Oversampling algorithm of imbalanced classification using DBSCAN and gaussian distribution

Yinuo Xiao[1]

Harbin Institute of Technology, xiaoyinuo@hit.edu.cn

Abstract. Classification over imbalanced data, which often leads to poor prediction ability for minority class(es), is a challenging tasks in machine learning. Many algorithms, including SMOTE and its derivatives, have been proposed to solve this problem by a means of rebanlancing the data distribution via oversampling. However, the effects of these algorithms are still unsatisfactory. This paper proposes new oversampling algorithm by integrating DBSCAN and gaussian distribution into oversampling. First, minority data points(data samples) are divided into core, borderline and noise points by DBSCAN. Second, the total number of oversampling data points is assigned for different kinds of data points. Third, we synthetic new points using gaussian distribution. The experiment results show that our methods have high prediction ability compared with existing methods.

Keywords: imbalance data · oversampling · DBSCAN · Gaussian distribution.

1 Introduction

Imbalanced data [5] refers to datasets that have an uneven amount of data between different classes. We assume that the majority class is negative class and the minority class is positive class in binary classification.

The actual demand shows the importance and difficulty of learning on imbalanced datasets. The imbalance occurs in many fields, such as medical diagnosis [19, 22] and fault detection [24]. In these cases ill samples and fault samples are minority samples. We tend to be interested in minority samples because minority class contains more value. The disease samples need to be predicted more accurately than the normal samples, and the cost of misjudging fault samples is much higher than misjudging normal samples. The difficulty lies in the fact that existing machine learning models skew the prediction results towards majority class in the process of prediction [17], then the minority class cannot be classified correctly [1]. Therefore, More advanced algorithms need to be proposed to improve the ability to predict minority class.

The methods dealing with imbalance can be roughly divided into two types: algorithm-level [18] and data-level [4]. Algorithm-based approaches include BalanceCascade [21], Adaboost, XGBoost [6], etc. Algorithm-based methods are limited to a single type of classifier [14] and cannot be extended to general machine learning models. Data-level methods perform data preprocessing to reduce the imbalance ratio, Then the data can be processed effectively. Data-level methods are generally divided into undersampling [20] and oversampling [4]. undersampling decreases the number of majority samples, while oversampling enhances minority class by introducing new samples [13]. In our work, oversampling is used to preprocess the data.

SMOTE is a classic oversampling algorithm, SMOTE realizes changing the data distributions by randomly generating new sample on the line of minority points. Numerous modifications of

SMOTE algorithm have been proposed, such as Borderline-SMOTE [10], ADASYN [11], etc. However, SMOTE and its derivatives have their weaknesses, they are susceptible to noise points and are difficult to deal with overlapping data distributions. The generated minority points and majority points are easily covered, which affects the separating hyperplane of the classifier. In addition, $SMOTE$ methods can't handle clusters of data. SMOTE is unable to capture the cluster distributions if the monority sample points are distributed in clusters. Furthermore, the method of syntheticing new points by SMOTE is difficult to maintain the original data distributions [7], After using $SMOTE$, the data distributions of minority sample points may change, which will affect the prediction progress.

SMOTE and its derivatives are also poor in multiple classification problems [14]. The relationship between two classes is relatively simple, but in multiclassification problems, the relationship between classes becomes much more complex [2]. When oversampling new points near the boundaries of multiple classes, creating overlapping data distributions [12] is hard to avoid. The existing methods to deal with multiclassification include transforming the problem into binary classification problems(OVO) [9] or dividing the problem into one to many problems(OVA). But processing two classes separately(OVO) will ignore the rest classes' information. Treating one class as a minority class and the rest as a majority class(OVA) cannot capture the relationship between majority classes effectively.

Our work proposes new oversampling methods, which overcome above methods' drawbacks. we proposed ODG(oversampling using DBSCAN and gaussian distribution) and MC-ODG (multiclassification oversampling using DBSCAN and gaussian) algorithms for binary and multiple classification. The proposed algorithms can solve the above problems, such as noise interference and oversampling. The generated points can keep the original data distributions. The new algorithms apply DBSCAN clustering method and gaussian distribution to fit the data. The experimental results show that our algorithms are superior to the existing algorithms on different metrics.

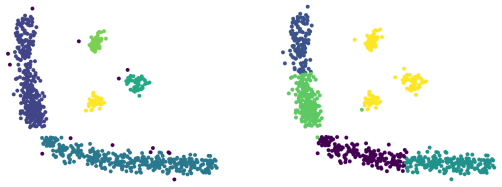
To summarize, our work makes the following contributions.

- ODG is an oversampling method that can effectively process clustered data.
- We explain the influence of noise data points on the oversampling process and ODG shows how to deal with noise data points.
- ODG preserves the original data distributions after oversampling.
- ODG is suitable for binary classification, and its extension MC-ODG is suitable for multiclassification problems.

The paper is organized as follow. The next section introduces related works. Section 3 introduces our ODG and MC-ODG algorithms in details, and gives the specific flow of the algorithms and the corresponding complexity analysis. In section 4, we introduce comparative experiments based on different models, and give the analysis and results of our experiments. Section 5 gives the summary and our future works.

2 Related works

This section provides required knowledge for our algorithms and some related oversampling methods.



