.019	0.022	0.023	0.028	0.224	0.044	0.034	0.020	0.123	0.034	0.032	0.056	**5.518**
Wine1	0.039	0.053	0.031	0.037	0.047	0.040	0.053	0.032	0.034	0.056	0.038	0.027	0.037	0.038	0.402	**1.596**
Wine2	0.018	0.011	0.018	0.026	0.021	0.018	0.015	0.020	0.024	0.027	0.029	0.023	0.019	0.014	0.060	**3.496**
Segment	0.005	0.007	0.005	0.007	0.007	0.014	0.015	0.009	0.016	0.005	0.009	0.005	0.009	0.012	0.594	**4.765**
Yeast3	0.014	0.017	0.019	0.023	0.014	0.018	0.015	0.072	0.020	0.016	0.009	0.013	0.015	0.013	0.036	**5.145**
Yeast4	0.067	0.101	0.036	0.042	0.048	0.056	0.051	0.661	0.086	0.031	0.036	0.042	0.081	0.044	0.053	**5.313**
Yeast6	0.061	0.035	0.050	0.045	0.062	0.066	0.040	0.281	0.069	0.124	0.052	0.077	0.046	0.093	0.122	**6.279**
Vowel0	0.024	0.040	0.013	0.040	0.018	0.029	0.018	0.052	0.068	0.009	0.026	0.021	0.074	0.021	0.110	**3.690**

Table 11
The experimental results compared with 14 state-of-the-art methods on the 1 artificial data set and 15 public testing data sets on Silhouette-score.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
Artificial	0.435	0.449	0.394	0.380	0.438	0.537	0.352	0.548	0.480	0.575	0.450	0.442	0.382	0.592	0.641	0.692
Ecoli1	0.256	0.260	0.344	0.175	0.287	0.329	0.146	0.313	0.142	0.101	0.155	0.305	0.253	0.258	0.352	**0.396**
Ecoli2	0.276	0.325	0.305	0.267	0.383	0.356	0.160	0.365	0.320	0.076	0.169	0.295	0.268	0.248	0.387	**0.439**
Glass1	0.133	0.140	0.144	0.134	0.151	0.153	0.095	0.149	0.163	0.165	0.067	0.151	0.214	0.109	0.388	**0.463**
Glass2	0.012	0.009	0.016	0.010	0.136	0.029	0.010	0.079	0.025	-0.028	0.026	0.016	0.209	0.207	0.113	**0.246**
Iris1	0.662	0.668	0.629	0.629	0.664	0.629	0.478	0.666	0.661	0.669	0.546	0.672	0.535	0.616	0.723	**0.882**
Iris2	0.003	0.001	0.002	0.003	-0.019	0.010	0.000	**0.428**	0.027	-0.019	0.021	0.017	0.290	0.151	0.011	0.245
ILPD1	0.012	0.013	0.011	0.012	0.047	0.005	0.007	0.031	0.021	0.057	0.008	0.025	0.094	0.044	0.175	**0.184**
ILPD2	0.013	0.023	0.026	0.016	0.092	0.038	0.013	0.055	0.053	-0.021	0.015	0.039	0.234	0.078	0.164	**0.239**
Wine1	0.178	0.182	0.158	0.157	0.157	0.195	0.131	0.188	0.177	0.209	0.155	0.209	0.300	0.205	0.299	**0.397**
Wine2	0.012	0.012	0.034	0.010	**0.200**	0.001	0.018	0.001	0.019	0.001	0.016	0.023	0.187	0.022	0.022	0.144
Segment	0.185	0.188	0.199	0.210	0.216	0.208	0.106	0.187	0.175	-0.057	0.081	0.216	0.264	0.146	0.613	**0.709**
Yeast3	0.155	0.171	0.120	0.121	**0.226**	0.187	0.105	0.223	0.215	0.045	0.106	0.187	0.209	0.066	0.141	0.197
Yeast4	0.196	0.218	0.267	0.193	0.181	0.335	0.136	**0.387**	0.333	0.181	0.153	0.209	0.351	0.136	0.291	0.347
Yeast6	0.299	0.338	0.438	0.261	0.180	0.453	0.209	0.445	0.457	0.180	0.216	0.290	0.018	0.138	0.468	**0.542**
Vowel0	0.089	0.093	0.371	0.301	0.259	0.097	0.071	0.093	0.093	0.097	0.072	0.109	0.227	0.216	0.486	**0.517**

