A Classifier Combining Local Distance Mean and Centroid for Imbalanced Datasets

Yingying Zhao[1] and Xingcheng Liu[1,2(✉)]

[1] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China
isslxc@mail.sysu.edu.cn

[2] School of Information Science, Xinhua College of Sun Yat-sen University, Guangzhou 510520, China

Abstract. The K-Nearest Neighbor (KNN) algorithm is widely used in practical life because of its simplicity and easy understanding. However, the traditional KNN algorithm has some shortcomings. It only considers the number of samples of different classes in k neighbors, but ignores the distance and location distribution of the unknown sample relative to the k nearest training samples. Moreover, classes imbalance problem is always a challenge faced with the KNN algorithm. To solve the above problems, we propose an improved KNN classification method for classes imbalanced datasets based on local distance mean and centroid (LDMC-KNN) in this paper. In the proposed scheme, different numbers of nearest neighbor training samples are selected from each class, and the unknown sample is classified according to the distance and position of these nearest training samples. Experiments are performed on the UCI datasets. The results show that the proposed algorithm has strong competitiveness and is always far superior to KNN algorithm and its variants.

Keywords: K-Nearest Neighbor (KNN) · Local distance mean · Centroid · Classes imbalance · Classifier

1 Introduction

Many algorithms of machine learning, such as support vector machine [1], decision tree [2], Bayesian classification [3], etc, train a model from training samples, and then use the model to classify unknown samples. Unlike these model-based algorithms, the KNN algorithm [4] has no training process. It makes statistic on

Supported by the National Natural Science Foundation of China (Grant Nos. 61572534 and 61873290), the Special Project for Promoting Economic Development in Guangdong Province (Grant No. GDME-2018D004), and the Opening Project of Guangdong Province Key Laboratory of Information Security Technology under Grant 2017B030314131.

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 313, pp. 126–139, 2020.
https://doi.org/10.1007/978-3-030-41117-6_11

the number of each class in k training samples nearest to the unknown sample, and assigns unknown sample to the class that occupies the largest number in the k neighbours. The KNN method is not only easy to understand, simple to implement, but also has remarkable classification performance, which has been widely used in real life and has been rated as one of the top ten data mining algorithms [5]. However, there are some problems with the standard KNN algorithm, so researchers proposed a series of improved algorithms to overcome these shortcomings of KNN algorithm.

Firstly, sensitivity problem of k value. Different values of k have a great impact on the classification effect. Generally speaking, the method of cross validation is used to get an optimal k value. By introducing the training stage, a local k value is learned for each testing sample to improve the effect of k in these classifiers [6,7]. However, their complex training stages make the KNN algorithm lose its advantages of simplicity and convenience. Secondly, the relative distance between different samples are ignored and all samples within k neighboring training samples are treated equally in traditional KNN algorithm. Zeng et al. [8] weighted the distance, so that the neighbours who are closer get more weight. Similarly, the simple majority voting principle also ignores the spatial distribution of samples and fails to consider the relative positions of unknown sample and k neighbors. To solve this problem, Mitani et al. [9] used the local mean vector of k nearest neighbors (LMKNN) to classify unknown samples. On this basis, Pan et al. [10] improved it and proposed a new k-harmonic nearest neighbor classifier based on the multi-local means (MLMKHNN), which not only improved the classification accuracy, but also improved the robustness of k value. However, only one aspect of k value, distance and location distribution are considered in the above schemes.

What's more, the problem of class imbalance has always been a big challenge in classification problems, and it is a problem that needs to be considered in many machine learning algorithms. Because of the existing classification algorithms, the classification results for unknown samples are often biased towards the majority class. For example, the Naive Bayes classifier obtains a classification model by calculating the prior probability and the conditional probability, and then assigns the unknown sample to the class with the largest posterior probability according to the model. According to Bayes' theorem, prior probability is a very important part of calculating posterior probability. The KNN algorithm makes statistic on the number of each class in k training samples closest to the unknown sample, and assigns it to the class that occupies a larger number in the k neighborhoods. Whether it is Naive Bayes, KNN or other machine learning algorithms, although they sometimes seem to be able to achieve a good classification accuracy, they are biased against minority classes for imbalanced datasets.