(a) $K-means$ (b) $DBSCAN$

Fig. 1. The comparision of $K-means$ and $DBSCAN$

2.1 Clustering methods

Clustering algorithm is to find groups of similar data, and each group of similar data forms a cluster. There may be multiple clusters in a single minority class data, we need to use clustering algorithm to identify multiple clusters and then process each of them. The most famous clustering methods include distance-based K-MEANS algorithm and density-based DBSCAN algorithm. K-MEANS algorithm is relatively simple, but this algorithm must determine the number of clusters K. Meanwhile, compared with density-based DBSCAN algorithm, K-MEANS cannot find clusters with various shapes. DBSCAN defines a cluster as the largest set of points connected by density and can find clusters of arbitrary shape. After the whole process, DBSCAN classifies minority points into core points, borderline points and noise points. Given the parameters *eps* and *min_pts*, we will get three kinds of points. Core points have no less than *min_pts* points within *eps* range. Borderline points refer to the points that were partitioned into a cluster after DBSCAN, but not belong to core points. After the DBSCAN algorithm, The points that were not divided into any cluster are defined as noise points. The core points can best represent the distribution of the corresponding cluster, therefore we can use the distribution of the core points to generate data. DBSCAN has been used for generating new points, such as DBSMOTE [3] and DSMOTE [23], but these methods failed to escape the SMOTE limitation in the method of systneticing new samples and they did not use the distribution of minority class data.

2.2 Related oversampling methods

Oversampling, balances the distribution of data by generating new data samples without losing information. The fundamental idea of SMOTE is to generate new samples by interpolation between minority class samples [8]. Specifically, for a certain sample x_i, one of the K nearest neighbors is randomly selected as $x_i^{'}$. Use the Eq.1 to generate a new sample x_{new}. $\epsilon \in [0,1]$ is a random number.

$$x_{new} = x_i + (x_i - x_i^{'}) \times \epsilon \tag{1}$$

SMOTE algorithm is easily affected by noise. When noise data is present , the new syntheticed samples may overlap with majority class samples. Borderline-SMOTE [10] is based on SMOTE, this algorithm classifies the minority samples into three types, *noise*, *danger* and $safe$. All the K nearest neighbors of the *noise* points are majority points, more than half of the K nearest neighbors of the *danger* points are majority points, and all of K nearest neighbors of $safe$ points are minority points. Borderline-SMOTE uses *danger* points to generate new samples. To a certain extent Borderline-SMOTE reduced the negative effect of noise and overlapping data distribution. ADASYN [11] is an

adaptive composite sampling method, a mechanism is used to determine how many samples need to be synthesized for each minority sample. In general, for a minority sample, more majority points in its K nearest neighbors, the more minority points will be generated near this point. There are also methods conbined oversampling and undersampling, such as SMOTE+ENN [22], This method first deleted the majority points of which all of K nearest neighbors are minority points, then use SMOTE to generate new samples.

CCR [15] and MC-CCR [14] algorithms are oversampling algorithms based on *energy*, In these methods, a spherical region is drawn at the center of each minority sample point. The spherical region is constantly expanding, when meeting a majority point the *energy* is reduced. When the *energy* is exhausted, the region will no longer extend. New data points are randomly generated inside the spherical region. The number of generated data is inversely proportional to the radius of the spherical region. In addition, this method also proposed cleaning process, the majority points inside the spherical area will be shifted out to reduce overlapping. But the CCR approach also has its drawbacks. In the process of generating data points, the CCR method generates data randomly in the region without considering the distribution of data points. In addition, the CCR method may synthetic too many sample points near the noise points. The reason is that the radius of the spherical region corresponding to a noise point is small, many sample points will be generated, which may affect the prediction results. However, with the help of DBSCAN, our algorithms can identify the noise of minority points and avoid generating data around the noise.

3 ODG and MC-ODG Algorithms

This section introduces ODG and MC-ODG algorithms. The methods based on SMOTE are not able to effectively deal with noise points and cluster distribution data. The noise in the data will lead to overlapping data distributions after oversampling. Also, the data point generated by SMOTE cannot preserve the original distributions. MC-CCR randomly generates points in a certain range without retaining the original distributions of a minority points. Furthermore, this method may generate too many samples of minority points near the noise, which will adversely affect the prediction results. Aiming at the shortcomings of the above methods, we propose ODG and MC-ODG to avoid the problems mentioned above.

3.1 Overview of ODG

This section gives an overview of ODG. First, we use DBSCAN to divide the minority points into core points, borderline points and noise points according to data distributions. Second, We calculate the number of sample points generated near different types of points. Third, we systnetic new points. We use gaussian distribution when we systnetic points near borderline points and core points. In addition, we use SMOTE to systetic points if minority points cannot form effective clusters. Some of import parameters we used in this section are shown in table1. The flow of ODG is shown in the Algorithm 1.

3.2 Binary imbalanced classification

We give the details of ODG algorithm, which is used to solve the binary classification problems.

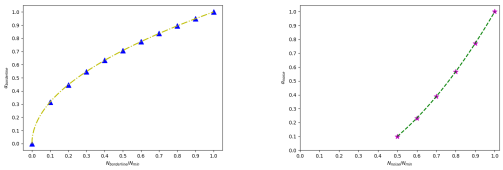
Table 1. Parameters of ODG

Parameter	meaning
eps	parameter of DBSCAN
min_pts	parameter of DBSCAN
M	majority points
m	minority points
$N_{oversampling}$	the number of oversampled points
N_{core}	the number of oversampled points near core points
$N_{borderline}$	the number of oversampled points near borderline points
N_{noise}	the number of oversampled points near noise points
$\alpha_{borderline}$	ratio of generated points near borderline points
α_{noise}	ratio of generated points near noise points
$translations$	the translations of majority points
$clusters$	the clusters of minority points
$n_{borderline}$	the number of generation near a single borderline point
n_{core}	the number of generation near one cluster

Algorithm 1 ODG

Input:
$M \leftarrow$ majority points
$m \leftarrow$ minority points
eps, min_pts
Output: Minority points and Majority points

m_{core}, $m_{borderline}$, m_{noise}, $clusters$=DBSCAN(m,eps,min_pts)
Calculate N_{noise}, $N_{borderline}$ and N_{core}.
$new\_data \leftarrow$ an empty set.
$translations \leftarrow$ a zero matrix, the same shape as M.
$generate\_borderline(N_{borderline}, m_{boaderline}, translations, new\_data)$
$generate\_core(clusters, N_{core}, new\_data)$
$generate\_noise(N_{noise}, m_{noise}$,new_data)
$M \leftarrow M + translations$
$m \leftarrow m$ concatenate with new_data
return M,m



(a) $\alpha_{borderline}$ (b) α_{noise}

Fig. 2.

