Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 9891971, 10 pages
https://doi.org/10.1155/2022/9891971

Research Article

A Neighborhood Model with Both Distance and Quantity Constraints for Multilabel Data

Xiaoli Jiang, Jing Zhou, Xinyue Qiao, Chang Peng, and Shiwen Su

College of Mathematical Sciences, Bohai University, Jinzhou 121013, China

Correspondence should be addressed to Xiaoli Jiang; jxls309@163.com

Received 25 June 2022; Accepted 26 July 2022; Published 19 September 2022

Academic Editor: Yugen Yi

In this paper, a novel distance-based multilabel classification algorithm is proposed. The proposed algorithm combines k-nearest neighbors (kNN) with neighborhood classifier (NC) to impose double constraints on the quantity and distance of the neighbors. In short, the radius constraint is introduced in the kNN model to improve the classification accuracy, and the quantity constraint k is added in the NC model to speed up computing. From the neighbors with the double constraints, the probabilities for each label are estimated by the Bayesian rule, and the classification judgment is made according to the probabilities. Experimental results show that the proposed algorithm has slight advantages over similar algorithms in calculation speed and classification accuracy.

1. Introduction

The multilabel classification problem stems from the text and image classification [1]. In practice, a text often has multiple keywords, and an image often has multiple scenes [2]. The authors of [3] proposed a multi-instance multilabel learning (MIML) approach for large datasets via subspace technique and stochastic gradient descent. To handle a data stream, an approach called multilabel learning with emerging new labels (MuENL) was proposed [4]. Combining the mixed dependency graph and the class cooccurrence, the authors of [5] constructed an optimization problem to deal with the label relevance for the multilabel learning with missing labels (MLML) and applies the ADMM algorithm to solve it. In reference [6], Zhang et al. reviewed the problem of binary correlation in multilabel learning.

The multilabel classification can be converted into a single-label problem by treating the labels as the vector value. However, the computational complexity is intolerable. In order to improve the computational speed, an improved algorithm called random k labelsets (RAkEL) was proposed [7]. Zhang and Zhou [8] established a multilabel version of kNN by Bayesian rule, and named it multilabel k-nearest neighbors (ML-kNN). In their experiments, ML-kNN was compared with the multilabel classification algorithms BOOSTEXTER [9], ADTBOOST.MH [10], and RANK-SVM [11]. ML-kNN treats each label as an independent binary classification and ignores the correlation between the labels. In order to make use of the correlation between the labels, the authors of [12] regarded the neighbor's labels as the features of instances and gave an extension of ML-kNN. For multilabel data streams, self-adjusting memory (SAM) [13] was adopted to deal with data drifting, and a multi-label k-nearest neighbor (ML-SAM-kNN) was established [14].

As well known, the kNN algorithm makes prediction by investigating the k-nearest neighbors of the unknown instances. Each instance is assigned a uniform parameter k. This is based on the assumption that the data is evenly distributed. In reality, most data are not uniformly distributed. The certainty factor measure [15] was designed for kNN classification to deal with to the skewed class distribution. Meanwhile, the shell neighbor imputation [16] fills the incomplete data by left and right nearest neighbors. For noisy data, the pseudo-nearest neighbors was identified by mutual k-nearest neighbor [17].

For a long time, the choice of parameter k was empirical. For example, the authors of [18] have tried to take it as the square root of the sample size, which does not address the problem of the uneven data. In view of this, the method of selecting different parameter k for different samples was proposed [19]. Different k values were learned by correlation matrices and assigned to different test data points [20]. A kTree method [21] to learn different optimal k values for different test samples was proposed with a sparse reconstruction model. The cost-sensitive kNN classifiers were designed and further improved with minimum-cost k-value selection, feature selection, and ensemble selection [22]. However, it will greatly increase the complexity of the kNN model and reduce the robustness of the algorithm. In extreme cases, if the different parameter is set for each sample, the computational complexity is unacceptable, especially for large data. In the [23], large data were first separated into several parts by k-means clustering, each of which has then conducted the kNN classification.