However, the distribution of classes is often imbalanced in practice. For example, early warning of oil and gas leaks, detection of machine failures and identification of fraudulent calls, etc. In these examples, the amount of data on oil and gas leaks, machine failures, and fraudulent calls are much lower than the amount

of data in normal times. However, the traditional machine learning algorithms have the problem of improving the overall classification accuracy by misjudging the samples of minority class. This is very unscientific in practical applications. What we really need is to improve the classification accuracy of each class, especially the minority class (such as those that require early warning samples).

In this paper, we propose a new classification standard. The local distance mean and the centroid distance are combined to serve as the basis for classification. This approach takes into account the distance and position distribution of the training samples relative to the unknown sample. In addition, we propose a new method to deal with the problem of class imbalance. We opt different neighbors from different classes, which does not increase the computational complexity or reduce the sample information. The experimental results show that the proposed classification method perform well in both classes balanced datasets and classes imbalanced datasets, especially for classes imbalanced datasets. The LDMC-KNN algorithm proposed in this paper has a great advantage over the standard KNN algorithm and the latest KNN improved algorithms.

The rest of the paper is organized as follows: Sect. 2 reviews the related works. Section 3 elaborates on the proposed algorithm LDMC-KNN. Our experimental results are presented in Sect. 4 and our conclusion is given in Sect. 5.

2 Related Work

Suppose $T = \{x_n \in R^m\}_{n=1}^N$ is the given m dimensional feature space, while N is the total number of training samples, x_n represents the $n-th$ training sample, R^m is the m dimensional real vector R. $y_n \in \{c_1, c_2, ..., c_N\}$ is the label of the training sample x_n. $T_i = \{x_{ij} \in R^m\}_{j=1}^{N_i}$ represents the collection of $i-th$ class training samples, T_i is a subset of T in feature space. x_{ij} represents the j-th nearest training sample in the i-th class. Suppose the testing sample or unknown sample is represented as x.

2.1 KNN

The basic process of the KNN algorithm is as follows:

The Euclidean distance (Other distance measures can also be used) are calculated from testing sample x to each training samples:

$$dist(x_n, x) = \sqrt{(x_n - x)^{\mathrm{T}}(x_n - x)}. \tag{1}$$

The distances $dist(x_n, x)$ are sorted from small to large, and the k training samples closest to the testing sample are selected. The number of each class is counted in the k training samples, and the testing sample is classified into the class that accounts for the majority of the k training samples:

$$C_x = \arg\max_{c_i} \sum_{x_n \in X_k} \mathrm{L}(C_{x_n} = C_i). \tag{2}$$

C_x represents the class of x, X_k is the set of k nearest neighbor training samples including x_n. When the class of x_n is the i-th class, $\mathrm{L}(\bullet) = 1$, otherwise, $\mathrm{L}(\bullet) = 0$.

2.2 LMKNN

The basic process of the LMKNN algorithm is as follows:

For a testing sample x, k nearest training samples are selected from each subset T_i (The value of k is less than the training sample number n_{c_i} of each class). The method of distance measurement uses Euclidean distance:

$$dist(x_{ij}, x) = \sqrt{(x_{ij} - x)^{\mathrm{T}}(x_{ij} - x)}. \tag{3}$$

The local mean vectors (i.e. local centroid) are calculated using the k nearest training samples in each class:

$$u_{ik} = \frac{1}{k}\sum_{j=1}^{k} x_{ij}. \tag{4}$$

The distances from the local mean vector of each class to the testing sample are calculated:

$$U_{ik} = \sqrt{(u_{ik} - x)^{\mathrm{T}}(u_{ik} - x)}. \tag{5}$$

Finally, the testing sample x is classified to the class with the shortest distance:

$$C_x = \arg\min_{c_i} U_{ik}. \tag{6}$$

2.3 Imbalance Datasets

For classes imbalanced datasets, the solution can be roughly summarized into two types. The first approach is to pre-process the training set. It generally over-samples the minority class and/or under-samples the majority class to obtain the same number of training samples for each class. One of the most common under-sampling methods is called Random Under-Sampling [11], where majority class samples are randomly discarded until this class contains as many samples as other classes. However, it will lose some information of the training set, thus decreasing the classification accuracy. An over-sampling method is proposed in [12], in which the synthesized samples are introduced along the line segments connecting less than or equal to k minority class nearest neighbors. He et al. proposed a new adaptive synthesis method [13], where different weights are assigned to the different samples of minority classes according to the learning difficulty degree of different minority classes samples. Samples of minority classes that are difficult to learn generate more composite data than samples of minority classes that are easy to learn. However, these over-sampling method will introduce a large number of new samples, increase the computational complexity, and thus prolong the classification time.

