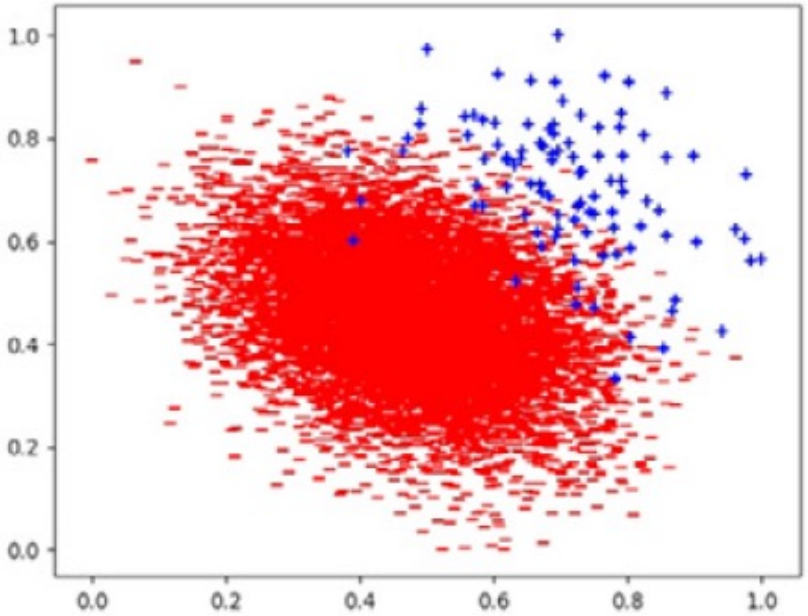
Binary imbalanced data classification based on diversity oversampling by generative models

Junhai Zhai, Jiaxing Qi, Chu Shen



Published in 2022 in Information Sciences

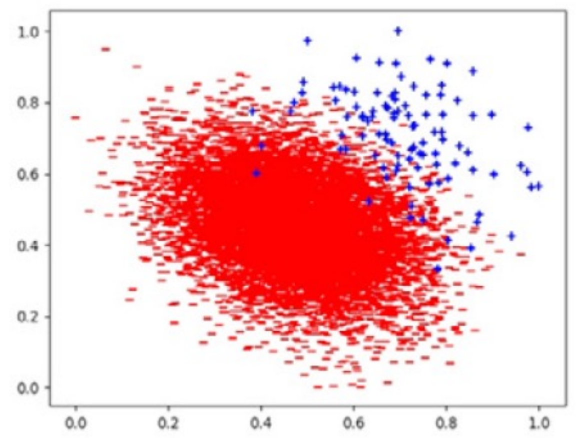
Presenter: Abdullah Mamun

Date: January 9, 2024

Summary

- Addresses the data imbalance problem in binary classification
- Overview of different data balancing tools: SMOTE, RSMOTE, AdaSYN, etc.
- Proposes two new **binary data imbalance classification (BIDC)** algorithms.

1. BIDC1 (uses extreme learning machine autoencoder)
2. BIDC2 (uses GAN)



<u>I will present BIDC2 first as I understood that one better.</u>

GAN

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 Gaussian distribution.