The kNN is a distance-based classifier, which estimates the labels of unknown instances according to the labels of the nearest neighbors. Therefore, the distance between the nearest neighbor and the unknown instance has a great impact on the accuracy of the estimation. In fact, with the increase of the distance from the unknown instance, the reference value of the nearest neighbors decreases gradually. In [24], a distance-weighted k-nearest neighbor algorithm (DW-kNNA) was introduced to solve a permanent magnet synchronous linear motor (PMSLM) model. Alfeilat, Hassanat, Lasassmeh, et al. showed that the performance of kNN classifier depends significantly on the distance used [25]. The l_{21}-norm based distance measurement was adopted in the loss function to improve the model robustness [26]. Meanwhile, the mathematical framework based on differential evolution with compressed sensing can learn the sparse module dictionaries and levels from the low-dimensional random composite measurements for reconstructing the high-dimensional data [27]. In 2020, a locality-constrained graph was introduced in the nonnegative matrix factorization algorithm to discover the geometric structure of the data [28]. A novel machine learning method based on the modified kNN algorithm was proposed. More features can be extracted from the datasets, and the datasets were updated during the training process instead of constructing the dataset beforehand [29], where a computational framework based on compressed sensing can be adopted to reduce dimensionality [30].

However, when the distance between the training instance and the unknown instance exceeds a certain value, the information of the nearest neighbors not only has no reference value for the judgment of the unknown instance, but also sometimes leads to misleading. Figure 1 shows a binary classification problem with nonuniformly distributed data. Assuming that the problem is linearly separable, the straight lines in the figure represents the classification hyperplane. The question mark "?" is an unknown instance, as shown in Figure 1(a), whose 3NN contains two positive instances and one negative instance. However, the correct judgment is that the unknown instance belongs to the negative class. The distance range investigated by the algorithm 3NN is too large in the sparse area. When the training instances are far away from the unknown instances, it means that their features are quite different from those of the unknown instances. At this point, it is easy to get wrong estimates if these training instances are considered as the classification basis. In view of the above analysis, we can assume that the referential meaning of the training instance is lost when its distance from the unknown instance exceeds a certain value. In other words, we just take the training instance within a certain range centered the unknown instance as classification basis.

Based on the above assumptions, we append a new parameter in the kNN model, which is the distance centered on an unknown instance. Only the k-nearest neighbors whose distance from the unknown instance is less than the given value are utilized for prediction. The neighbors for prediction are constrained by both quantity and distance. The distant constraint aims to pick up a certain number of neighbors among the k-nearest neighbors. As shown in Figure 1(b), the label of the unknown instance can be correctly judged by the nearest neighbor with the distance constraint. At this point, the two positive instances in 3 nearest neighbors will no longer be used as reference.

In the neighborhood classifier, the neighborhood contains a large number of instances for the data with high distribution density, especially numerical data. The computation complexity will be greatly increased by counting the label information of the neighbors in the neighborhood. In order to improve the computational efficiency, we only consider the k-nearest neighbors in the neighborhood as the reference for classification.

Different from simple label classification and the classification based on density estimation, multilabel classification predicts a set of labels associated with each sample, where the number and category of labels are both random. Therefore, the complexity of multilabel classification is exponentially higher than that of unilabel classification and the classification based on density estimation, which is also one of the main problems faced by multilabel classification. In order to alleviate the exponential growth, multilabel classification tasks are decomposed into independent single-label classification tasks in this paper. It not only reduces the complexity of the problem, but also facilitates parallel computing.

In Section 2, we present a mathematical description of the multilabel classification and introduce some of the symbols and concepts to be used. A novel multilabel classification algorithm based on ML-kNN and the neighborhood classifier will be presented in Section 3. In order to verify the effectiveness of the algorithm, we selected some common multilabel data and carried out some comparative experiments. The results will be reported and analyzed in Section 4.

2. Preliminaries

Multilabel training data are composed of features and labels. The features are measurable properties of an instance. The labels represent the classes to which the instance belongs. The purpose of classification learning is to train a classifier from the feature and label data. The classifier can predict the class labels from the measurable features.

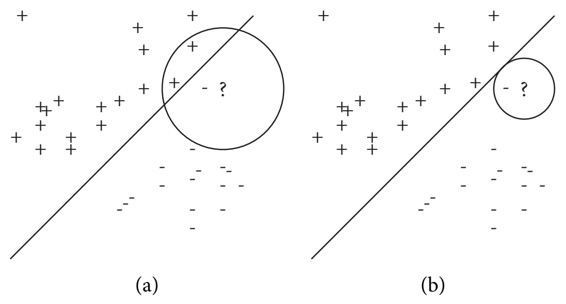

Figure 1: (a) 3NN and (b) bounded 3NN for nonuniformly distributed data.