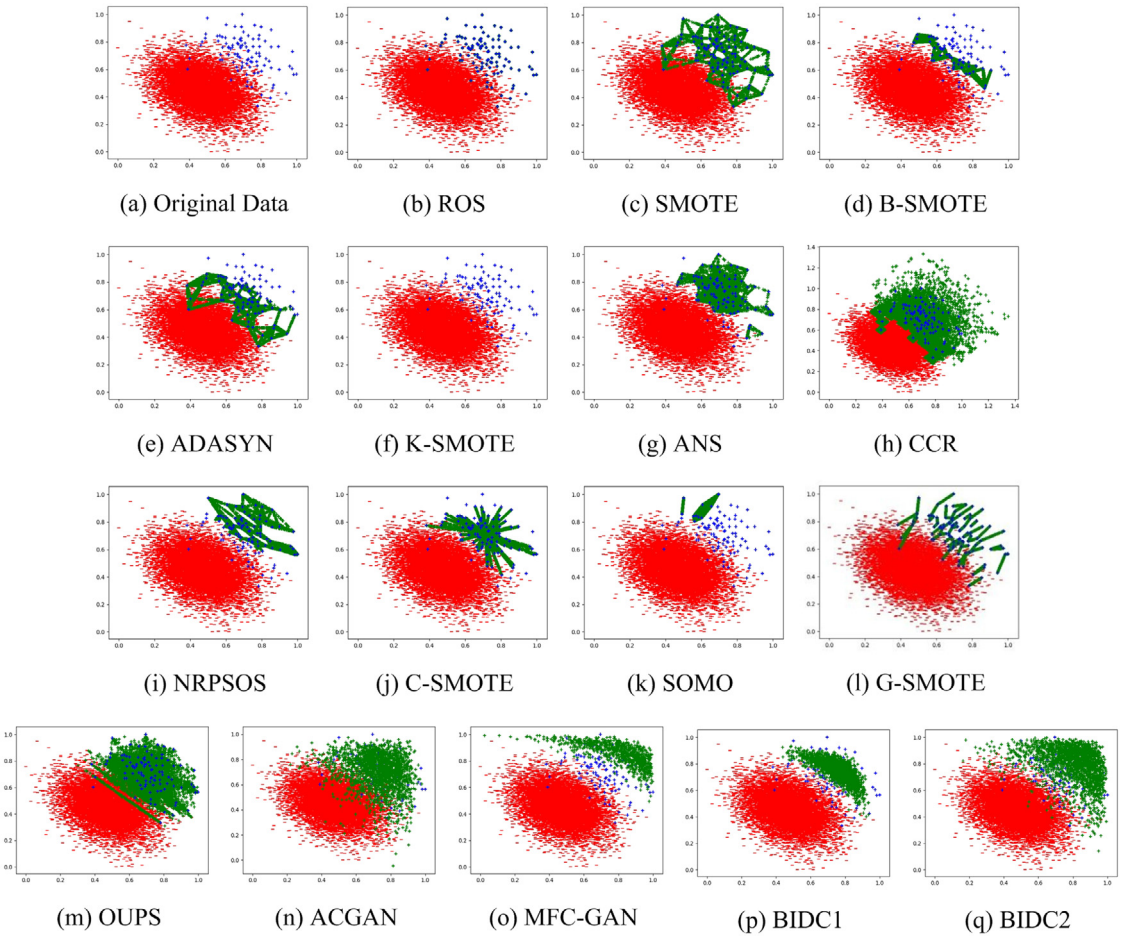

Fig. 5. The visualization of the generated positive samples with different methods on artificial data set.

Table 12
The number of hidden layer neurons and the number of iterations.

Data sets	#Hidden nodes	#Iterations
CM1	40	2
JM1	20	2
MC1	45	5
MC2	40	1
PC1	25	4
KC2	20	2
KC3	35	2
Liver1	10	10
Liver2	20	6
Liver3	10	7

than ELMAE and can better learn the distribution of positive samples. In addition, it is also observed that some methods with extremely low MMD-score and Silhouette-score exhibit low performances. For example, SOMO on the Ecoli1 data set and ANS and CCR on the Glass2 data set have low MMD-score and Silhouette-score with low performance. Since the BIDC1 and BIDC2 uses the Silhouette-score to ensure no examples are generated within overlapping decision regions, this oversampling mechanism facilitates learning the true decision boundary due to the higher separability between the two classes. This finding is illustrated in Fig. 5(p) and (q) for the artificial data set. Furthermore, a comparison of the distribution of the generated samples and the original positive samples in Fig. 5(p) and 5(q) with the real balanced distribution in Fig. 4 indicates that the BIDC1 and BIDC2 improve the overall classification performance although there may be small differences between

Table 13
The experimental results compared with 14 state-of-the-art methods on the 10 application-oriented data sets on F-measure.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
CM1	0.333	0.343	0.286	0.415	0.110	0.400	0.000	**0.974**	0.110	0.000	0.020	0.669	0.742	0.044	0.462	0.502
JM1	0.642	0.630	0.647	0.669	0.756	0.001	0.000	0.005	0.001	0.757	0.001	0.003	0.771	0.387	0.694	**0.786**
MC1	0.335	0.421	0.375	0.456	0.000	**0.694**	0.000	0.000	0.682	0.000	0.033	0.672	0.000	0.043	0.517	0.460
MC2	0.000	0.041	0.030	0.154	0.576	0.044	**0.685**	0.252	0.044	0.305	0.112	0.237	0.568	0.553	0.211	0.236
PC1	0.526	0.538	0.556	0.601	0.071	0.010	**0.668**	0.329	0.093	0.574	0.030	0.139	0.072	0.071	0.625	0.611
KC2	0.376	0.342	0.400	0.398	**0.778**	0.018	0.671	0.023	0.145	0.689	0.014	0.138	0.360	0.439	0.504	0.527
KC3	0.012	0.407	0.392	0.410	0.140	0.022	0.013	0.268	0.133	0.000	0.024	0.598	0.000	0.000	0.530	**0.644**
Liver1	0.000	0.785	0.896	0.823	0.000	0.727	0.720	0.720	0.642	0.000	0.645	0.764	0.000	0.019	0.921	**0.958**
Liver2	0.000	0.406	0.664	0.750	0.000	0.730	0.732	0.730	0.697	0.000	0.662	0.740	0.008	0.010	0.825	**0.871**
Liver3	0.000	0.842	0.883	0.793	0.000	0.735	0.727	0.730	0.505	0.000	0.656	0.739	0.010	0.000	0.904	**0.934**

Table 14
The experimental results compared with 14 state-of-the-art methods on the 10 application-oriented data sets on G-mean.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
CM1	0.667	0.482	0.688	0.657	0.129	0.098	0.000	**0.958**	0.000	0.000	0.072	0.000	0.749	0.154	0.690	0.724
JM1	0.814	0.808	0.793	0.802	0.769	0.008	0.000	0.065	0.008	0.715	0.011	0.000	0.779	0.558	0.821	**0.852**
MC1	0.541	0.563	0.533	0.527	0.000	0.339	0.000	0.000	0.248	0.000	0.123	0.000	0.000	0.164	**0.625**	0.567
MC2	0.000	0.145	0.126	0.330	**0.642**	0.071	0.000	0.452	0.071	0.434	0.207	0.157	0.651	0.645	0.333	0.417
PC1	0.618	0.646	0.661	0.640	0.094	0.034	0.000	0.458	0.000	**0.959**	0.038	0.000	0.197	0.197	0.692	0.686
KC2	0.493	0.579	0.511	0.556	**0.805**	0.044	0.000	0.097	0.073	0.596	0.053	0.022	0.479	0.565	0.600	0.652
KC3	0.508	0.546	0.539	0.558	**0.832**	0.034	0.036	0.406	0.130	0.000	0.086	0.049	0.000	0.000	0.588	0.737
Liver1	0.000	0.612	0.581	0.736	0.000	0.504	0.512	0.475	0.687	0.000	0.568	0.629	0.000	0.265	0.884	**0.893**
Liver2	0.000	0.597	0.624	0.713	0.000	0.509	0.539	0.513	0.746	0.000	0.588	0.543	0.070	0.100	0.853	**0.906**
Liver3	0.000	0.643	0.708	0.758	0.000	0.529	0.528	0.508	0.766	0.000	0.566	0.542	0.077	0.000	0.897	**0.924**