The second method is to keep the original datasets unchanged and improve the classifier to relieve the class imbalance. Mullick et al. proposed a class-based global weighting scheme, named Global Imbalance Handling Scheme (GIHS) [6], which takes the ratio of ideal probability and current probability of a class as the

global weight related to this class. Zhang et al. [14] proposed k Rare-class Nearest Neighbour (KRNN) classification algorithm, which adjusts the posterior estimation of unknown samples to make it more partial to minority classes. Dubey et al. proposed a modified KNN algorithm [15]. In this method, the weighting factor for each class is calculated by classifying the neighbors of unknown samples using the existing KNN classifier. Li et al. suggested a training stage which exemplar minority class training instances are identified, and the samples of minority classes are extended to a Gaussian sphere [16], this method will make classification more sensitive to minority classes. Liu et al. proposed a class confidence weighting method [17], the samples are weighted by using the probability of attribute values given class labels in KNN algorithm. This approach can correct the preference of traditional KNN algorithm to majority classes. However, the above algorithms either introduce the training stage or need to adjust parameters, which increases the time complexity and eliminates the advantages of KNN algorithm that is simple and easy to implement.

3 Proposed Method

In this section, we propose a new method to eliminate the class imbalance while improving the accuracy of KNN classifier. First, we assume that the distribution of classes is balanced, the number of training samples of each class is the same, and KNN algorithm has no preference for each class. The standard KNN algorithm simply counts the number of classes of k neighbor samples, and does not care about the distance of the k samples. Therefore, under the condition that we guarantee the same number of training samples taken from each class (assuming that k training samples are taken from each class), to calculate the average distance between k training samples in each class and unknown sample.

For an unknown sample x, Its distance to all training samples are calculated using Euclidean distance:

$$dist(x_{ij}, x) = \sqrt{(x_{ij} - x)^{\mathrm{T}}(x_{ij} - x)}. \tag{7}$$

The training samples in each subset T_i are sorted in an increasing order according to their corresponding distances to the unknown sample x. And k nearest training samples are selected from each subset T_i, the corresponding distance $dist(x_{ij}, x), j = 1, ..., k$ are recorded. Then the average distance of the nearest k training samples to the unknown sample are calculated:

$$D_{ik} = \frac{1}{k}\sum_{j=1}^{k} dist(x_{ij}, x). \tag{8}$$

Furthermore, the position distribution of the training samples in each class relative to the unknown sample is considered. The centroid of k nearest training samples are calculated in each class:

$$u_{ik} = \frac{1}{k}\sum_{j=1}^{k} x_{ij}. \tag{9}$$

Then the distance from the centroid of each class to the unknown sample are calculated:

$$U_{ik} = \sqrt{(u_{ik} - x)^{\mathrm{T}}(u_{ik} - x)}. \tag{10}$$

Our ultimate goal is to find a class in which the average distance between k nearest training samples and unknown sample is the shortest (This means that the samples in this class are closer to the unknown sample), and the distance between the centroid of the k nearest training samples and the unknown sample is also the shortest. The shorter the distance between the unknown sample and the centroid, the stronger the enveloping ability of this class of samples to the unknown sample, and the greater the probability that the unknown sample belongs to this class. When the k training samples are uniformly distributed around the unknown sample, the centroid distance is 0.

Therefore, we combined the average distance and centroid distance of k nearest training samples as the basis for judging the class of unknown sample. The final judgment formula is:

$$C_x = \arg\min_{c_i} (U_{ik} + D_{ik}). \tag{11}$$

The above discussion is based on the assumption that the number of samples in each class is balanced. For the imbalanced datasets of classes, the number of training samples of different classes is different. If the same number of nearest neighbors from different classes are opted, it is unfair for the minority classes. Generally speaking, the distribution of samples of minority class is more sparse, the same number of nearest neighbors are opted as the majority class may cause the mean distance between the unknown sample and the nearest neighbors of the minority class to be larger.