Let X be a nonempty finite set of instances. An instance can be denoted as a vector $x = (x_1, x_2, \ldots, x_n) \in X$, which implies that its feature values are $x_1, x_2, \ldots, x_n$, respectively. The numerical and nominal feature values are expressed as real numbers and natural numbers, respectively. The distance between the instances is defined as

$$d(x, x') = \sqrt{\sum_{i=1}^{n} d_{i(x_i, x_i')}}, x, x' \in X. \quad (1)$$

For numeric features, let

$$d_i(x_i, x_i') = (x_i - x_i')^2,$$
$$d_i(x_i, x_i') = \begin{cases} 1, & x_i = x_i', \\ 0, & x_i \neq x_i', \end{cases}, x, x' \in X, \quad (2)$$

be nominal features. For each instance $x \in X$, let $N^k(x)$ denote the set of kNNs of x, and the set $N_\delta(x) = \{y \in X\colon 0 < d(x, y) \leq \delta\}$ is said to be the δ-neighborhood of the instance x. The elements in the intersection

$$N_\delta^k(x) = N_\delta(x) \cap N^k(x), \quad (3)$$

are called δ-kNNs of the instance x.

Let L be the collection of labels. The multilabel classification is to learn a function $h\colon X \longrightarrow 2^L$ from the training data

$$D = \{(x, L_x)\colon x \in X\}, \quad (4)$$

where the label subset $L_x \subset L$ is the set of labels associated with the instance x. Each label $l \in L$ defines a random variable $l\colon X \longrightarrow \{0, 1\}$. If the instance x has the label $l \in L$, then $l(x) = 1$, otherwise, $l(x) = 0$. A random variable $C^k\colon X \longrightarrow \{0, 1, \ldots, k\}$ is defined as

$$C^k(x) = \sum_{x' \in N^k(x)} l(x'), \forall x \in X, \quad (5)$$

to count the number of the instances with the label $l \in L$ in the neighborhood $N^k(x)$. In the same way, a random variable $C_\delta\colon X \longrightarrow Z^+$ is defined as

$$C^k(x) = \sum_{x' \in N_\delta(x)} l(x'), \forall x \in X, \quad (6)$$

to count the number of the instances with the label $l \in L$ in the neighborhood $N_\delta(x)$, where Z^+ is the positive integer set.

Multilabel k-nearest neighbor (ML-kNN) makes prediction with the label information embodied in the k neighbors by the maximum a posteriori (MAP) rule. The pseudo-code of ML-kNN is shown in Table 1. Here, we only listed the basic framework to illustrate its idea, see [1, 8] for details.

The innovation of this paper lies in the establishment of a new algorithm combining the ideas of ML-kNN and neighborhood classifier (NC) [31]. The pseudo-code of NC is given in Table 2 to illustrate the basic idea of the proposed algorithm.

From Table 1 and Table 2, it can be seen that the difference between the two algorithms lies in the definitions of the neighborhoods. The algorithm ML-kNN limits the number of the neighbors, while NC restricts the distance of neighbors. The proposed algorithm in this paper will double constrain the nearest neighbor from both quantity and distance.

3. Bounded ML-kNN

In order to reduce the computational burden, a new radius parameter is introduced into the kNN model to reduce the number of neighbors, and a novel mult-label classification algorithm is established in this section.

We decompose the multilabel data D into a series of binary classification data.

$$D_l = \{(x, l)\colon x \in X\}, l \in L, \quad (7)$$

which are single-label data with the same feature data as the multilabel one D. Then, the classifier deals with each binary classification independently. The neighborhood $N_\delta^k(x)$ of each instance $x \in X$ is calculated according to the distance defined in Section 2. The random variables $C\colon X \longrightarrow \{0, 1, \ldots, k\}$ can be defined as

$$C(x) = \sum_{x' \in N_\delta^k(x)} l(x'), \forall x \in X, \quad (8)$$

to count the number of the instances with the label $l \in L$ in the neighborhood $N_\delta^k(x)$. The label l of each instance $x \in X$ is estimated by maximizing the posteriori probability $p(l \mid C)$, i.e.,

Table 1: Pseudo-code of ML-kNN.