Clustering We cluster the minority sample points firstly. An result of K-MEANS and DBSCAN are shown in Fig.1. The same color represents the same cluster. Obviously, the result of DBSCAN is more consistent with the data distribution and can easily detect noise points. After DBSCAN, the minority sample points are divided into three types as mentioned. SMOTE cannot avoid synthetic points on the line between two points belong to different clusters. After dividing multiple clusters, we won't generating sample points between clusters to avoid overlapping.

Number of generation After clustering, we determine the value of $N_{borderline}$, N_{core} and N_{noise}. $N_{oversampling}$ is calculated as Eq.2.

$$N_{oversampling} = N_{maj} - N_{min} \tag{2}$$

After obtaining the total number of oversampled samples, we need to determine the proportion of generated in different regions. The parameter $\alpha_{borderline}$ can be specified as a hyperparameter or can be adaptive determined based on the ratio of borderline points. In general, $\alpha_{borderline}$ is greater than $\frac{N_{borderline}}{N_{min}}$, because we prefer to generate data near borderline points, then the classification surface is biased to minority points. Fig.2(a) shows the relation with $\alpha_{borderline}$ and $\frac{N_{borderline}}{N_{min}}$. $\alpha_{borderline}$ is evaluated according to the Eq.3.

$$\alpha_{borderline} = (\frac{N_{borderline}}{N_{min}})^{\frac{1}{2}} \tag{3}$$

In fact, we can calculate $\alpha_{borderline}$ in a variety of ways. We just need to make sure $\alpha_{borderline}$ is greater than $\frac{N_{borderline}}{N_{min}}$, in order to oversample more points near borderline. We could also use the function of 1/3 power, or 1/4 power, etc. We can also specify $\alpha_{borderline}$ by a hyperparameter if we want to generate more points near borderline points.

In general, we separate the noise points by DBSCAN algorithm, We ignore the noise during oversampling to avoid the impact of noise. so no new sample points will be synthesized near noise points. Only when minority points cannot form effective clusters, that is, most of the sample points are marked as noise points, such as when the ratio of noise points *ratio* exceeds a certain super-parameter *noise_ratio*(generally set as 0.5), part of new samples will be synthesized near noise points. As the amount of noise sample point data increases, the ratio of generated data near noise points also increases. The proportion of samples generated near noise point is α_{noise} is shown in equation4.

$$\alpha_{noise} = \begin{cases} \frac{0.9}{1-noise\_ratio^2} \times ratio^2 + 1 - \frac{0.9}{1-noise\_ratio^2} & ratio \geq noise\_ratio \\ 0 & ratio < noise\_ratio \end{cases} \tag{4}$$

Algorithm 2 *generate_borderline*

Input: $N_{borderline}$,$m_{borderline}$,*translations*,*new_data*
for *point* in $m_{borderline}$ **do**
  Calculate K nearest neighbors of *point*.
  $cov\_mat \leftarrow$ covariance matrix of the core points related to *point*.
  $num\_core\_points \leftarrow$ The number of core points in the same cluster as *point*.
  $cov\_mat = cov\_mat/num\_core\_points$
  $d\_max \leftarrow$ maximum distance of knearest neighbors.
  $n_{borderline} = \frac{n\_knearest\_maj}{\sum_{i \in m} n\_knearest\_maj} \times N_{boardline}$
  $tem\_data = multivariate\_normal(point, cov\_mat, n_{borderline})$
  add *tem_data* to *new_data*.
  for m_j in $k\_nearest\_maj$ **do**
    $translations_j$=$translations_j$+$\frac{d_{max}-d_j}{d_j} \times (m_j - point)$

The reason for using equation4 is that when *ratio* is exactly *noise_ratio*, the generated ratio is 0.1. Similar with *borderline_ratio*, We can also have many methods to calculate α_{noise}, we use the quadratic function curve similarly with $\alpha\_borderline$ in our work. When the *noise_ratio* is 0.5, the trend of α_{noise} is shown in Fig.2(b). We calculate N_{noise} first, and then determine the value of $N_{borderline}$ and N_{core}. N_{noise}, $N_{borderline}$ and N_{core} are shown in Eq.5, Eq.6 and Eq.7.

$$N_{noise} = \alpha_{noise} \times N_{oversampling} \tag{5}$$

$$N_{borderline} = (N_{oversampling} - N_{noise}) \times \alpha_{borderline} \tag{6}$$

$$N_{core} = N_{oversampling} - N_{borderline} - N_{noise} \tag{7}$$

Generate data near borderline points We first need to determine $n_{boardline}$ for a particular borderline point. Then we calculate the K nearest neighbors of each borderline point, if the K nearest neighbors of the minority point have more majority sample points, means the point is closer to the decision boundary, and more sample points should be generated near this point. We set the number of majority sample points is $n\_knearest\_maj$, then $n_{boardline}$ can be obtained in Eq.8.

$$n_{borderline} = \frac{n\_knearest\_maj}{\sum_{i \in m} n\_knearest\_maj} \times N_{boardline} \tag{8}$$

In order to make the distribution of the generated new sample points consistent with original data distributions, gaussian distribution is adopted. After clustering, each borderline point corresponds to a certain cluster, and the core points of this cluster can best represent the cluster's distribution. Therefore we calculate the covariance matrix of the core points. But we cannot use the matrix directly, we assume the number of core points in this cluster is num_core_points, we need to scale the matrix by num_core_points to limit the scope of the generated points. We set the particular borderline point is *point*, *cov* is the function to calculate the covariance matrix, The function to generate new sample points using gaussian distribution is *multivariate_normal*, New sample points *new_data* can be obtained by Eq.9.

$$\begin{aligned} cov\_mat &= cov(core\_points)/num\_core\_points \\ new\_data &= multivariate\_normal(point, cov\_mat, n_{boardline}) \end{aligned} \tag{9}$$

The algorithm for generating new sample points near borderline points is algorithm2.