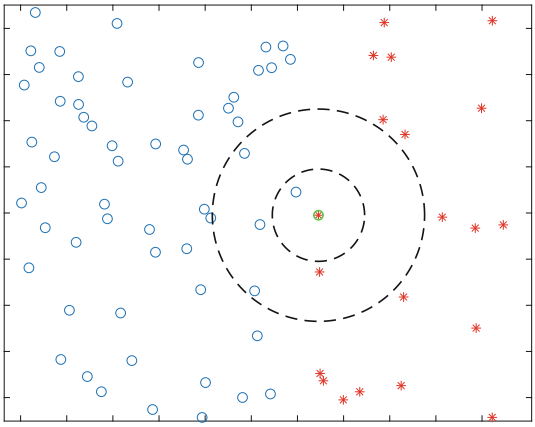

Fig. 1. Sample distribution example of class imbalanced dataset.

In terms of the sample distribution in Fig. 1. The ratio of the sample of the blue circle to the red asterisk is 3:1. It can be seen from the figure that if

the standard KNN algorithm is used, the samples of red asterisk close to the classification boundary are easily classified into blue circle class. As far as the samples of red asterisk surrounded by a green circle is concerned, no matter what the k is, it cannot be classified correctly. If we take different number of training samples according to the number of samples in each class. Samples of the majority class need to contribute samples further away from the unknown sample, which is equivalent to giving training samples of the majority class with less weight. The rate of misclassification of minority class samples decreases. In practical applications, it is very important to correctly identify the minority class in the unknown samples

Therefore, we eliminate the class imbalance problem by selecting different numbers of nearest neighbors from different classes. The specific method is as follows:

First, the number of classes $classNum$ and the number of training samples in each class are counted $N = \{n_{c_1}, n_{c_2}..., n_{c_{classNum}}\}$. According to the number of training samples of each class, the number of training samples selected in each class is determined. The class with the smallest training sample is used as a benchmark, and the k nearest training samples are selected from this class. Then the number of training samples selected from other classes is:

$$k_{c_i} = k * round(n_{c_i}/min(N)). \tag{12}$$

Since the number of samples selected must be an integer, $round(\bullet)$ is used to round it. Then, the distance mean and centroid distance of k_{c_i} training samples in each class were calculated. The unknown samples are classified into the class with the shortest combining local distance mean and centroid distance. Note that when the dataset is balanced, the algorithm degenerates to choose k training samples from each class, so the algorithm is equally applicable to the balanced datasets.

We substitute Eqs. (8, 9, 10, 12) into Eq. (11) to get the final judgment formula of the unknown sample:

$$C_x = \arg\min_{c_i} \Big(\sqrt{ \Big(\frac{1}{k_{c_i}} \sum_{j=1}^{k_{c_i}} x_{ij} - x \Big)^{\mathrm{T}} \Big(\frac{1}{k_{c_i}} \sum_{j=1}^{k_{c_i}} x_{ij} - x \Big) } + \frac{1}{k_{c_i}} \sum_{j=1}^{k_{c_i}} \sqrt{(x_{ij} - x)^{\mathrm{T}} (x_{ij} - x)} \Big). \tag{13}$$

The pseudo-code of LDMC-KNN is shown in Algorithm 1.

4 Experiments and Results

4.1 Degree of Imbalance

We use the imbalance ratio (IR) to quantify the imbalanced degree of classes. For the dataset of the two classes, IR is expressed as the ratio of the number of training samples of the majority class and the number of training samples of the minority class. For multi-class datasets, IR is defined as the maximum value of IR

Algorithm 1. The proposed LDMC-KNN classifier

Input: Training sample set T, training sample class set Y, unknown sample x, nearest neighbor number k

Output: The class of the unknown sample

1: Calculate the number of training sample classes $classNum$, and the number of training samples in each class $N = \{n_{c_1}, n_{c_2}..., n_{c_{classNum}}\}$
2: **for** $i = 1$ to $classNum$ **do**
3: $\quad k_{c_i} = k * round(n_{c_i}/min(N))$
4: **end for**
5: **for** $i = 1$ to $classNum$ **do**
6: $\quad$ **for** $j = 1$ to n_{c_i} **do**
7: $\quad\quad dist(x_{ij}, x) = \sqrt{(x_{ij} - x)^{\mathrm{T}}(x_{ij} - x)}$
8: $\quad$ **end for**
9: $\quad$ Sort the distance $dist(x_{ij}, x)$ and take out the first k_{c_i} training samples
10: $\quad u_{ik} = \frac{1}{k_{c_i}} \sum_{j=1}^{k_{c_i}} x_{ij}$
11: $\quad U_{ik} = \sqrt{(u_{ik} - x)^{\mathrm{T}}(u_{ik} - x)}$
12: $\quad D_{ik} = \frac{1}{k_{c_i}} \sum_{j=1}^{k_{c_i}} \sqrt{(x_{ij} - x)^{\mathrm{T}}(x_{ij} - x)}$
13: **end for**
14: $C_x = \arg\min_{c_i} (U_{ik} + D_{ik})$

between all two classes. Based on IR values, we divided the datasets into either balanced datasets ($IR \leq 1.15$), mildly imbalanced datasets ($1.15 < IR \leq 3.5$) and highly imbalanced datasets ($IR > 3.5$).