Input:	D, k, and unknown instance t
Output:	L_t
Step 1	For $x \in X$
Step 2	Identify k neighbors $N^k(x)$
Step 3	End for
Step 4	For $l \in L$
Step 5	Learn the probabilities $p(C^k(x) = j)$ for $j = 1, 2, \ldots, k$.
Step 8	End for
Step 9	Identify k neighbors $N^k(t)$
Step 5	For $l \in L$
Step 6	Calculate the statistics $C^k(t) = \sum_{x' \in N^k(t)} l(x')$.
Step 8	End for
Step 7	Return L_t.

Table 2: Pseudo-code of NC.

Input:	D, k, and unknown instance t
Output:	L_t
Step 1	For $x \in X$
Step 2	Identify k neighbors $N_\delta(x)$
Step 3	End for
Step 4	For $l \in L$
Step 5	Learn the probabilities $p(C_\delta(x) = j)$ for $j = 1, 2, \ldots, k$.
Step 8	End for
Step 9	Identify k neighbors $N_\delta(t)$
Step 5	For $l \in L$
Step 6	Calculate the statistics $C_\delta(t) = \sum_{x' \in N_\delta(t)} l(x')$.
Step 8	End for
Step 7	Return L_t.

$$l(x) = \mathbf{arg\,max}_{l \in \{\mathbf{0},\mathbf{1}\}}\, p(l \mid C(x)), \tag{9}$$

where the symbol $C(x)$ represents the random event $C = C(x)$ that there are $C(x)$ instances with the label l in the neighborhood $N_\delta^{\mathbf{k}}(x)$ of the instance $x \in X$. The number $C(x)$ could be any positive integer between 0 and k. Bayes' theorem implies

$$p(l \mid C) = \frac{p(l)p(C \mid l)}{p(C)}. \tag{10}$$

Since $p(C)$ is constant whether $l = \mathbf{1}$ or $l = \mathbf{0}$, the optimization problem can be reduced to

$$l(x) = \mathbf{arg\,max}_{l \in \{\mathbf{0},\mathbf{1}\}}\, p(l)p(C(x) \mid l). \tag{11}$$

In the training phase, we learn the probability $p(l)$ from the training data D_l. The symbol $|X|$ represents the cardinality of the set X, i.e., the total number of instances in the set X. The ratio

$$\frac{\sum_{x \in X} l(x)}{|X|}, \tag{12}$$

is taken as an estimate of the probability $p(l = \mathbf{1})$. For the convenience of computer calculation, we introduce an auxiliary parameter s, so that the probabilities can be approximated as

$$p(l = \mathbf{1}) = \frac{s + \sum_{x \in X} l(x)}{\mathbf{2}s + |X|}, \tag{13}$$

$$p(l = \mathbf{0}) = \mathbf{1} - p(l = \mathbf{1}). \tag{14}$$

The second probability to be learned is the conditional probability $p(C \mid l)$. Consider the following instance sets

$$\begin{aligned} D_j^1 &= \{x \in X\colon C(x) = j, l(x) = 1\}, j = 0, 1, \ldots, k, \\ D_j^0 &= \{x \in X\colon C(x) = j, l(x) = 0\}, j = 0, 1, \ldots, k, \end{aligned} \tag{15}$$

which are the sets of the training instances x with the label l (when $l(x) = \mathbf{1}$) or without the label l (when $l(x) = \mathbf{0}$), while each x's neighborhood $N_\delta^{\mathbf{k}}(x)$ contains exactly j neighbors with the label l. Therefore, the conditional probabilities $p(C = j \mid l = \mathbf{1})$ can be approximated by the ratio

$$\frac{\left|D_j^{\mathbf{1}}\right|}{\sum_{j=\mathbf{0}}^{\mathbf{k}} \left|D_j^{\mathbf{1}}\right|}, \tag{16}$$

where the cardinality of the set $D_j^{\mathbf{1}}$ is denoted by $|D_j^{\mathbf{1}}|$, which is equal to the number of the instances in the set $D_j^{\mathbf{1}}$. The auxiliary parameter s is introduced to establish the estimation

$$p(C = j \mid l = \mathbf{1}) = \frac{s + \left|D_j^{\mathbf{1}}\right|}{s(\mathbf{k} + \mathbf{1}) + \sum_{j=\mathbf{0}}^{\mathbf{k}} \left|D_j^{\mathbf{1}}\right|}. \tag{17}$$