Table 15
The experimental results compared with 14 state-of-the-art methods on the 10 application-oriented data sets on AUC-area.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
CM1	0.682	0.691	0.590	0.715	0.535	0.496	0.498	**0.961**	0.498	0.500	0.505	0.500	0.855	0.508	0.747	0.772
JM1	0.814	0.808	0.823	0.837	0.864	0.500	0.500	0.503	0.500	0.772	0.500	0.500	0.874	0.584	0.850	**0.893**
MC1	0.578	0.609	0.641	0.618	0.500	0.562	0.500	0.500	0.546	0.500	0.510	0.500	0.500	0.511	0.702	**0.717**
MC2	0.500	0.510	0.507	0.524	**0.756**	0.513	0.500	0.605	0.519	0.593	0.538	0.518	0.683	0.647	0.556	0.604
PC1	0.622	0.653	0.680	0.695	**0.964**	0.524	0.500	0.593	0.500	0.500	0.506	0.500	0.518	0.518	0.686	0.735
KC2	0.611	0.661	0.623	0.592	**0.747**	0.505	0.500	0.505	0.513	0.671	0.502	0.501	0.608	0.643	0.677	0.710
KC3	0.598	0.582	0.621	0.609	0.547	0.494	0.500	0.573	0.534	0.500	0.500	0.503	0.500	0.500	0.634	**0.743**
Liver1	0.500	0.772	0.846	0.865	0.500	0.626	0.620	0.613	0.732	0.500	0.598	0.692	0.500	0.480	0.961	**0.969**
Liver2	0.500	0.714	0.785	0.851	0.500	0.630	0.640	0.631	0.697	0.500	0.605	0.649	0.495	0.460	0.926	**0.948**
Liver3	0.500	0.803	0.869	0.864	0.500	0.635	0.632	0.627	0.637	0.500	0.593	0.646	0.498	0.498	0.885	**0.914**

Table 16
The experimental results compared with 14 state-of-the-art methods on the 10 application-oriented data sets on MMD-score.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
CM1	0.034	0.151	0.033	0.033	0.043	0.060	0.054	2.380	0.039	0.023	0.106	0.045	0.086	0.055	0.055	**4.406**
JM1	0.001	0.001	0.001	0.171	0.001	0.001	0.002	0.245	0.001	0.001	0.000	0.001	0.002	0.001	0.004	**3.759**
MC1	0.131	0.036	0.031	0.051	0.159	0.097	0.041	0.085	0.044	0.061	0.048	0.042	0.052	0.053	0.218	**4.201**
MC2	0.033	0.059	0.061	0.134	0.037	0.037	0.099	0.437	0.040	0.080	0.038	0.054	0.038	0.058	0.202	**3.040**
PC1	0.010	0.012	0.008	0.009	0.016	0.008	0.013	0.615	0.040	0.021	0.010	0.008	0.006	0.007	0.022	**5.024**
KC2	0.017	0.026	0.007	0.007	0.016	0.014	0.014	0.228	0.011	0.010	0.014	0.026	0.009	0.013	0.033	**4.411**
KC3	0.061	0.057	0.084	0.069	0.083	0.091	0.110	0.566	0.047	0.085	0.071	0.077	0.050	0.138	0.133	**4.202**
Liver1	0.026	0.009	0.006	0.019	0.019	0.010	0.007	0.170	0.019	0.015	0.018	0.009	0.011	0.005	**7.738**	5.880
Liver2	0.002	0.002	0.005	0.002	0.002	0.002	0.001	0.021	0.002	0.002	0.008	0.001	0.003	0.002	**7.742**	6.154
Liver3	0.011	0.010	0.003	0.006	0.006	0.009	0.003	0.046	0.002	0.003	0.009	0.009	0.002	0.005	**7.741**	5.967

Table 17
The experimental results compared with 14 state-of-the-art methods on the 10 application-oriented data sets on Silhouette-score.

Data sets	ROS	SMOTE	B-SMOTE	ADASYN	K-SMOTE	ANS	CCR	NRPSOS	C-SMOTE	SOMO	G-SMOTE	OUPS	AC-GAN	MFC-GAN	BIDC1	BIDC2
CM1	0.049	0.052	0.064	0.051	0.249	0.073	0.041	0.436	0.129	0.078	0.058	0.071	0.168	0.213	0.446	**0.498**
JM1	0.034	0.033	0.033	0.021	**0.476**	0.058	0.009	0.141	0.060	0.391	0.083	0.078	0.121	0.282	0.436	0.450
MC1	0.059	0.068	0.139	0.065	0.054	0.128	0.044	0.054	0.090	0.054	0.049	0.064	0.062	0.244	0.154	**0.258**
MC2	0.061	0.060	0.045	0.038	0.225	0.044	0.052	0.225	0.094	0.213	0.070	0.114	0.341	0.128	0.176	**0.378**
PC1	0.081	0.078	0.084	0.066	0.463	0.105	0.054	0.197	0.149	0.405	0.141	0.092	0.316	0.644	0.484	**0.596**
KC2	0.163	0.174	0.139	0.124	**0.561**	0.239	0.046	0.320	0.239	0.433	0.167	0.262	0.246	0.297	0.484	0.527
KC3	0.040	0.052	0.041	0.037	0.344	0.064	0.040	0.225	0.077	0.096	0.060	0.071	0.086	0.259	0.258	**0.478**
Liver1	0.074	0.082	0.091	0.054	-0.130	0.147	0.059	0.148	0.173	-0.130	0.019	0.068	0.143	0.534	**0.972**	0.788
Liver2	0.065	0.067	0.051	0.040	-0.088	0.121	0.052	0.086	0.174	-0.088	0.018	0.086	0.230	0.301	**0.878**	0.698
Liver3	0.076	0.075	0.059	0.048	-0.116	0.122	0.057	0.107	0.144	-0.116	0.018	0.085	0.430	0.162	**0.937**	0.754

Table 18
The statistical analysis of the experimental results compared with 14 state-of-the-art methods on 10 application-oriented data sets on F-measure.