In this section, we use the UCI [18] datasets to demonstrate our proposed approach. The information for the 20 datasets is shown in Table 1. According to the IR value, we can see that the first 3 datasets are either balanced datasets, the middle 12 datasets are mildly imbalanced datasets, and the last 5 datasets are highly imbalanced datasets. (For the Segment, Led7dight, and Glass datasets, one class is used as the minority class, and the others are combined as the majority class, which is the same as in [14,19]). According to the information in Table 1, we can see the datasets used in our experiment is a good example of a wide range of number of instances, from 208 to 7400, and a wide range of number of features, from 3 to 60.

4.2 Indices for Evaluation of Classification Performance

We use the following three indices to evaluate the performances of classifiers:

Accuracy. For a testing set containing M testing samples, it is assumed that the number of correctly classified samples is m. Accuracy is defined as $accuracy = m/M$. The more the unknown samples can be correctly classified, the higher the accuracy is. However, it does not take into account the classification of each class, so it is not suitable to judge the class imbalance data. Therefore, in our experiment, we only use accuracy to evaluate the performance of the classifiers on the class either balanced datasets.

Table 1. Dataset description of 20 real-world datasets from UCI repository.

Dataset	Samples	Classes	Features	Class number ratio	IR
Ringnorm	7400	2	20	3736:3664	1.02
Waveform3	5000	3	21	1657:1647:1696	1.02
Sonar	208	2	60	97:111	1.14
Spambase	4597	2	57	2785:1812	1.54
Cloud	1024	2	10	627:397	1.58
Pima	768	2	8	268:500	1.87
Diabetes	768	2	8	500:268	1.87
Saheart	462	2	9	302:160	1.89
Tictactoc	958	2	9	626:332	1.89
Contraceptive	1473	3	9	629:333:511	1.89
German	1000	2	24	300:700	2.32
Breast	277	2	9	81:196	2.42
Haberman	306	2	3	225:81	2.78
Mammographic	748	2	4	278:570	2.81
Parkinsons	195	2	22	48:147	3.06
Hayesroth	160	2	4	129:31	4.16
Balance	625	3	4	49:288:288	5.88
Segment	2310	2	18	1980:330	6
Led7digit	500	2	7	455:45	10.11
Glass	214	2	9	17:185	10.88

Gmeans. Gmeans is a commonly used evaluation standard for imbalanced datasets. It is based on two classes of confusion matrices. Here, we extend Gmeans to multi-classes problem. We assume that the testing set contains a total of M samples, among which M_c testing samples belong to class c ($c = 1, 2, ..., classNum$), the number of correctly classified in class c is m_c. The calculation method of Gmeans is as follows:

$$Gmeans = \left(\prod_{c=1}^{classNum} (m_c/M_c) \right)^{1/classNum} \tag{14}$$

Compared with the accuracy, Gmeans takes into account the classification performance of each class, which is more suitable to be the judgment basis of imbalanced datasets of the class.

Area Under Receiver Operating Characteristics Curve (AUROC). The Receiver Operating Characteristics (ROC) Curve can comprehensively reflect the performance of the classifier, which is also the performance evaluation standard of the class imbalanced classifier. Researchers usually use the area under the

ROC curve, namely AUROC, to further quantify and compare the performance of classifiers. It is calculated as follows [6]:

$$AUROC = ((1 + TPR - FPR)/2), \tag{15}$$

where TPR represents true positive rate and FPR represents false positive rate. Here, minority class is seen as positively labeled. But the AUROC cannot be directly applied to multi-classes scenario, so we only use AUROC as the evaluation standard for two classes of imbalanced problems.