Similarly, we can obtain the estimation

$$p(C = j \mid l = \mathbf{0}) = \frac{s + \left|D_j^{\mathbf{0}}\right|}{s(\mathbf{k} + \mathbf{1}) + \sum_{j=\mathbf{0}}^{\mathbf{k}} \left|D_j^{\mathbf{0}}\right|}. \tag{18}$$

For each unknown instance t, the label is determined by the following principle:

$$l(t) = \arg\max_{l \in \{0,1\}} p(l)p(C(t) \mid l). \tag{19}$$

The pseudo-code for bounded ML-kNN (BML-kNN) is presented in Table 3.

In the pseudo-code, the multilabel classification is decomposed into the independent single-label problem for each label $l \in L$. From Step 2 to Step 16, the probabilities $p(l)$ and $p(C \mid l)$ are learned from the single-label training data. The counter $c(j)$ records the number of the training instances with the label l, and whose neighborhood contains j instances with the label l for each $j = 0, 1, 2\cdot, k$. Simultaneously, the counter $\bar{c}(j)$ is the total number of training instances without the label l, and whose neighborhood contains j instances with the label l. From Step 17 to Step 20, the optimization problem can be solved by applying the probabilities (13), (14), (17), and (18). The labels of the unknown instances are given one-by-one.

4. Experiment Results

In this section, some experimental results are reported to compare our proposed methods with ML-kNN. The experiments were conducted on a computer with Intel(R)

Table 3: Pseudo-code of BML-kNN.

Input:	X, L, k, radius $\delta > 0$, and unknown instance t
Output:	$l(t), l \in L$
Step 1	For $l \in L$
Step 2	Compute $p(l = 1)$ and $p(l = 0)$ as (13) and (14)
Step 3	$c = \text{zeros}(k+1), \bar{c} = \text{zeros}(k+1)$
Step 4	For $x \in X$
Step 5	Compute $C(x)$ as (8)
Step 6	If $l(x) == 1$
Step 7	$c(C(x)) = c(C(x)) + 1$
Step 8	Else
Step 9	$\bar{c}(C(x)) = \bar{c}(C(x)) + 1$
Step 10	End if
Step 11	End for
Step 12	For $j = 0$: k
Step 13	$\lvert D_j^1 \rvert = c(j)$, $\lvert D_j^0 \rvert = \bar{c}(j)$
Step 14	Compute $p(C = j \mid l = 1)$ and $p(C = j \mid l = 0)$ as (17) and (18)
Step 15	End for
Step 16	End for
Step 17	For $l \in L$
Step 18	Compute $C(t)$ as (8)
Step 19	Compute $l(t)$ as (19)
Step 20	End for

Table 4: Multilabel data sets.

Dataset	Type	Instances	Features	Label	Domain
Enron	Nominal	1702	1001	53	Text
Medical	Nominal	978	1449	45	Text
Core15k	Nominal	5000	499	374	Images
Genbase	Nominal	662	1185	27	Biology
Yeast	Numerical	2417	103	14	Biology
Emotion	Numerical	593	72	6	Music
CAL500	Numerical	500	62	174	Music
Scene	Numerical	2407	294	6	Images
Mediamill	Numerical	43907	120	101	Video
Nus-wide	Numerical	269648	129	81	Images

Core(TM) i5-7200U CPU at 2.50 GHz and 16 GB memory. Some common and typical multilabel datasets were selected as experimental data. Four are symbolic data and four are numerical data. The datasets are all available at http://mlkd.csd.auth.gr/multilabel.html/#Datasets and http://meka.sourceforge.net/#datasets. Detailed information about the datasets is listed in Table 4. We divided each data set into two parts, eight tenths of which were used as training sets and two as test sets.

First, we examine the sensitivity of the classification effect with respect to the radius of the neighborhood. We run the algorithm at different radii and calculate the corresponding classification metrics. The curves of the classification measures varying with radius are drawn in the figures. The experimental results show that the variation on each dataset is roughly the same. Here, we only list the results for the datasets Scene and Yeast, as shown in Figures 2 and 3.