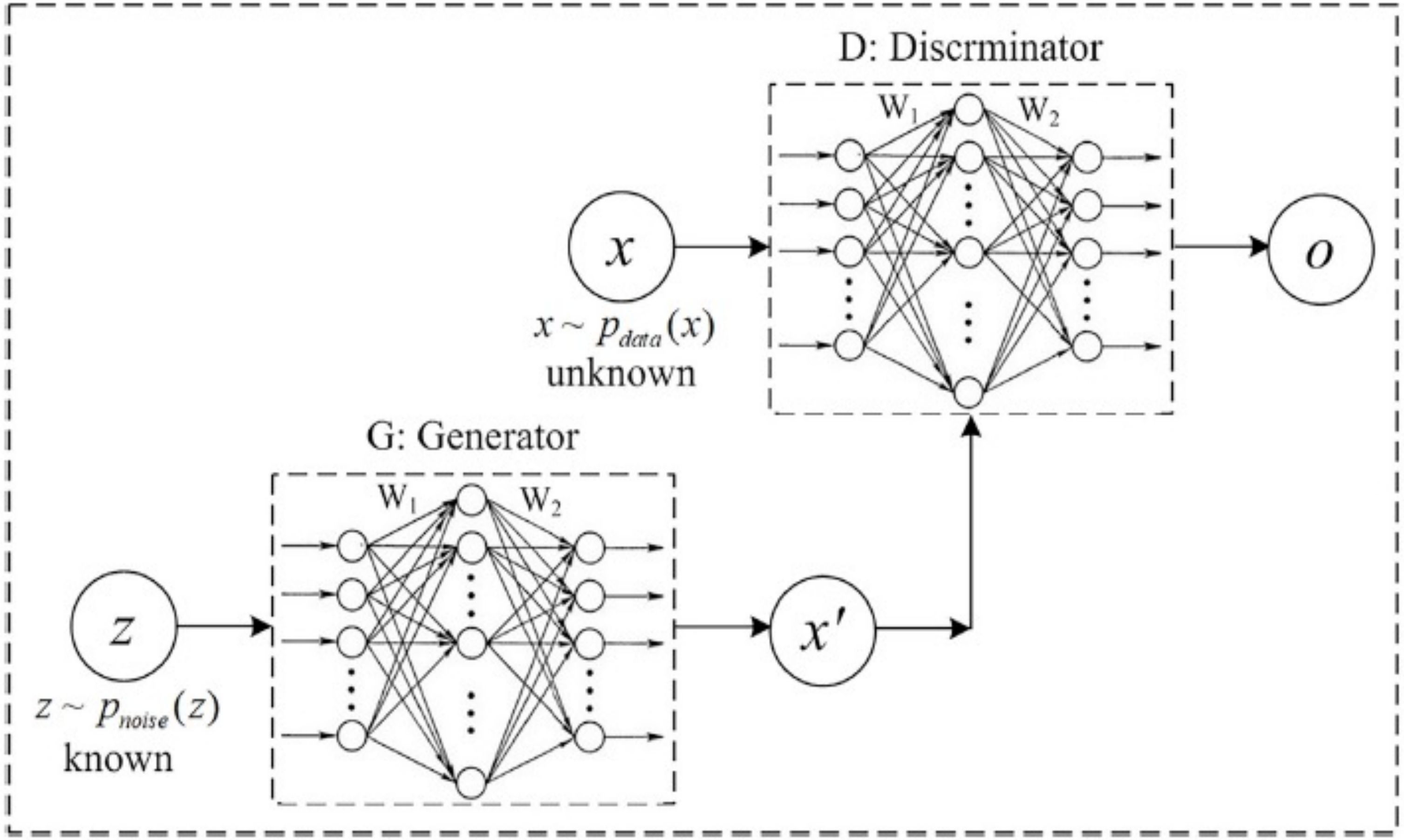


Fig. 3. The architecture of generative adversarial network.

GAN training

- In every step:
- Train the discriminator k times
- Train the generator once

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:

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

  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:

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

end

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

GAN training

- In every step:
- Train the discriminator k times
- Train the generator once

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

Input: The training set $S_{tr} = \{\mathbf{x_i}, 1 \leq i \leq n\}$, 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 $(i = 1; i \leq t; i = i + 1)$ **do**

  for $(j = 1; j \leq k; j = j + 1)$ **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:

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

  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:

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

end

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

Probability of a real sample detected real.

Generated sample

Probability of a generated sample detected real.

Negative of loss. So, we want to maximize it. Hence the gradient ascend.

GAN training

- In every step:
- Train the discriminator k times
- Train the generator once

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

Input: The training set $S_{tr} = \{\mathbf{x_i}, 1 \leq i \leq n\}$, 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 $(\theta^{(D)}, \theta^{(G)})$.

for $(i = 1; i \leq t; i = i + 1)$ **do**
  for $(j = 1; j \leq k; j = j + 1)$ **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:

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

  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:

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

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

Probability of a real sample detected real.

Generated sample

Probability of a generated sample detected real.

Negative of loss. So, we want to maximize it. Hence the gradient ascend.

Example 1: <u>Perfect discriminator</u>
D(real) = 1
D(fake) = 0
So, negative loss = log 1 + log (1-0) = 0 + 0 = 0

Another example: <u>(Classify all as real)</u>
D(real) = 1
D(fake) = 1
So, negative loss = log 1 + log (1-1) = 0 + (-inf) = -inf (i.e. loss = INF)

GAN training

- In every step:
- Train the discriminator k times
- Train the generator once

Negative of loss. So, we want to maximize it. Hence the gradient ascend.