4.3 Experimental Procedure

In practice, in order to avoid the influence of different units and ranges of different dimensional features on the classification, it is necessary to standardize the features first. We use z-score standardization in our experiment

$$Z_i = \frac{X_i - E(X_i)}{\sqrt{D(X_i)}}, \tag{16}$$

where, X_i denotes the original i-dimensional sample feature, Z_i represents the i-th dimensional sample feature after standardization, $E(X_i)$ is the mean of the i-th feature samples, $\sqrt{D(X_i)}$ is the standard deviation of the i-th dimensional feature. Using Eq. (16), the original feature data can be normalized to a mean of zero and a variance of one. It makes data of different magnitudes to be converted to the same magnitude, increasing the comparability of the data. All experiments were conducted on the computer with Intel(R) Core(TM) i7-8700 CPU at 3.20 GHz, 16 GB RAM and Windows 10 64-bit Operating System running with the Matlab R2016b platform-based programs.

In the experiment, the samples are randomly divided into ten, one as the testing set, and the remaining nine as the training set. In order to ensure the fairness of the experiment, the partition of each experimental datasets is performed in the same dataset and is kept unchanged across the different algorithms, to ensure that the testing set and training set used in each algorithm are the same.

Four algorithms are compared in the experiment, which are standard KNN algorithm [4], MLMKHNN algorithm [10], Adaknn2GIHS algorithm [6] and AdaknnGIHS algorithm [6]. These four methods have been briefly introduced in the Sects. 1, 2, where MLMKHNN algorithm is an improvement of KNN algorithm, without taking into account the class imbalance problem. Adaknn2GIHS algorithm and AdaknnGIHS algorithm are proposed for class imbalance datasets to alleviate class imbalance problems, and two methods of adaptive k value are used in these two algorithms to improve the performance of classifiers. In the experiment, for the traditional KNN, the MLMKHNN and the LDMC-KNN proposed by us, the range of k value is 1–20, and each k value is cross-verified ten times to find the optimal k value, and then the corresponding classification performance is compared. For the Adaknn2GIHS algorithm and the AdaknnGIHS algorithm, since they are adaptive to select k value and have a lot of randomness.

Table 2. Comparison of classifiers in terms of Gmeans on imbalance datasets.

Dataset	KNN	LDMC-KNN	MLMKHNN	Adaknn2GIHS	AdaknnGIHS
Glass	0.2158	**0.6550**	0.2158	0.3983	0.4396
Led7digit	0.7966	**0.8945**	0.8765	0.8094	0.8199
Segment	0.9538	**0.9686**	0.9570	0.9388	0.9437
Balance	0.1428	**0.8208**	0.5282	0.5776	0.5246
Hayesroth	0.7371	**0.9786**	0.9628	0.7973	0.8044
Parkinsons	0.9362	**0.9426**	0.9362	0.9087	0.9053
Mammographic	0.6100	**0.6840**	0.5685	0.6575	0.6131
Haberman	0.5159	**0.6138**	0.4934	0.5352	0.5732
Breast	0.5708	**0.6612**	0.5998	0.5849	0.5704
German	0.6404	**0.7050**	0.5910	0.6545	0.6498
Contraceptive	0.4929	**0.5208**	0.4438	0.4750	0.4617
Tictactoc	0.7663	**0.8637**	0.8456	0.7652	0.7511
Saheart	0.5948	**0.6918**	0.6226	0.6498	0.6450
Diabetes	0.6786	**0.7544**	0.7267	0.7119	0.7056
Pima	0.7290	**0.7481**	0.7105	0.7175	0.7108
Cloud	0.9653	**0.9780**	0.9552	0.9572	0.9545
Spambase	0.9082	**0.9281**	0.9267	0.8970	0.9000

Table 3. Comparison of classifiers in terms of AUROC for two classes of imbalance datasets.

Dataset	KNN	LDMC-KNN	MLMKHNN	Adaknn2GIHS	AdaknnGIHS
Glass	0.5679	**0.7036**	0.5567	0.5707	0.6428
Led7digit	0.8523	**0.8976**	0.8854	0.8495	0.8564
Segment	0.9550	**0.9688**	0.9583	0.9394	0.9442
Hayesroth	0.8058	**0.9796**	0.9652	0.8539	0.8543
Parkinsons	0.9391	**0.9436**	0.9391	0.9115	0.9088
Mammographic	0.6503	**0.6897**	0.5792	0.6647	0.6254
Haberman	0.6150	**0.6418**	0.5845	0.5800	0.6004
Breast	0.6153	**0.6703**	0.6335	0.6019	0.5914
German	0.6524	**0.7074**	0.6234	0.6575	0.6543
Tictactoc	0.7905	**0.8712**	0.8551	0.7756	0.7611
Saheart	0.6432	**0.6959**	0.6481	0.6586	0.6548
Diabetes	0.6986	**0.7574**	0.7340	0.7226	0.7096
Pima	0.7336	**0.7517**	0.7093	0.7212	0.7156
Cloud	0.9657	**0.9781**	0.9558	0.9615	0.9550
Spambase	0.9084	**0.9282**	0.9270	0.8970	0.9003