Datasets	p-value1	p-value2	p-value3	p-value4	p-value5	p-value6	p-value7	p-value8	p-value9	p-value10	p-value11	p-value12	p-value13	p-value14
CM1	1.05e-13	2.46e-15	3.45e-13	6.04e-14	7.37e-11	4.52e-22	3.96e-11	3.41e-07	7.25e-11	1.96e-11	1.70e-11	1.17e-04	2.10e-05	2.64e-11
JM1	1.18e-17	8.70e-15	3.40e-15	6.26e-16	2.84e-11	5.82e-10	1.23e-10	3.54e-10	5.69e-10	5.24e-09	3.68e-10	2.15e-10	4.72e-07	3.05e-11
MC1	5.94e-14	2.05e-11	4.02e-12	1.34e-06	1.99e-11	9.10e-06	3.73e-11	2.33e-11	1.90e-05	2.24e-11	1.90e-11	1.29e-05	1.92e-11	9.33e-11
MC2	1.03e-13	5.05e-16	7.71e-14	9.43e-17	7.66e-06	1.64e-12	1.21e-06	1.47e-02	8.31e-14	1.07e-01	4.51e-18	7.43e-04	1.93e-06	1.47e-05
PC1	9.70e-13	1.08e-17	6.74e-16	3.52e-06	1.53e-11	1.11e-10	5.05e-01	2.97e-12	7.34e-11	3.04e-12	6.39e-11	5.23e-11	2.86e-11	8.05e-11
KC2	2.95e-18	1.09e-14	1.19e-16	6.11e-16	3.73e-05	2.76e-11	6.51e-04	3.21e-10	8.44e-13	5.03e-04	3.26e-10	2.35e-11	2.66e-16	4.39e-21
KC3	8.41e-11	5.73e-14	7.66e-11	1.15e-14	1.89e-10	1.15e-10	7.75e-11	4.10e-11	3.00e-11	3.68e-10	1.96e-10	6.50e-13	7.64e-11	4.28e-10
Liver1	8.85e-10	2.12e-12	1.92e-13	2.47e-15	6.57e-10	1.02e-14	1.99e-14	4.77e-14	6.01e-12	7.56e-10	7.00e-13	7.44e-14	1.27e-09	5.23e-10
Liver2	6.67e-10	2.16e-11	8.38e-13	5.65e-14	5.05e-10	2.93e-14	2.34e-16	1.12e-13	5.59e-15	3.27e-10	2.14e-11	1.84e-14	4.13e-10	2.50e-10
Liver3	7.67e-10	1.73e-23	2.55e-20	4.85e-18	3.10e-10	1.44e-14	4.22e-14	6.14e-16	5.47e-11	4.74e-10	4.90e-13	3.56e-14	8.50e-10	4.79e-10

Table 19
The statistical analysis of the experimental results compared with 14 state-of-the-art methods on 10 application-oriented data sets on G-mean.

Datasets	p-value1	p-value2	p-value3	p-value4	p-value5	p-value6	p-value7	p-value8	p-value9	p-value10	p-value11	p-value12	p-value13	p-value14
CM1	1.12e-08	3.56e-09	9.83e-09	6.04e-10	6.43e-09	6.19e-09	7.02e-09	8.01e-09	6.26e-09	6.67e-09	6.31e-09	6.80e-09	2.03e-08	6.98e-09
JM1	1.59e-08	6.39e-08	3.00e-08	8.72e-08	2.73e-08	7.30e-09	6.99e-09	6.63e-09	6.09e-09	8.01e-09	7.29e-09	6.86e-09	9.81e-09	6.21e-09
MC1	2.86e-08	8.41e-03	6.75e-07	1.35e-09	6.05e-09	4.35e-09	5.90e-09	5.45e-09	4.63e-09	6.66e-09	8.45e-09	7.97e-09	6.43e-09	5.31e-09
MC2	6.23e-09	8.66e-09	1.04e-08	2.68e-09	1.41e-08	7.24e-09	7.32e-09	5.38e-07	6.95e-09	2.50e-05	5.96e-09	6.46e-09	6.96e-09	3.17e-09
PC1	1.23e-08	1.02e-09	3.52e-10	3.91e-09	8.28e-09	6.55e-09	6.91e-09	5.81e-09	8.00e-09	9.50e-09	6.51e-09	5.76e-09	6.29e-09	5.80e-09
KC2	4.65e-09	1.78e-09	8.05e-09	2.74e-08	7.36e-09	7.32e-09	7.61e-09	5.16e-09	5.29e-09	5.56e-08	7.16e-09	8.50e-09	6.46e-09	1.50e-08
KC3	5.94e-09	4.53e-09	7.47e-09	1.21e-08	6.05e-09	6.96e-09	6.47e-09	6.49e-09	6.13e-09	8.97e-09	6.63e-09	6.83e-09	5.71e-09	6.78e-09
Liver1	7.25e-09	7.48e-09	6.11e-09	5.67e-09	6.25e-09	5.06e-09	9.75e-09	7.42e-09	6.80e-09	6.37e-09	5.96e-09	5.38e-09	6.29e-09	4.91e-09
Liver2	6.10e-09	6.94e-09	9.98e-09	6.32e-09	6.47e-09	6.94e-09	8.04e-09	8.84e-09	1.02e-08	5.94e-09	6.80e-09	4.93e-09	5.99e-09	6.73e-09
Liver3	6.52e-09	5.79e-09	6.09e-09	8.06e-09	7.71e-09	9.09e-09	6.70e-09	5.65e-09	7.10e-09	7.15e-09	9.55e-09	7.11e-09	7.45e-09	6.76e-09

Table 20
The statistical analysis of the experimental results compared with 14 state-of-the-art methods on 10 application-oriented data sets on AUC-area.