Example 1: <u>Perfect discriminator</u>

D(real) = 1

D(fake) = 0

So, negative loss = log 1 + log (1-0) = 0 + 0 = 0

Another example: <u>(Classify all as real)</u>

D(real) = 1

D(fake) = 1

So, negative loss = log 1 + log (1-1) = 0 + (-inf) = -inf (i.e. loss = INF)

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

Input: The training set $S_{tr} = \{\mathbf{x}_i, 1 \leq i \leq n\}$, 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 $(\theta^{(D)}, \theta^{(G)})$.

for $(i = 1; i \leq t; i = i + 1)$ **do**

  for $(j = 1; j \leq k; j = j + 1)$ **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:

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

  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:

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

end

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

Probability of a real sample detected real.

Generated sample

Probability of a generated sample detected real.

This will be the negative of reward for the generator, because it wants to fool the discriminator. Generated sample detected real is good for the generator.

Reward = 0 if the fake is detected as fake with 100% confidence. And INF if fake is detected as real with 100% confidence.

BIDC2 algorithm

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;`

$S_{tr}^{+} = S_{t+1}^{+}$;

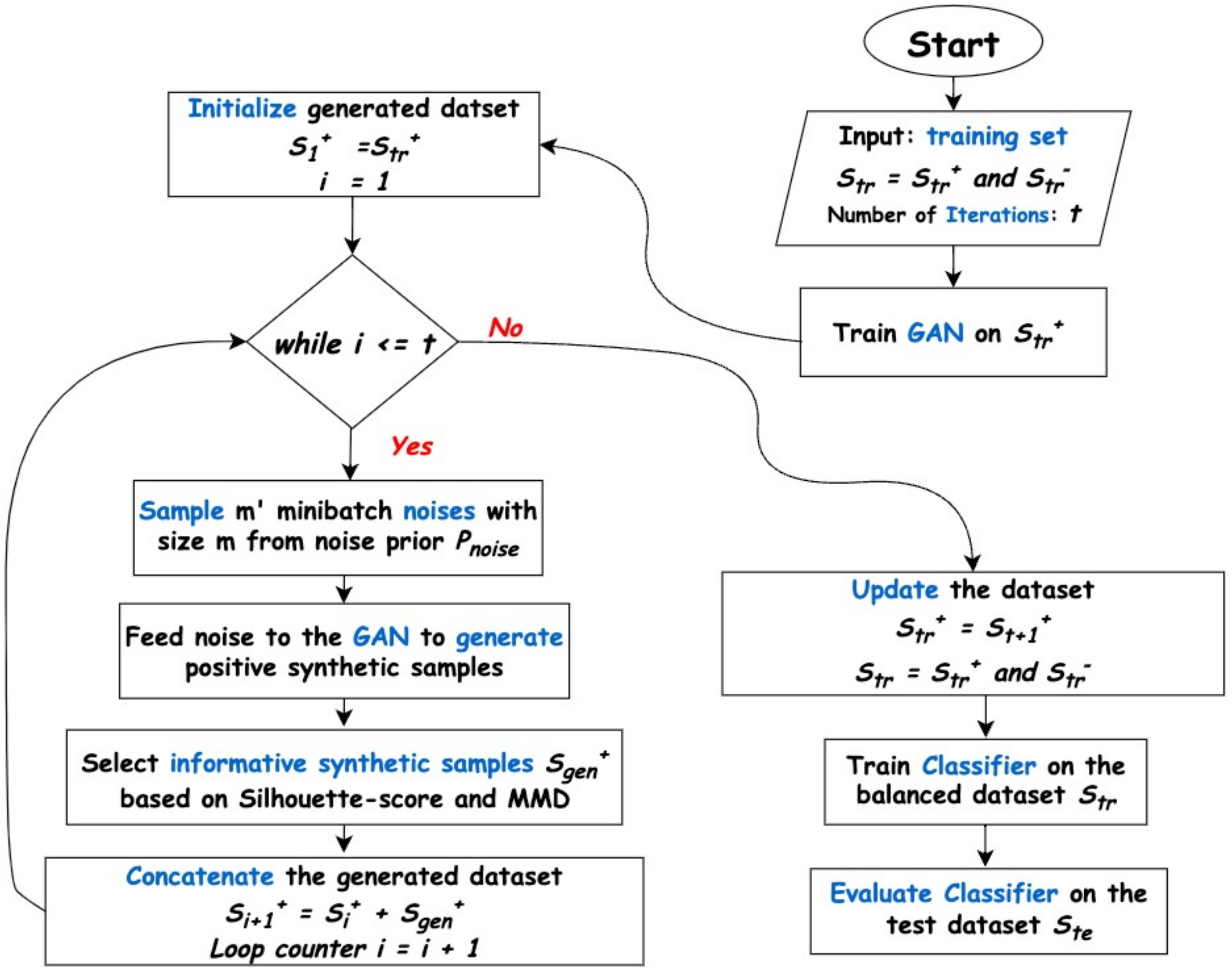
$S_{tr} = S_{tr}^{+} + S_{tr}^{-}$;