Algorithm 3 *generate_core*

Input: *clusters*,N_{core},*new_data*
for *cluster* in *clusters* **do**
$n_{core} = \frac{num\_data\_cluster}{\sum_{i \in cluster} num\_data\_cluster} \times N_{core}$
core_points ← core points of *cluster*
$cov\_mat = cov(core\_points)$
$tem\_data = multivariate\_normal(mean(core\_points), cov\_mat, n_{core})$
add *tem_data* to *new_data*

Algorithm 4 *generate_noise*

Input: N_{noise},m_{noise},*new_data*
for i from 1 to N_{noise} **do**
point ← random choice from m_{noise}
new_point ← using SMOTE to generate a new sample on point
add *new_point* to *new_data*

Generate data near core points We generate new data from the mean and variance of the core points of each cluster using gaussian distribution. After DBSCAN, the number of generation near the core points of each cluster depends on the number of sample points in the cluster. We assume the number of sample points in the cluster is *num_data_cluster*, the number of generalization in this cluster is n_{core}, as shown in 10.

$$n_{core} = \frac{num\_data\_cluster}{\sum_{i \in cluster} num\_data\_cluster} \tag{10}$$

The generation method is similar to that near the borderline points. We assume the distribution of sample points in a single cluster obeys gaussian distribution. The equation11 is to generate new sample points near core points.

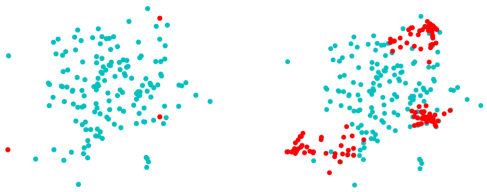
$$\begin{aligned} &cov\_mat = cov(core\_points) \\ &new\_data = multivariate\_normal(mean(core\_points), cov\_mat, n_{core}) \end{aligned} \tag{11}$$

The algorithm for generating new sample points at core points is algorithm3.

Generate data near noise points we use the most basic SMOTE for generation near the noise points. Gaussian distribution cannot be used for generation because effective clusters cannot be generated. Therefore SMOTE is used. A single sample is randomly selected from the K nearest neighbors of the minority point, and sample points are randomly generated on the line between the two points. We set $\gamma \in [0, 1]$ is a random value, the two points are $point1$ and $point2$, then the generation equation is 12.

$$new\_point = point1 + \gamma \times (point2 - point1) \tag{12}$$

As shown in Fig.3, (a) and (b) are distribution of original data and distribution of oversampled data. Minority samples in the dataset are too rare and far apart to form effective cluster, we cannot use gaussian distribution to generate new points, we use SMOTE instead. Algorithm *generate_noise* 4 is used to generate new sample points near the noise point.



(a) (b)

Fig. 3. (a) and (b) are distribution of original data and distribution of oversampled data. Minority points in the dataset are too rare and far apart to form effective cluster. We use SMOTE to generate new points.

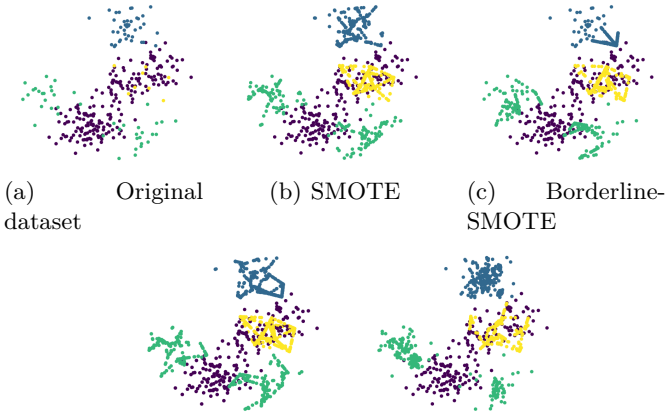
(a) Original (b) SMOTE (c) CCR (d) ODG

Fig. 4. (a) is Original dataset (b),(c),(d) are the oversampling results of SMOTE, CCR and ODG.

Translation of majority points New samples are generated according to the distribution characteristics of minority points. At the same time we also clean the majority points near minority points to alleviate the overlapping data distributions. We use a similar approach to the model CCR [15]. When dealing with borderline points, the K nearest neighbors of each minority point is calculated. With this minority point as the center and the maximum distance in the K neighbors as the radius, a spherical region is drawn. We move the majority points in these K neighbors out of this region.

We compare the result of SMOTE, CCR and ODG. We assume that some noise points exist in the original data set, as well as clustered minority points. The result shows in Fig.4. We can find that after SMOTE, there are overlapping regions, many generated minority points overlap with majority points. Although CCR algorithm avoided overlapping, but some minority points were synthesized near the noise, which may produce some interference to predicting results.

Computational complexity analysis We present the time complexity of the oversampling method ODG. We define the number of majority points is N_{maj}, the number of minority points is N_{min}, the total number of samples is N, and the number of attributes on the dataset is m. In the DBSCAN algorithm, the time complexity can be controlled at $O(mN_{min}\log(N_{min})$. which can be simplied to $O(mNlog(N))$. When calculating the K nearest neighbors of minority points, The distance between sample points needs to be calculated and sorted, and the complexity of this step is $O(mN_{min}N + N_{min}Nlog(N))$, it can be simplied as $O(mN^2)$. Therefore the time complexity of ODG is $O(mN(N + log(N)))$.



(a) Original dataset (b) SMOTE (c) Borderline-SMOTE

(d) SMOTE+ENN (e) MC-ODG

Fig. 5. (a) is Original dataset (b)(c)(d)(e) are oversampling result of SMOTE, Borderline-SMOTE, SMOTE+ENN and MC-ODG.