It can be seen from Figures 2 and 3 that, with the increase of radius, the five classification evaluation measures get better uniformly. At first, the classification effect does not increase significantly and increases sharply when the radius reaches a certain value. However, when the radius increases to a certain extent, the values of the five classification evaluation measures are almost unchanged. On the other hand, corresponding to different parameter k, the trend of the curve is basically similar.

From the above analysis, we can conclude that there is an optimal radius for each dataset, and different data correspond to different optimal radius, which does not change with the change of parameter k.

In the second experiment, with a given radius, we investigated the sensitivity of the classification accuracy to the parameter k. We also draw the curve as shown in Figure 4. We can see that the accuracy curve varies greatly at different radii. When the radius is too small, the parameter k does not play any role in the algorithm. For example, the blue curve in Figure 4 when the radius is 0.6. This indicates that the influence of radius on accuracy is more significant than that of parameter k in this algorithm. On the other hand, we can roughly conclude from the figures that the accuracy increases with the increase of k in the first stage, but decreases when k reaches a certain value. Similar to the radius, there is also an optimal parameter k for each dataset.

The last pictures in Figures 2, 3, 4, and 5 show the variation of running time with parameters k and radius. The running time tends to increase slightly as the two parameters increase, but not significantly. This means that the two parameters have little effect on the speed of the algorithm.

Figure 5 shows the joint effects of two parameters on classification accuracy. When the parameter k is below 10, its influence on classification accuracy is obvious. When the parameter k is above 10, the classification accuracy is mainly determined by the radius. We can see the parallel mountains in the graph, their height depends on the radius. This indicates that, relative to parameter k, the radius is the determinant of the classification effect. From the point of experiment, it is reasonable to introduce the radius as a parameter in this paper.

In the second part, we compare the operation speeds of five algorithms neighborhood classifier (NC), ML-kNN, BOOSTEXTER, RANK-SVM, and bounded ML-kNN (BML-kNN). Under the same experimental conditions, the algorithms are applied to some generic datasets, and the average running time (s) are shown in Table 5. The underlining indicates the best of the three algorithms.

The experimental results show that the algorithm NC is not advantageous in running time compared with the other algorithms, especially for some datasets with high density. The algorithms ML-kNN and BML-kNN are faster than the other algorithms. However, BML-kNN has a slight numerical advantage, and it performed well on six of the ten datasets, especially on the larger datasets Mediamill and Nus-wide.

In the last part of the experiment, we try to compare the classification accuracy of the three algorithms. At present, the evaluation criteria of multilabel classification in related literature are various. In this paper, we chose five common metrics Hamming loss, One-error, Coverage, Ranking loss,