Datasets	p-value1	p-value2	p-value3	p-value4	p-value5	p-value6	p-value7	p-value8	p-value9	p-value10	p-value11	p-value12	p-value13	p-value14
CM1	5.93e-09	4.21e-09	4.52e-09	4.62e-08	6.84e-09	6.17e-09	6.03e-09	1.08e-08	9.19e-09	6.14e-09	7.70e-09	4.85e-09	1.84e-08	9.73e-09
JM1	1.42e-08	8.21e-10	3.38e-10	2.94e-10	2.71e-08	6.87e-09	8.88e-09	6.16e-09	9.18e-09	5.59e-09	8.49e-09	9.35e-09	5.82e-06	5.51e-09
MC1	8.70e-09	2.90e-09	1.26e-09	1.84e-08	3.77e-09	1.29e-08	4.43e-09	3.50e-09	8.32e-09	6.26e-09	9.23e-09	8.26e-09	1.02e-08	7.10e-09
MC2	1.60e-09	3.07e-09	5.31e-09	7.88e-09	2.19e-09	3.23e-09	1.83e-08	8.45e-01	2.76e-08	2.61e-09	4.10e-09	3.80e-09	3.31e-08	1.68e-08
PC1	3.31e-09	5.05e-09	3.03e-09	1.25e-11	4.90e-09	6.02e-09	9.79e-09	7.75e-09	6.24e-09	1.17e-08	5.76e-09	1.12e-08	6.95e-09	7.03e-09
KC2	8.05e-09	8.15e-08	7.10e-09	4.79e-09	1.60e-07	7.62e-09	5.08e-09	8.64e-09	9.67e-09	9.21e-09	1.19e-08	4.60e-09	5.20e-09	1.25e-08
KC3	1.08e-08	9.00e-09	7.21e-09	4.06e-09	6.54e-09	4.73e-09	5.96e-09	9.66e-09	8.55e-09	8.22e-09	4.54e-09	3.61e-09	5.45e-09	4.07e-09
Liver1	8.27e-09	1.16e-08	4.67e-09	8.83e-09	7.54e-09	4.49e-09	6.23e-09	6.34e-09	6.23e-09	4.59e-09	6.81e-09	4.41e-09	8.57e-09	6.67e-09
Liver2	6.47e-09	8.32e-09	1.24e-08	6.41e-09	6.58e-09	5.16e-09	6.25e-09	7.07e-09	6.75e-09	6.06e-09	6.14e-09	6.84e-09	9.81e-09	8.69e-09
Liver3	6.76e-09	2.39e-09	3.68e-08	3.97e-09	5.22e-09	1.17e-08	4.90e-09	4.12e-09	9.93e-09	5.67e-09	8.40e-09	5.93e-09	7.46e-09	7.20e-09

Table 21
The statistical analysis of the experimental results compared with 14 state-of-the-art methods on 10 application-oriented data sets on MMD-score.

Datasets	p-value1	p-value2	p-value3	p-value4	p-value5	p-value6	p-value7	p-value8	p-value9	p-value10	p-value11	p-value12	p-value13	p-value14
CM1	3.59e-09	3.70e-09	4.07e-09	3.55e-09	4.02e-09	3.48e-09	3.67e-09	1.98e-09	3.98e-09	4.06e-09	4.66e-09	3.72e-09	3.76e-09	3.85e-09
JM1	3.62e-09	3.73e-09	3.86e-09	3.49e-09	3.73e-09	3.69e-09	2.66e-09	3.30e-09	3.50e-09	3.73e-09	4.18e-09	3.54e-09	3.10e-09	3.81e-09
MC1	4.21e-09	3.85e-09	4.05e-09	4.13e-09	4.42e-09	4.12e-09	3.32e-09	4.21e-09	3.69e-09	3.66e-09	4.21e-09	4.03e-09	4.04e-09	3.81e-09
MC2	3.23e-09	3.93e-09	2.47e-09	2.89e-09	3.31e-09	3.33e-09	2.69e-09	2.63e-09	3.42e-09	3.36e-09	3.80e-09	3.33e-09	4.23e-09	2.94e-09
PC1	4.01e-09	3.98e-09	4.61e-09	4.38e-09	4.32e-09	3.79e-09	4.15e-09	3.88e-09	3.84e-09	3.88e-09	4.30e-09	4.18e-09	4.28e-09	3.85e-09
KC2	4.54e-09	4.22e-09	3.99e-09	4.05e-09	4.26e-09	3.82e-09	3.83e-09	3.34e-09	3.94e-09	4.44e-09	3.78e-09	4.11e-09	3.61e-09	4.13e-09
KC3	3.62e-09	3.83e-09	3.51e-09	3.52e-09	3.51e-09	3.96e-09	3.88e-09	3.42e-09	4.12e-09	3.63e-09	4.10e-09	4.27e-09	3.89e-09	3.60e-09
Liver1	6.91e-09	6.66e-09	6.84e-09	6.88e-09	6.85e-09	6.93e-09	6.88e-09	6.79e-09	6.96e-09	6.92e-09	7.19e-09	6.71e-09	6.90e-09	7.08e-09
Liver2	6.79e-09	6.93e-09	6.85e-09	6.73e-09	6.72e-09	6.83e-09	6.99e-09	7.03e-09	7.23e-09	6.89e-09	7.00e-09	7.04e-09	6.90e-09	6.69e-09
Liver3	6.88e-09	6.82e-09	7.09e-09	6.84e-09	6.90e-09	6.82e-09	6.91e-09	7.13e-09	6.82e-09	6.71e-09	6.87e-09	6.94e-09	6.89e-09	6.95e-09

Table 22
The statistical analysis of the experimental results compared with 14 state-of-the-art methods on 10 application-oriented data sets on Silhouette-score.