3.3 Multiple classification problem

Multiclassification on imbalance dataset is more difficult to solve than that of binary problem. In the binary classification, it is easy to distinguish the majority class and the minority class. However, in the multi-classification problem, the relationship between classes is more complex and difficult to deal. In addition, overlapping and noise problems will be more serious. The traditional OVA and OVO methods have their own disadvantages, [14, 16] presents a method to deal with multiclassification problems, Combine this method with ODG, we get our algorithm MC-ODG for multiclassification. We iterate over multiple classes. When we oversample each class, A subset is selected as a majority class from the already processed classes, and this class is treated as a minority class. We first sort the classes in reverse order according to the number of samples, and for each minority class, from the ones that have been processed, select a subset of the majority classes, and mark the minority class as the majority class after processing the ODG algorithm. Start from the second largerst class, each class can be treated as minority class. Each majority class has the same number of samples during sampling. This algorithm makes use of the distribution of other majority points when oversampling each class. The overall flow of the MC-ODG algorithm is shown in the algorithm5.

Fig.5 shows the original dataset and the oversampling results of SMOTE, Borderline-SMOTE, SMOTE+ENN and MC-ODG. We can find that the original data distributions can be retained after oversampling using the $MC-ODG$ algorithm.

Computational complexity analysis We calculate the time complexity of MC-ODG. We also define the total number of samples is N, and the number of attributes on the dataset is m. The number of classes is c. The entire MC-ODG process is equivalent to c-1 times ODG. Therefore the time complexity of MC-ODG is $O(cmN(N+log(N)))$.

Algorithm 5 MC-ODG

Input: $X \leftarrow$ a data matrix,which has multiple different labels.
$C \leftarrow$ different classes **Output:** $X \leftarrow$ A new data matrix which has been translated and oversamped.

for $i \leftarrow 1$ to $|C|$ **do**
  $n_{classes} \leftarrow$ number of classes with higher number of points than C_i
  if $n_{classes}>0$ **then**
    $X_{min} \leftarrow X^{C_i}$
    $X_{maj} \leftarrow \emptyset$
    for $j \leftarrow 1$ to $n_{classes}$ **do**
      add $\lfloor \frac{|X^{C_1}|}{n_{classes}} \rfloor$ randomly chosen points from $X^{(j)}$ to X_{maj}
    $X^{'}_{maj}, S = ODG(X_{maj}, X_{min})$
    $X^{C_i} \leftarrow X^{C_i} \cup S$
    substitute points used to construct X_{maj} with $X^{'}_{maj}$.
return X

Table 2. Confusion matrix for the two-class problem

Actual label	Predicted positive	Predicted negative
Positive	TP	FP
Negative	FP	FN

4 Experiments

In this section, we will detail a series of experiments on ODG and MC-ODG. We compared the ODG, MC-ODG algorithms with the existing oversampling methods, We published the code of our model on github[1]. We ran our code on Intel(R) Xeon(R) CPU E5-2678.

4.1 Evaluation matrics

We evaluated different oversampling algorithms using different metrics. Prediction accuracy is not a good metric on imbalanced problems. Due to the imbalance of data, the accuracy will naturally incline to the majority class. The predictive ability of the model for minority classes cannot be reflected by accuracy.

Binary classification We labeled the majority class as 0(Negative) and the minority class as 1(Positive). Precision, Recall, F-value and Auc are more reasonable metrics. Table2 gives the confusion matrix for the binary classification problems. Using the confusion matrix, we can calculate

[1] https://github.com/Xiaoctw/MC-ODG

Table 3. Binary and multiple classification datasets

Datasets	Instances	Features	Classes	Class distribution	IR
Transfusion(TR)	748	5	2	570/178	3.2
Wisconsin(WIS)	699	25	2	458/241	1.9
Adult(AD)	32561	108	2	24720/7841	3.2
Haberman(HA)	336	3	2	225/81	2.8
Glass(GL)	213	8	6	76/69/29/17/13/9	8.4
Automobile(AU)	159	25	6	48/46/29/13/3	16
Wine(WI)	157	12	3	71/58/48	1.5
Yeast(YE)	1484	7	10	463/429/244/163/51/44/35/30/20/4	115.6
Ecoli(EC)	336	6	8	143/77/52/35/20/5/2/2	71

the Precision, Recall and F-value. The calculation formulas are as follows:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F-value &= \frac{(1+\beta^2)\times Precision \times Recall}{\beta^2 \times Precision + Recall} \end{aligned} \tag{13}$$

The value of Auc is defined as the area bounded by the curve of *roc* and the coordinate axis. In general, the value of Auc is greater than 0.5. The closer the value is to 1, the better the prediction result is.

Multiple classification The evaluation of multiclassification is more complex, and there is not a unified evaluation standard and it is still an open problem. In addition to the above mentioned Precision, Recall and F-value, mGM and CBA [2] are also introduced. The equation of mGM and CBA are as follow:

$$CBA = \frac{\sum_{i=1}^{M} \frac{mat_{i,j}}{max(\sum_{j=1}^{M} mat_{i,j}, \sum_{j=1}^{M} mat_{j,i})}}{M} \tag{14}$$

$$mGM = \sqrt[M]{\Pi_{i=1}^{M} Recall_i} \tag{15}$$

4.2 Dataset

The dataset used in the experiment is the classification datasets on UCI[2], datasets' details are in table3.