We repeated the experiment ten times, and conducted cross validation ten times for each experiment, then take the average result as the basis for comparison.

Table 2 shows the Gmeans performance for 17 imbalanced datasets. We can find that the algorithm proposed in this paper is always better than and far superior to the other four algorithms on the comparison datasets. Table 3 shows the performance comparison of five classifiers in terms of AUROC for two classes of imbalanced datasets. It can also be seen that our proposed method has obvious advantages. This is because we not only consider the distance and location

distribution of each class of samples relative to the unknown samples, but also consider the problem of class imbalance.

Table 4. Comparison of classifiers in terms of accuracy on balance datasets.

Dataset	KNN	LDMC-KNN	MLMKHNN	Adaknn2GIHS	AdaknnGIHS
Sonar	0.9048	**0.9286**	0.9190	0.8476	0.8667
Waveform3	0.8520	**0.8548**	0.8402	0.8436	0.8476
Ringnorm	0.7511	**0.9431**	0.9296	0.6420	0.7286

What's more, to demonstrate that our algorithm is equally applicable to class-balanced datasets, we use three class-balanced datasets for a simple illustration (Because the algorithm proposed in this paper is mainly to solve the classes imbalance problem, we will not discuss the classes balance datasets too much here). Table 4 shows the classification accuracy of five algorithms on three balanced datasets, We can see that for class balanced datasets, although the advantages of our algorithm are not as great as it is for class imbalanced datasets, it is generally superior to the other four methods.

Table 5. The running times(s) of the five algorithms on different datasets.

Dataset	KNN	LDMC-KNN	MLMKHNN	Adaknn2GIHS	AdaknnGIHS
Ringnorm	**3.2252**	5.1621	6.5298	32.4742	32.3622
Waveform3	**1.4002**	2.3725	3.3218	14.5838	15.2613
Sonar	**0.0034**	0.0070	0.0224	0.0853	0.2596
Spambase	**1.3189**	2.0880	2.6676	13.2528	13.9233
Cloud	**0.0601**	0.1032	0.1924	0.8012	0.9646
Pima	**0.0341**	0.0605	0.1255	0.5228	0.7007
Diabetes	**0.0337**	0.0598	0.1252	0.5184	0.6896
Saheart	**0.0127**	0.0240	0.0616	0.2426	0.4188
Tictactoc	**0.0529**	0.0919	0.1761	0.7254	0.8894
Contraceptive	**0.1209**	0.2136	0.4039	1.6684	0.8198
German	**0.0594**	0.1022	0.1890	0.8433	1.0217
Breast	**0.0050**	0.0102	0.0319	0.1231	0.2868
Haberman	**0.0059**	0.0117	0.0352	0.1362	0.3060
Parkinsons	**0.0027**	0.0059	0.0208	0.0851	0.2523
Hayesroth	**0.0020**	0.0052	0.0226	0.0683	0.2367
Balance	**0.0233**	0.0457	0.1194	0.3773	0.5474
Segment	**0.3044**	0.5037	0.7295	3.3785	3.6780
Led7digit	**0.0150**	0.0279	0.0675	0.2579	0.4449
Glass	**0.0032**	0.0069	0.0228	0.0853	0.2701

Finally, we analyze the complexity of the algorithm. Table 5 shows the running times of the five algorithms on different data sets, running time is measured in seconds. As we can see, the running time of our algorithm is only longer than

the standard KNN algorithm, and the difference is very small. This is because our algorithm is compared with the standard KNN algorithm, it just has an extra work on the calculation of Eqs. (12) and (13). Although it seems that our algorithm has a loop nesting, it actually splits the entire large training set T into $classNum$ small subset T_i for calculation. Therefore, the amount of computation is not much different from the standard KNN. The running time of the MLMKHNN algorithm is slightly larger because it calculates multiple local mean vectors to calculate the harmonic average distance. The Adaknn2GIHS algorithm and the AdaknnGIHS algorithm introduce a relatively complex training stage. This training phase itself requires running KNN algorithms many times. Therefore, the Adaknn2GIHS algorithm and AdaknnGIHS algorithm require much longer running time.