Datasets	p-value1	p-value2	p-value3	p-value4	p-value5	p-value6	p-value7	p-value8	p-value9	p-value10	p-value11	p-value12	p-value13	p-value14
CM1	6.88e-09	6.08e-09	5.37e-09	5.79e-09	1.07e-08	8.94e-09	5.55e-09	1.36e-08	6.94e-09	6.74e-09	8.29e-09	6.69e-09	7.51e-09	1.04e-08
JM1	5.83e-09	6.26e-09	8.10e-09	7.83e-09	5.54e-07	6.23e-09	6.00e-09	7.48e-09	7.41e-09	5.98e-08	7.62e-09	9.12e-09	8.65e-09	9.73e-09
MC1	1.18e-08	6.69e-09	5.55e-09	1.36e-08	1.37e-08	3.41e-09	7.07e-09	1.12e-08	9.94e-09	6.68e-09	4.79e-09	6.27e-09	7.70e-09	1.24e-09
MC2	6.10e-09	6.88e-09	5.52e-09	4.78e-09	4.40e-09	7.23e-09	8.94e-09	1.01e-08	5.17e-09	5.80e-09	9.15e-09	5.72e-09	2.47e-10	8.07e-09
PC1	7.35e-09	6.72e-09	7.29e-09	7.82e-09	1.64e-08	7.18e-09	5.25e-09	7.46e-09	8.16e-09	9.86e-09	6.72e-09	6.94e-09	6.47e-09	1.25e-07
KC2	4.95e-09	6.82e-09	6.85e-09	6.91e-09	1.89e-08	7.39e-09	5.69e-09	7.01e-09	7.19e-09	3.54e-09	8.65e-09	5.73e-09	6.64e-09	7.01e-09
KC3	6.35e-09	8.41e-09	6.92e-09	6.33e-09	5.83e-09	6.47e-09	6.39e-09	7.16e-09	6.77e-09	8.66e-09	8.17e-09	6.52e-09	8.40e-09	7.44e-09
Liver1	7.64e-09	6.76e-09	5.61e-09	7.68e-09	6.01e-09	6.96e-09	6.14e-09	6.78e-09	6.86e-09	7.18e-09	5.87e-09	5.99e-09	5.96e-09	6.63e-09
Liver2	6.77e-09	6.30e-09	7.87e-09	6.55e-09	6.89e-09	6.55e-09	6.78e-09	6.28e-09	6.18e-09	7.85e-09	7.49e-09	6.65e-09	6.74e-09	8.99e-09
Liver3	5.97e-09	7.00e-09	5.86e-09	5.90e-09	6.39e-09	7.39e-09	6.81e-09	6.64e-09	7.13e-09	6.11e-09	6.33e-09	5.71e-09	5.45e-09	5.85e-09

the classification boundaries. This result is confirmed by the classification results. For example, many examples generated by CCR occur in the overlap region, and the F-measure and G-mean of this method are the worst among all methods.

In summary, the proposed methods BIDC1 and BIDC2 outperform the 14 state-of-the-art methods regarding the F-measure, G-mean, AUC-area, MMD-score, and Silhouette-score in most cases. We believe the likely reasons are that (1) instead of generating positive samples around a ground truth sample and its nearest neighbors, the proposed methods learn the distribution of positive samples. (2)the synthetic positive samples generated by the proposed methods have good diversity and separability, which is confirmed by the higher MMD and Silhouette scores of the proposed methods than the 14 state-of-the-art approaches. To verify this result, we visualized the experimental results (i.e., the generated positive data points) on the artificial data set (Fig. 5). Fig. 5 shows the sample distribution after oversampling on the artificial data set for different methods. The red "-" represents the negative sample, the blue "+" represents the positive sample, and the green "+" represents the generated positive sample. The samples generated by the traditional oversampling methods (e.g., SMOTE and ADASYN) have a large overlap with the negative samples, and the generated samples have low diversity. As a result, the traditional methods result in a smaller improvement in the prediction accuracy of the positive class and low accuracy of the negative class. Although the samples generated by AC-GAN are relatively scattered, they overlap significantly with the negative samples, resulting in low classification performance. MFC-GAN has less overlap between the positive and negative classes, but the diversity is low. The two proposed methods are superior to the other methods in terms of the diversity of the generated positive samples and the overlap between the positive and negative classes. The visualization results on the artificial data set are consistent with the results on the MMD-score and Silhouette-score, confirming that the two proposed methods substantially improve the prediction accuracy of the positive and negative class. Furthermore, the experimental results in Tables 7 to 11 indicate that the BIDC1 and BIDC2 are not only suitable for data sets with low imbalance ratios, e.g., the data sets ILPD1 and ILPD2, but also for data sets with high imbalance ratios, such as data sets Yeast4 and Yeast6.

4.2. Comparison of the 16 algorithms on the 10 application-oriented data sets

To demonstrate the applicability of the two algorithms BIDC1 and BIDC2, we also experimentally compared the two algorithms with 14 state-of-the-art algorithms on the 10 application-oriented data sets. The experimental environment and parameter settings are the same as in the previous experiments, except for the number of hidden layer neurons and iterations, which are listed in Table 12, the experimental results on the F-measure, G-mean, AUC-area, MMD-score, and Silhouette-score are listed in Tables 13–17.

It is observed from the experimental results listed in Tables 13–17 that the proposed methods BIDC1 and BIDC2 outperform more than half of the state-of-the-art methods on all metrics. Similarly, as on the artificial and 15 public testing data sets, our methods outperform all state-of-the-art methods on metric of MMD-score and BIDC2 performs the best as it achieves most maximum values. K-SMOTE outperforms our methods on few small data sets. However, the margins are not large and our performance is close to that of K-SMOTE. The reason could be that when there are only limited samples, it is harder for generative models to learn the correct distributions. In summary, the BIDC1 and BIDC2 outperform the 14 state-of-the-art methods in terms of the F-measure, G-mean, AUC-area, MMD-score, Silhouette-score in most case. The BIDC2 significantly outperforms the 14 state-of-the-art methods in terms of the MMD-score on 7 software defect prediction data sets, and the BIDC1 significantly outperforms the 14 state-of-the-art methods in terms of the MMD-score on 3 liver data sets, which have high imbalance ratios. In addition, the experimental results on the MMD-score imply that the samples generated by the BIDC1 and BIDC2 have high diversity, considerably extending the training domain of the positive samples.