4.3 Experiment results and analysis

We compared the prediction results of different oversampling models. In this experiment, we used three widely used machine learning models, C4.5, KNN and LR. We compared different oversampling methods, SMOTE, Borderline-SMOTE, ADASYN, SMOTE+ENN, MC-CCR and ODG or

[2] http://archive.ics.uci.edu/ml/datasets.php

Table 4. Parameters of different algorithms

Algorithm	Parameters	Algorithm	Parameters
MC-CCR	$energy \in [0.25, 0.5, 1]$ cleaning strategy: translation selection strategy: proportional multi-class decomposition method: sampling	ODG /MC-ODG	k=5 eps=0.08 min_pts=4 fit_borderline_ratio=true borderline_ratio=0.6 noise_ratio=0.1
SMOTE	k_neighbors $\in [3, 5, 7]$	KNN	k $\in [3, 5, 7]$
Borderline-SMOTE	k_neighbors $\in [3, 5, 7]$ kind="borderline-1"	C4.5	max_depth=5 min_samples_splits=3
ADASYN	n_neighbors $\in [3, 5, 7]$		
SMOTE+ENN	k_neighbors $\in [3, 5, 7]$		

MC-ODG. we adopted the cross validation method of K folding, set K=5, repeated the experiment for 10 times, and took the mean value as the final result. Parameters of ODG, MC-ODG and other algorithms are shown in table4. *fit_borderline_ratio* means being adaptive to determine the *borderline_ratio* or not. We gave a comparison of different oversampling methods on different datasets based using different machine learning models. The parameters of DBSCAN n are *eps* and *min_pts*. *borderline_ratio* of some datasets is artificially specified. Parameter settings on different datasets are shown in table5. When the parameter had multiple optional values, we took the optimal result. For binary datasets, KNN, C4.5 and LR results on multiple metrics are shown in

Table 5. Parameters of ODG, MC-ODG

Datasets	Parameters	Datasets	Parameters
TR	eps=0.15 min_pts=3 borderline_ratio=0.5	GL	eps=0.15 min_pts=3
WIS	eps=0.5 min_pts=3 borderline_ratio=0.7	AU	eps=1.8 min_pts=3 borderline_ratio=0.4 noise_ratio=0.2
AD	eps=1.6 min_pts=5 borderline=0.7	WI	eps=0.36 min_pts=2
HA	eps=0.14 min_pts=3	EC	eps=0.14 min_pts=3
		YE	eps=0.13 min_pts=3

table6, table7 and table8. For multiclassification datasets, KNN, C4.5 and LR results in table9 and table10 and table11.

In the oversampling of binary classification, we found that the ODG algorithm can achieve the best results on different metrics of most datasets. Best results can be obtained because we consider

Table 6. KNN model deals with binary classification datasets

Datasets	Precision				Recall				F1-score				Auc			
	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA
Original dataset	0.823	0.934	0.823	0.426	0.646	0.921	0.839	0.282	0.717	0.926	0.83	0.332	0.837	0.955	0.913	0.604
SMOTE	0.843	0.926	0.819	0.465	0.604	0.935	0.837	0.337	0.7	0.93	0.827	0.381	0.824	0.954	0.912	0.617
Borderline-SMOTE	0.853	0.93	0.824	0.421	0.606	0.91	0.839	0.269	0.706	0.919	0.83	0.319	0.827	0.949	0.912	0.621
ADASYN	**0.873**	**0.937**	0.819	**0.452**	0.618	0.912	0.843	0.36	**0.721**	0.924	0.829	0.383	0.834	0.95	0.913	0.619
SMOTE+ENN	0.71	0.921	0.797	0.412	0.677	0.953	0.876	0.537	0.69	0.936	0.831	0.46	0.806	0.958	0.912	0.64
CCR	0.738	0.925	0.794	0.368	**0.808**	0.94	0.862	**0.626**	0.709	0.931	0.825	0.457	0.834	0.958	0.909	0.622
ODG	0.8	0.921	**0.831**	**0.452**	0.75	**0.956**	**0.879**	0.517	0.711	**0.938**	**0.84**	**0.476**	**0.846**	**0.965**	**0.916**	**0.667**

Table 7. C4.5 model deals with binary classification datasets

Datasets	Precision				Recall				F1-score				Auc			
	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA
Original dataset	0.841	0.936	0.819	0.419	0.6	0.918	0.848	0.305	0.697	0.926	0.832	0.34	0.823	0.959	0.926	0.623
SMOTE	**0.844**	0.93	0.823	0.448	0.625	0.923	0.845	0.316	**0.714**	0.935	0.832	0.359	0.835	0.957	0.923	0.605
Borderline-SMOTE	0.839	0.931	0.812	**0.482**	0.587	0.922	0.852	0.315	0.684	0.925	0.83	0.373	0.816	0.958	0.924	0.594
ADASYN	0.843	0.928	0.815	0.475	0.596	0.924	0.848	0.338	0.694	0.924	0.83	0.388	0.822	0.957	0.924	0.627
SMOTE+ENN	0.697	0.919	0.789	0.41	0.673	0.954	**0.882**	0.552	0.678	0.936	0.834	0.449	0.799	0.959	0.927	0.628
CCR	0.751	0.918	0.79	0.367	**0.809**	**0.954**	0.88	**0.628**	0.688	0.935	0.832	**0.465**	0.836	0.965	0.924	0.635
ODG	0.755	**0.938**	**0.824**	0.429	0.681	0.951	0.859	0.524	0.672	**0.943**	**0.84**	0.455	**0.843**	**0.968**	**0.928**	**0.656**

Table 8. LR model deals with binary classification datasets

Datasets	Precision				Recall				F1-score				Auc			
	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA	TR	WIS	AD	HA
Original dataset	0.872	**0.962**	0.848	**0.691**	0.571	0.946	0.84	0.284	0.568	0.953	0.832	0.261	0.847	**0.996**	0.952	0.663
SMOTE	0.892	**0.962**	0.843	0.673	0.549	0.941	0.82	0.322	0.536	0.951	0.829	0.309	0.848	0.995	0.96	0.642
Borderline-SMOTE	0.871	0.961	**0.857**	0.653	0.553	0.946	0.824	0.286	0.549	0.953	0.837	0.313	0.845	0.992	0.961	0.663
ADASYN	**0.921**	0.958	0.852	0.658	0.553	0.95	0.828	0.321	0.646	0.953	**0.839**	0.309	0.847	0.995	0.96	0.644
SMOTE+ENN	0.779	0.951	0.775	0.536	0.701	0.962	0.895	**0.569**	0.663	0.956	**0.839**	0.443	0.847	**0.996**	0.957	0.658
CCR	0.69	0.951	0.779	0.525	0.763	0.967	0.884	0.445	**0.665**	0.958	0.832	0.396	0.848	0.965	0.957	0.635
ODG	0.655	0.956	0.775	0.543	**0.781**	**0.968**	**0.906**	0.457	0.657	**0.959**	0.834	**0.457**	**0.853**	**0.996**	**0.959**	**0.668**