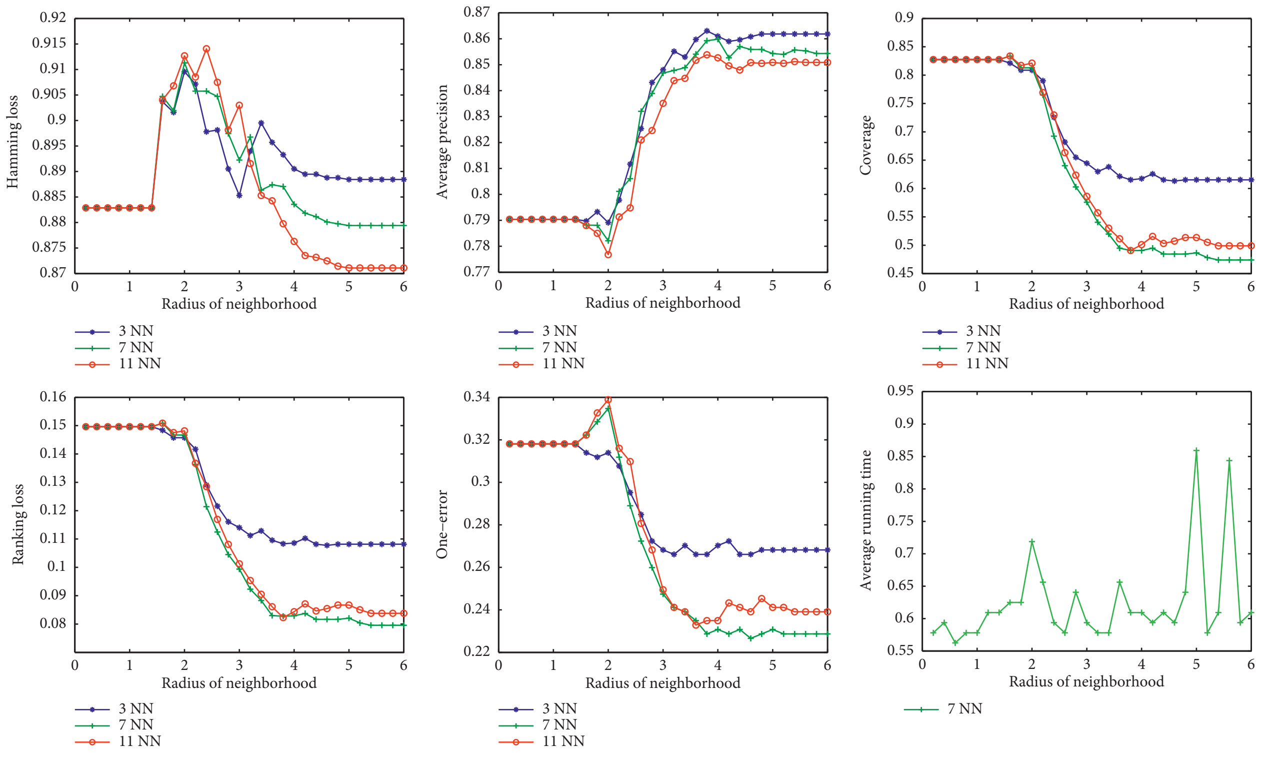

Figure 2: Variation of evaluation metrics with radius of neighborhood on dataset Scene.

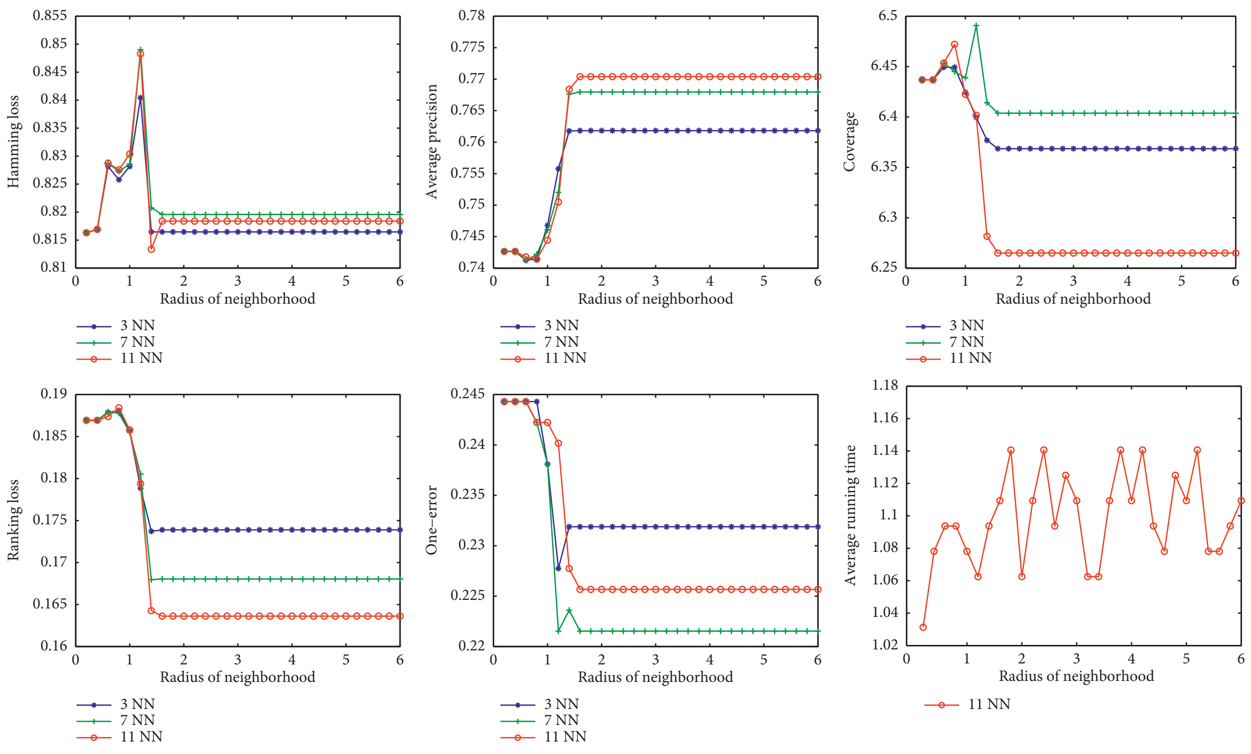

Figure 3: Variation of evaluation metrics with radius of neighborhood on dataset Yeast.