To further confirm the superiority of the BIDC1 and BIDC2, we statistically analyzed the experimental results on the F-measure, G-mean, AUC-area, MMD-score and Silhouette-score using paired T-test with a confidence level of 0.05 [49,50]. For each data set and each method, we run the 5-fold cross-validation 5 times and obtained fourteen 25-dimensional statistics denoted by $\boldsymbol{S}_i (1 \leqslant i \leqslant 15)$, which correspond to ROS, SMOTE, B-SMOTE, ADASYN, K-SMOTE, ANS, CCR, NRPSOS, C-SMOTE, SOMO, G-SMOTE, OUPS, AC-GAN, MFC-GAN, and the BIDC1 or BIDC2, respectively. Next the paired T-test is applied to the experimental results by calling the Python library function ttest_rel$(\boldsymbol{S}_i, \boldsymbol{S}_{15})$ $(1 \leqslant i \leqslant 14)$. Due to page limitation, we only provide the results of the statistical analysis on the experimental results compared the BIDC2 with the 14 related methods on the 10 application-oriented data sets. The results of the statistical analysis on the F-measure, G-mean, AUC-area, MMD-score and Silhouette-score are listed in Tables 18-22. The p-values listed in the five tables show that the BIDC2 statistically outperforms ROS, SMOTE, B-SMOTE, ADASYN, K-SMOTE, ANS, CCR, NRPSOS, C-SMOTE, SOMO, G-SMOTE, OUPS, AC-GAN, MFC-GAN.

5. Conclusions

We focus on addressing binary classification problem in this paper. Previous SMOTE-based approaches generate synthetic samples on the lines between the positive sample and its nearest neighbors, and the generated positive samples lack diversity. Previous generative model based approaches suffers model collapse probem and cannot generate diverse positive samples. To alleviate the problems, two binary imbalanced data classification algorithms BIDC1 and BIDC2 based on diversity oversampling by ELMAE and GAN were proposed. The algorithms have three advantages: (1) The ideas of the two algorithms

are simple but they are highly effective. They provide excellent performance for data sets with low and high imbalance ratios. (3) They are suitable for different practical scenarios. The experimental results suggest that previous methods generate positive samples with limited diversity and cannot avoid the overlapping between positive and negative classes. In contrast, the two proposed methods are superior to other methods, outperforming the 14 related state-of-the-art methods in terms of F-measure, G-mean, AUC-area, MMD-score, Silhouette-score in most cases. Future studies will focus on (1) the scalability of the BIDC1 and BIDC2 in big data scenarios, and (2) theoretical analysis of the discrepancy between the original and generated data distributions.

CRediT authorship contribution statement

Junhai Zhai: Conceptualization, Methodology, Resources, Investigation, Visualization, Funding acquisition, Project administration, Supervision, Writing - review & editing. **Jiaxing Qi:** Validation, Investigation, Data curation, Visualization. **Chu Shen:** Validation, Investigation, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research is supported by the National natural science foundation of China (71371063), by the Key R&D program of science and technology foundation of Hebei Province, China (19210310D), and by the Natural science foundation of Hebei Province, China (F2021201020).

References

[1] N.V. Chawla, K.W. Bowyer, L.O. Hall, et al, SMOTE: Synthetic minority oversampling technique, Journal Artificial Intelligence Research 16 (2002) 321–357.
[2] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new oversampling method in imbalanced data sets learning, in: International Conference on Advances in Intelligent Computing, Springer-Verlag, 2005, pp. 878–887.
[3] H.B. He, Y. Bai, E.A. Garcia, et al, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, IEEE International Joint Conference on Neural Networks, IJCNN (2008) 1322–1328.
[4] G. Douzas, F. Bacao, Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE, Information Sciences 501 (2019) 118–135.
[5] K. Shin, J. Han, S. Kang, MI-MOTE: Multiple imputation-based minority oversampling technique for imbalanced and incomplete data classification, Information Sciences 575 (2021) 80–89.
[6] B. Chen, S. Xia, Z. Chen, et al, RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise, Information Sciences 553 (2021) 397–428.
[7] A. Gretton, K.M. Borgwardt, M. Rasch, et al, A Kernel Method for the Two-Sample Problem, in: Advances in Neural Information Processing Systems 19 (NIPS, 2006, pp. 1672–1679.
[8] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[9] H.B. He, E.A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.
[10] D. Elreedy, A.F. Atiya, A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance, Information Sciences 505 (2019) 32–64.
[11] A. Fernández, S. Garcá, F. Herrera, et al, SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary, Journal of Artificial Intelligence Research 61 (2018) 863–905.
[12] K.G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, Information Sciences 465 (2018) 1–20.
[13] W. Siriseriwan, K. Sinapiromsaran. Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling. Songklanakarin Journal of Science and Technology, 39(5), 565–576..
[14] M. Koziarski, M. Wozniak, CCR: A combined cleaning and resampling algorithm for imbalanced data classification, International Journal of Applied Mathematics and Computer Science 27 (2017) 727–736.
[15] W.A. Rivera, Noise Reduction A Priori Synthetic Over-Sampling for class imbalanced data sets, Information Sciences 408 (2017) 146–161.
[16] L. Ma, S.H. Fan, CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests, BMC Bioinformatics 18 (1) (2017) 169–179.
[17] G. Douzas, F. Bacao, Self-Organizing Map Oversampling (SOMO) for imbalanced data set learning, Expert Systems with Applications 82 (2017) 40–52.
[18] H. Lee, J. Kim, S. Kim, Gaussian-Based SMOTE Algorithm for Solving Skewed Class Distributions, International Journal of Fuzzy Logic and Intelligent Systems 17 (2017) 229–234.
[19] W.A. Rivera, P. Xanthopoulos, A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets, Expert Systems with Applications 66 (2016) 124–135.
[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al, Generative adversarial nets, Advances in Neural Information Processing Systems 1 (2014) 2672–2680.
[21] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, Proceedings of the International Conference on Machine Learning 70 (2017) 2642–2651.
[22] M. Zhai, L. Chen, G. Mori, Hyper-LifelongGAN: Scalable Lifelong Learning for Image Conditioned Generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR2021), June 2021, pp. 2246–2255.
[23] M. Zhai, L. Chen, F. Tung, et al, Lifelong GAN: Continual Learning for Conditional Image Generation, 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 2759–2768, https://doi.org/10.1109/ICCV.2019.00285.
[24] M. Zhai, L. Chen, J. He, et al. Piggyback GAN: Efficient Lifelong Learning for Image Conditioned Generation. In: A. Vedaldi, H. Bischof, T. Brox, J. Frahm. (eds) Computer Vision-ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, vol 12366. Springer, Cham. doi: 10.1007/978-3-030-58589-1_24..