Train a classifier on S_{tr}, and use the trained classifier to classify $\mathbf{x} \in S_{te}$;

BIDC2 algorithm

Three stages:

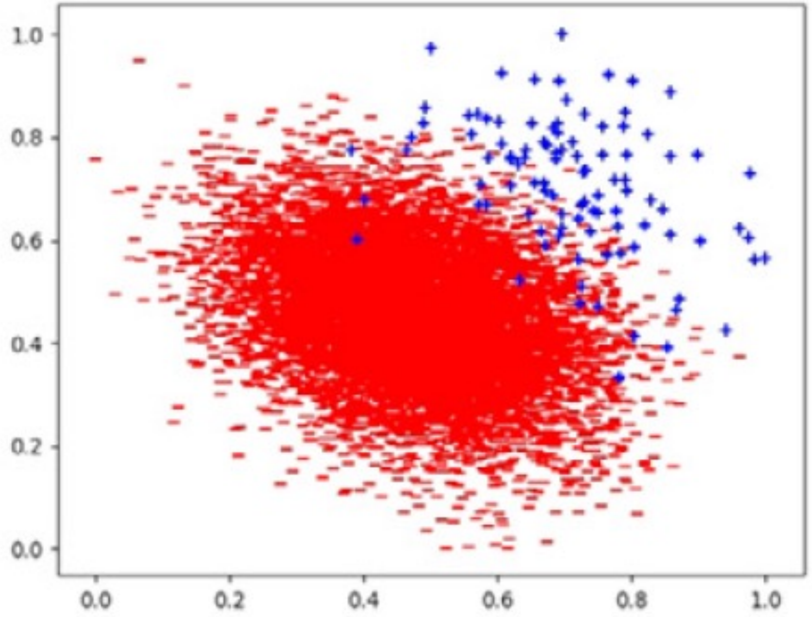
1. **Training the GAN** on the positive training examples S_{tr}^{+}
2. **Generating** synthetic positive samples with the trained GAN model
3. **Train and evaluate** the classifier



Silhouette score

- a = Dissimilarity of a sample within its cluster (we want it to be small)

- b = Dissimilarity of a sample with every other clusters (we want it to be large)



Silhouette score of a cluster is the average of the Silhouette scores of all the samples of that cluster.

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

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

So, a higher silhouette score is better.

where $a(\mathbf{x})$ is the average dissimilarity of sample **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 **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 $s(\mathbf{x})$

MMD (maximum mean discrepancy)

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).

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

Kernel functions in SVM: https://www.geeksforgeeks.org/major-kernel-functions-in-support-vector-machine-svm/

Extreme Learning Machine Autoencoder (ELMAE)