Table 9. KNN model deals with multiclassification datasets

Datasets	Precision					Recall					F1-score					mGM					CBA				
	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE
Original dataset	**0.765**	0.84	0.57	0.962	0.5	0.718	0.78	0.575	0.96	0.492	0.716	0.767	0.555	0.954	0.465	0.713	0.761	0.651	0.958	0.467	0.643	0.717	0.497	0.92	0.419
SMOTE	0.742	0.839	0.572	0.962	0.506	0.695	0.793	0.554	0.969	0.496	0.692	0.802	0.541	**0.963**	**0.492**	0.643	0.762	0.593	**0.968**	0.484	0.618	0.748	0.484	0.931	0.45
Borderline-SMOTE	0.716	**0.843**	0.578	0.957	0.498	0.677	0.789	0.562	0.963	0.491	0.667	0.802	0.547	0.957	0.486	0.599	0.736	0.633	0.961	0.483	0.591	0.741	0.486	0.924	0.444
ADASYN	0.716	**0.843**	0.576	0.955	0.496	0.668	0.793	0.562	0.961	0.494	0.661	0.807	0.547	0.955	0.489	0.621	0.768	0.632	0.959	0.488	0.583	0.756	0.488	0.922	0.449
SMOTE+ENN	0.588	0.823	0.391	0.947	**0.51**	0.563	0.815	0.466	0.956	0.485	0.538	0.81	0.401	0.947	0.466	0.481	0.8	0.562	0.953	0.465	0.473	**0.757**	0.349	0.903	0.41
MC-CCR	0.719	0.774	0.603	0.955	0.455	0.737	0.803	0.642	0.967	0.5	0.706	**0.811**	0.597	**0.963**	0.482	0.713	0.775	0.652	0.958	**0.496**	0.626	0.717	0.525	0.934	0.445
MC-ODG	0.738	0.805	**0.637**	**0.963**	0.481	**0.75**	**0.815**	**0.664**	**0.97**	**0.511**	**0.721**	0.805	**0.622**	**0.963**	0.489	**0.725**	**0.803**	**0.658**	0.967	0.484	**0.641**	0.745	**0.546**	**0.935**	**0.453**

Table 10. C4.5 model deals with multiclassification datasets

Datasets	Precision					Recall					F1-score					mGM					CBA				
	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE
Original dataset	0.738	0.762	0.658	0.911	**0.541**	0.7	0.736	0.63	0.906	0.478	0.695	0.731	0.625	0.904	0.479	0.7	0.723	0.59	0.901	0.574	0.626	0.68	0.567	0.859	0.434
SMOTE	0.702	0.8	0.662	0.934	**0.541**	0.685	0.761	0.617	0.918	0.499	0.675	0.768	0.617	0.916	0.5	0.686	0.731	0.639	0.915	0.583	0.614	0.715	0.558	0.873	0.387
Borderline-SMOTE	0.741	**0.802**	0.67	0.915	0.532	0.718	0.75	0.631	0.908	0.497	0.709	0.759	0.622	0.908	0.498	0.679	0.728	0.594	0.905	0.57	0.642	0.701	0.552	0.869	0.439
ADASYN	0.749	0.789	0.646	0.915	0.539	0.716	0.748	0.619	0.915	0.501	0.708	0.755	0.611	0.912	0.504	0.695	0.739	0.619	0.912	0.576	0.637	0.695	0.543	0.881	0.449
SMOTE+ENN	0.577	0.761	0.45	0.906	0.491	0.564	0.739	0.528	0.907	0.442	0.524	0.726	0.464	0.901	0.443	0.542	0.71	**0.677**	0.902	0.523	0.495	0.658	0.411	0.853	0.387
MC-CCR	**0.773**	0.788	0.658	0.923	0.435	**0.748**	**0.788**	0.675	0.931	0.492	0.731	0.777	0.638	0.929	0.491	0.73	0.771	0.64	0.928	**0.605**	0.666	0.717	0.567	0.891	0.443
MC-ODG	0.762	0.797	**0.684**	**0.945**	0.501	0.738	0.786	**0.694**	**0.944**	**0.536**	**0.742**	**0.781**	**0.663**	**0.943**	**0.504**	**0.749**	**0.773**	0.634	**0.943**	0.584	**0.676**	**0.726**	**0.592**	**0.913**	**0.45**