[25] A. Ali-Gombe, E. Elyan, MFC-GAN: Class-imbalanced dataset classification using Multiple Fake Class Generative Adversarial Network, Neurocomputing 361 (2019) 212–221.
[26] M. Zhang, T. Li, R. Zhu, et al, Conditional Wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification, Information Sciences 512 (2020) 1009–1023.
[27] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, 2009 IEEE Symposium on Computational Intelligence and Data Mining (2009) 324–331, https://doi.org/10.1109/CIDM.2009.4938667.
[28] S.H. Khan, M. Hayat, M. Bennamoun, et al, Cost-Sensitive Learning of Deep Feature Representations from Imbalanced Data, IEEE Transactions on Neural Networks and Learning Systems 29 (8) (2018) 3573–3587.
[29] C. Zhang, K.C. Tan, H.Z. Li, et al, A Cost-Sensitive Deep Belief Network for Imbalanced Classification, IEEE Transactions on Neural Networks and Learning Systems 30 (1) (2019) 109–122.
[30] Q. Dong, S. Gong, X. Zhu, Imbalanced Deep Learning by Minority Class Incremental Rectification, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (6) (2019) 1367–1381.
[31] B. Mateusz, M. Atsuto, M.A. Mazurowski, A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks, Neural Networks 106 (2018) 249–259.
[32] N.V. Chawla, A. Lazarevic, L.O. Hall, et al, SMOTEBoost: Improving prediction of the minority class in boosting, In Proceeding of Knowledge Discovery in Databases (2003) 107–119.
[33] Z. Chen, T. Lin, X. Xia, et al, A synthetic neighborhood generation based ensemble learning for the imbalanced data classification, Applied Intelligence 48 (2018) 2441–2457.
[34] B.S. Raghuwanshi, S. Shukla, Class imbalance learning using UnderBagging based kernelized extreme learning machine, Neurocomputing 329 (2019) 172–187.
[35] B. Tang, H.B. He, GIR-based ensemble sampling approaches for imbalanced learning, Pattern Recognition 71 (2017) 306–319.
[36] Z. Chen, J. Duan, L. Kang, et al, A hybrid data-level ensemble to enable learning from highly imbalanced dataset, Information Sciences 554 (2021) 157–176.
[37] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.
[38] Y. Sun, M.S. Kamel, A.K. Wong, et al, Cost-sensitive Boosting for Classification of Imbalanced Data, Pattern Recognition 40 (12) (2007) 3358–3378.
[39] X. Tao, Q. Li, W. Guo, et al, Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification, Information Sciences 487 (2019) 31–56.
[40] P.A. Alaba, S.I. Popoola, L. Olatomiwa, et al, Towards a more efficient and cost-sensitive extreme learning machine: A state-of-the-art review of recent trend, Neurocomputing 350 (2019) 70–90.
[41] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.
[42] W. Cao, M.J.A. Patwary, P. Yang, et al. An Initial Study on the Relationship Between Meta Features of Dataset and the Initialization of NNRW. 2019 International Joint Conference on Neural Networks (IJCNN), Jul. 14, 2019, pp. 1–8..
[43] X.L. Zhou, X.Z. Wang, C. Hu, et al, An analysis on the relationship between uncertainty and misclassification rate of classifiers, Information Sciences 535 (2020) 16–27.
[44] M.J.A. Patwary, X.Z. Wang, D. Yan, Impact of fuzziness measures on the performance of semi-supervised learning, International Journal of Fuzzy Systems 21 (5) (2019) 1430–1442.
[45] B.S. Raghuwanshi, S. Shukla, Classifying imbalanced data using ensemble of reduced kernelized weighted extreme learning machine, International Journal of Machine Learning and Cybernetics 10 (11) (2019) 3071–3097.
[46] J. Liu, M.J.A. Patwary, X. Sun, et al, An experimental study on symbolic extreme learning machine, International Journal of Machine Learning and Cybernetics 10 (4) (2019) 787–797.
[47] Y.F. Chu, C.Y. Feng, C.L. Guo, et al, Network embedding based on deep extreme learning machine, International Journal of Machine Learning and Cybernetics 10 (10) (2019) 2709–2724.
[48] P.K. Wong, X.H. Gao, K.I. Wong, et al, Initial-training-free online sequential extreme learning machine based adaptive engine air-fuel ratio control, International Journal of Machine Learning and Cybernetics 10 (9) (2019) 2245–2256.
[49] D. Janez, Statistical comparisons of classifiers over multiple datasets, Journal of Machine Learning Research 7 (1) (2006) 1–30.
[50] P.K. Singh, R. Sarkar, M.M. Nasipuri, Significance of non-parametric statistical tests for comparison of classifiers over multiple datasets, International Journal of Computing Science and Mathematics 7 (5) (2016) 410–442.