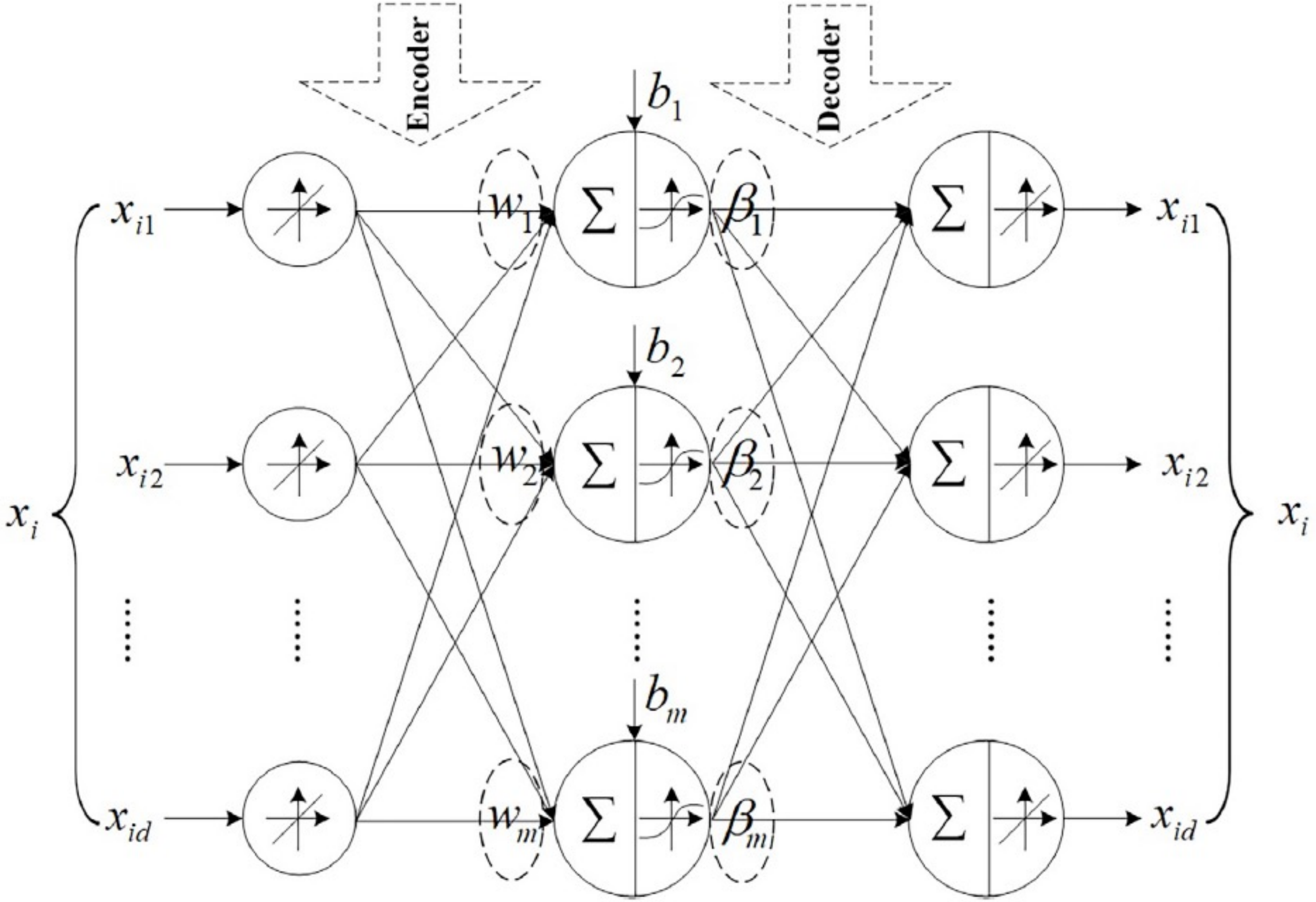


Fig. 2. The extreme learning machine autoencoder.

Extreme Learning Machine (ELM)

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 **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

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

Extreme Learning Machine (ELM)

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).

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

where ξ_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

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

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

BIDC1 algorithm

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 $\mathbf{x} \in S_{te}$.

`// Stage 1: training the ELMAE on` S_{tr};

for $(j = 1; j \leq m; j = j + 1)$ **do**

  Randomly assign input weights $\mathbf{w}_j$ and b_j;

end

Calculate the hidden layer output matrix $\mathbf{H}$;

Calculate output weights matrix $\hat{\beta} = (\frac{1}{C}\mathbf{I} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}\mathbf{X}^T$;

BIDC1 algorithm (cont.)

// Stage 2: generating synthetic positive samples with the trained ELMAE model;
$S_1^+ = S_{tr}^+$;
for $(i = 1; i \leq t; i = i + 1)$ **do**
  Input S_i^+ into ELMAE, and compressed vectors can be obtained by the encoder;
  Take these vectors added Gaussian noise with normal distribution as input of decoder, then get the 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 $\mathbf{x} \in S_{te}$;

Datasets

1 artificial dataset and 15 public datasets.

Table 6
The dimension of noise variable **z** and the number of hidden nodes of generator G and discriminator D.