5 Conclusions

In this paper, we propose an improved KNN algorithm based on combining local distance mean and centroid for imbalanced datasets. This method not only considers the distance from the unknown sample to each class, but also considers the position of the unknown sample in each class. In addition, the problem of class imbalance is solved by taking out different number of samples from different classes.

To evaluate the performance of the proposed LDMC-KNN algorithm, we compare it with the standard KNN and three state-of-the-art KNN-based approaches. The experiment was performed on the datasets of UCI database. Experimental results show that the performance (Gmeans and AUROC) of our proposed algorithm is far better than any of the other four algorithms on the imbalanced datasets. For the balanced datasets, our algorithm is also superior to other algorithms of interest in accuracy. Further, we compared the running times of the five algorithms. The experimental results show that the running time of our algorithm is not much different from the standard KNN algorithm, but it is obviously shorter than any of the other three improved KNN algorithms, demonstrating the advantages of our algorithm.

References

1. Wu, X., Zuo, W., Lin, L., Jia, W., Zhang, D.: F-SVM: combination of feature transformation and SVM learning via convex relaxation. IEEE Trans. Neural Netw. Learn. Syst. **29**(11), 5185–5199 (2018)
2. Safavian, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Trans. Syst. Man Cybern. **21**(3), 660–674 (1991)
3. Jiang, L., Zhang, L., Li, C., Wu, J.: A correlation-based feature weighting filter for Naive Bayes. IEEE Trans. Knowl. Data Eng. **31**(2), 201–213 (2019)
4. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(10), 21–27 (1967)
5. Wu, X., et al.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)

6. Mullick, S.S., Datta, S., Das, S.: Adaptive learning-based k-nearest neighbor classifiers with resilience to class imbalance. IEEE Trans. Neural Netw. Learn. Syst. **29**(11), 5713–5725 (2018)
7. García-Pedrajas, N., Romero del Castillo, J.A. Cerruela-García, G.: A proposal for local k values for k-nearest neighbor rule. IEEE Trans. Neural Netw. Learn. Syst. **28**(2), 470–475 (2017)
8. Zeng, Y., Yang, Y., Zhao, L.: Pseudo nearest neighbor rule for pattern classification. Pattern Recogn. Lett. **36**(2), 3587–3595 (2009)
9. Mitani, Y., Hamamoto, Y.: A local mean-based nonparametric classifier. Pattern Recogn. Lett. **27**(10), 1151–1159 (2006)
10. Pan, Z., Wang, Y., Ku, W.: A new k-harmonic nearest neighbor classifier based on the multi-local means. Expert Syst. Appl. **67**, 115–125 (2017)
11. Japkowicz, N.: The class imbalance problem: significance and strategies. In: Proceedings of the 2000 International Conference on Artificial Intelligence: Special Track on Inductive Learning, Las Vegas, pp. 111–117 (2000)
12. Chawla, N.V., Bowyer, K.W., Hall, L.O., Philip Kegelmeyer, W.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**(1), 321–357 (2002)
13. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IEEE International Joint Conference on Neural Networks, pp. 1322–1328. IEEE, Hong Kong (2008)
14. Zhang, X., Li, Y., Kotagiri, R., Wu, L., Tari, Z., Cheriet, M.: KRNN: k rare-class nearest neighbour classification. Pattern Recogn. **62**, 33–44 (2017)
15. Dubey, H., Pudi, V.: Class based weighted k-nearest neighbor over imbalance dataset. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 305–316. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_26
16. Li, Y., Zhang, X.: Improving k nearest neighbor with exemplar generalization for imbalanced classification. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 6635, pp. 321–332. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20847-8_27
17. Liu, W., Chawla, S.: Class confidence weighted kNN algorithms for imbalanced data sets. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 6635, pp. 345–356. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20847-8_29
18. Dua, D., Graff, C.: UCI machine learning repository (2019)
19. Zhang, X., Li, Y.: A positive-biased nearest neighbour algorithm for imbalanced classification. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 293–304. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_25