Table 11. LR model deals with multiclassification datasets

Datasets	Precision					Recall					F1-score					mGM					CBA				
	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE	AU	EC	GL	WI	YE
Original dataset	0.668	0.657	**0.642**	0.984	**0.523**	0.62	0.607	0.575	0.98	0.416	0.617	0.679	0.615	0.973	0.423	0.709	0.702	0.583	0.984	0.521	0.594	0.538	0.519	0.963	0.471
SMOTE	**0.741**	0.674	0.622	0.984	0.548	0.676	0.601	0.594	0.985	0.425	0.681	0.675	0.608	0.984	0.436	0.678	0.716	0.584	0.985	0.533	0.618	0.544	0.525	0.969	0.48
Borderline-SMOTE	0.733	0.663	0.63	0.985	0.502	0.665	0.605	0.592	0.984	0.411	0.669	0.66	0.618	0.984	0.437	0.629	0.719	0.603	0.984	0.591	0.603	0.548	0.509	0.967	0.472
ADASYN	0.707	0.687	0.626	0.984	0.502	0.636	0.602	0.610	0.984	0.415	0.648	0.656	0.603	0.985	0.44	0.626	0.706	0.610	0.983	0.591	0.592	0.558	0.531	0.971	0.474
SMOTE+ENN	0.57	0.551	0.61	0.965	0.434	0.558	0.543	0.622	0.975	0.41	0.528	0.62	0.605	0.967	0.407	0.537	0.768	**0.679**	0.97	**0.67**	0.478	0.502	0.506	0.946	0.39
MC-CCR	0.718	0.679	0.624	0.983	0.493	0.711	0.693	**0.673**	**0.99**	0.515	0.701	0.672	0.63	0.983	0.537	0.691	0.866	0.652	0.985	0.581	0.644	0.628	0.566	0.869	0.517
MC-ODG	0.7	0.723	0.633	**0.988**	0.473	**0.74**	0.758	0.659	0.986	**0.52**	**0.705**	**0.723**	**0.657**	**0.989**	**0.558**	**0.763**	**0.882**	0.671	**0.99**	0.586	**0.646**	**0.663**	**0.577**	**0.979**	**0.525**

that the data may be distributed in clusters, and we adopt different oversampling methods for different kinds of sample points. Minority sample data were generated retaining the distribution characteristics of the original data, enhancing the ability of machine learning. It is obvious that the MC-ODG and MC-CCR are superior to other oversampling methods in multiclassification tasks. The reason is that both approaches take into account the relationships between different classes and can avoid overlapping data distributions. Meanwhile, the MC-ODG algorithm is superior to MC-CCR, because MC-ODG algorithm can retain the original data distribution when generating data. We also found the following additional conclusions:

- Good results can be achieved on the Precision metric without any oversampling progress. For example, best result obtained on WIS without oversampling. if we generate large numbers of minority sample points, the model will be inclined to predict the results as minority samples. In this way, the value of TP+FP will become larger, while the value of TP is limited, then the Precision will decrease.
- Different oversampling methods are suitable for different datasets. Such as CCR method is very suitable for TR, SMOTE+ENN is suitable for AD.
- CCR method can alse obtain good results, the reason is that the CCR method can effectively avoid overlapping and handle noise data.

5 Conclusions and future work

This paper presents new oversampling algorithms. Aiming at the problems of existing oversampling algorithms, we proposed ODG and MC-ODG algorithms to decrease the imbalance ratio. These algorithm applys DBSCAN and gaussian distribution to avoid the interference of noise and overlapping data distributions and obtained original distributions after oversampling. For multiclassification, the generation process makes use of the information of various classes in the original data.

In our future work, we will consider how to implement oversampling in a simpler way. We find that lots of parameters to be processed in ODG and MC-ODG. we will consider how to reduce the parameters, which can also achieve relatively good results.

References

1. P. Branco, L. Torgo, and R. P. Ribeiro. *A Survey of Predictive Modeling on Imbalanced Domains.* ACM, 2016.

2. Paula Branco, Luís Torgo, Rita P. Ribeiro, Paula Branco, Luís Torgo, Rita P. Ribeiro, Paula Branco, Luís Torgo, Rita P. Ribeiro, and Paula Branco. *Relevance-Based Evaluation Metrics for Multi-class Imbalanced Domains*. 2017.
3. Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Dbsmote: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36(3):664–684, 2012.
4. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.
5. Nv Chawla, N Japkowic, A Kotcz, and N Japkowicz. Editorial: Special issues on learning from imbalanced data sets. *Annals of Nuclear Energy*, 36(3):255–257, 2004.
6. Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.
7. Parr Ian David, Garfoot Robert James, and Walker Neil. Data distribution system and method, 2008.
8. Alberto Fernandez, Salvador Garcia, Nitesh V Chawla, and Francisco Herrera. Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.
9. Alberto Fernández, Victoria López, Mikel Galar, María José Del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110, 04 2013.
10. Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, 2005.
11. Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2008.
12. Jierui, Xie, Stephen, Kelley, Boleslaw, K., and Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm Computing Surveys*, 2013.
13. Mrs. A. Kanimozhi. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2010.
14. Micha Koziarski, Micha Woniak, and Bartosz Krawczyk. Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise. 2020.
15. Michal Koziarski and Michal Wozniak. Ccr: A combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer ence*, 27(4):727–736, 2017.
16. Bartosz Krawczyk, Michal Koziarski, and Michal Wozniak. Radial-based oversampling for multiclass imbalanced data classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–14, 2019.
17. Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information ences*, 250:113–141, 2013.
18. Hamed Masnadi-Shirazi and Nuno Vasconcelos. High detection-rate cascades for real-time object detection. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007.
19. Jesmin Nahar, Tasadduq Imam, Kevin S. Tickle, and Yi Ping Phoebe Chen. Computational intelligence for heart disease diagnosis: A medical knowledge driven approach. *Expert Systems with Applications*, 40(1):96–104, 2013.
20. Dana C. Peters, Frank R. Korosec, Thomas M. Grist, Walter F. Block, James E. Holden, Karl K. Vigen, and Charles A. Mistretta. Undersampled projection reconstruction applied to mr angiography. *Magnetic Resonance in Medicine*, 43(1):91–101, 2015.
21. Bhagat Singh Raghuwanshi and Sanyam Shukla. Class imbalance learning using underbagging based kernelized extreme learning machine. *Neurocomputing*, 329(FEB.15):172–187, 2019.
22. Li Sun, Ziwei Shang, Qing Cao, Kang Chen, and Jiyun Li. Electrocardiogram diagnosis based on smote+enn and random forest. 2019.

23. X. U. Xiaolong, Chen Wen, and Sun Yanfei. Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics*, (6), 2019.
24. Yuyan Zhang, Xinyu Li, Liang Gao, Lihui Wang, and Long Wen. Imbalanced data fault diagnosis of rotating machinery using synthetic oversampling and feature learning. *Journal of Manufacturing Systems*, 48(Part C):34–50, 2018.