Data sets	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

Datasets

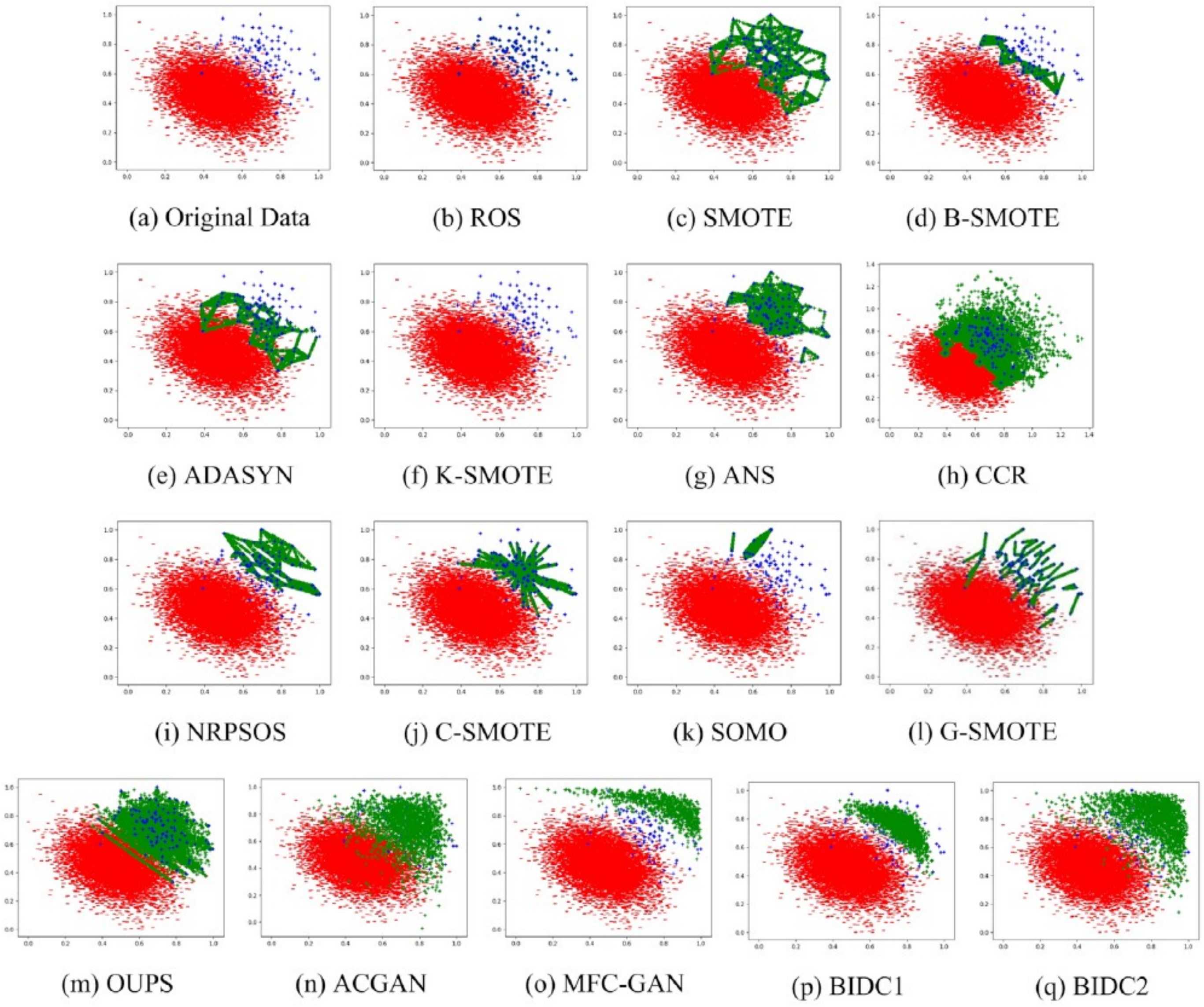
1 artificial dataset and 15 public datasets.

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

Experimental results – visualize the generated data

Comparing BIDC1 and BIDC2 on test dataset against 14 state of the art methods



Experimental results

Comparing BIDC1 and BIDC2 on test dataset against 14 state of the art methods
F measure is reported here.
Test set is not balanced.

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**

Experimental results

Geometric mean of precision and recall is reported here. (The test set is not balanced)

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**

Experimental results

AUC is reported here. (The test set is not balanced)

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**

Appendix – dissimilarity function:

1

To measure the dissimilarity within a cluster you need to come up with some kind of a metric. For categorical data, one of the possible ways of calculating dissimilarity could be the following:

```
d(i, j) = (p - m) / p
```

where:

- `p` is the number of classes/categories in your data
- `m` is the number of matches you have between samples `i` and `j`

For example, if your data has 3 categorical features and the samples, `i` and `j` are as follows:

```
    Feature1  Feature2  Feature3
i    x          y          z
j    x          w          z
```

So here, we have 3 categorical features, so `p=3` and out of these three, two features have same values for the samples `i` and `j`, so `m=2`. Therefore

```
d(i,j) = (3 - 2) / 3
d(i,j) = 0.33
```

https://stackoverflow.com/questions/52512987/measuring-dissimilarity-within-the-cluster-kmodes