and Average precision. For the definitions and calculation methods, please refer to Reference [1]. Among them, the larger the value of metric Average precision is, the better the classification effect is, and the smaller the value of other measures, the better the classification accuracy. In this paper, we apply the five algorithms to the same datasets and compare the five classification metrics to measure the classification effect of the algorithms.

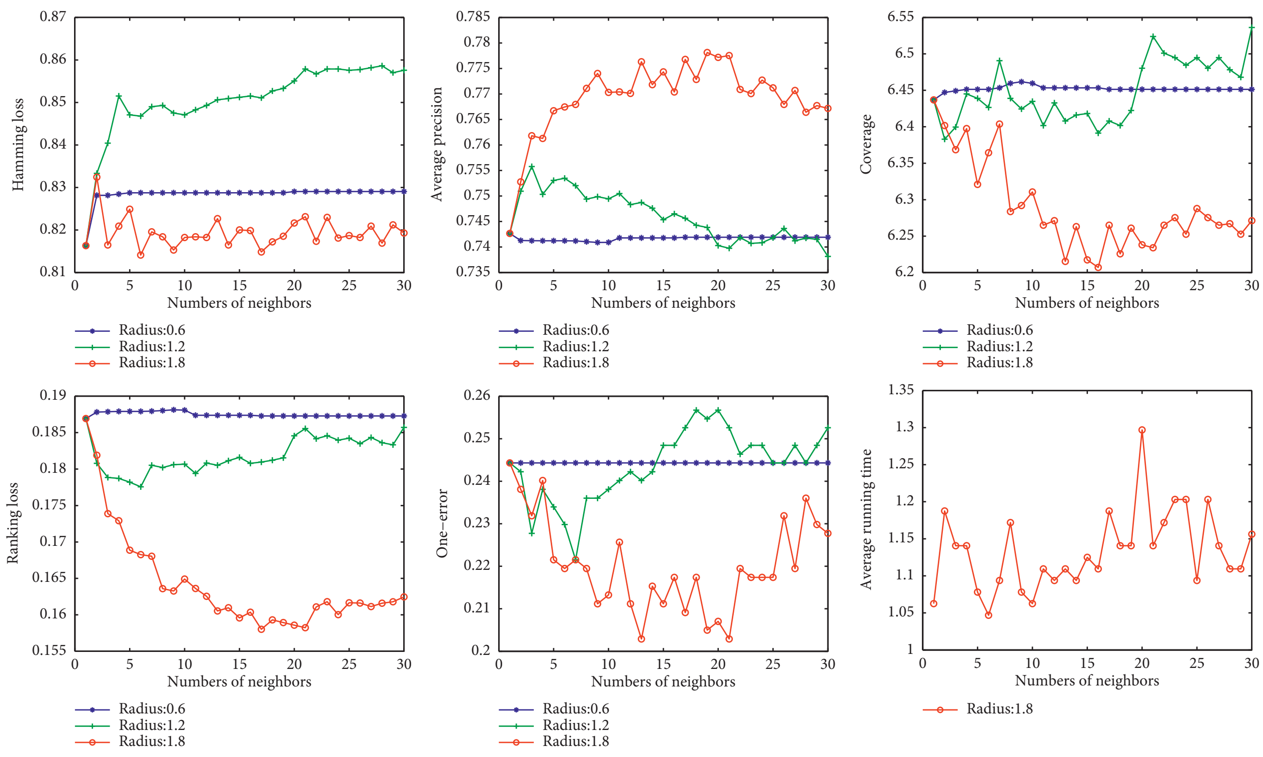

Figure 4: Variation of evaluation metrics with parameter k on dataset Yeast.

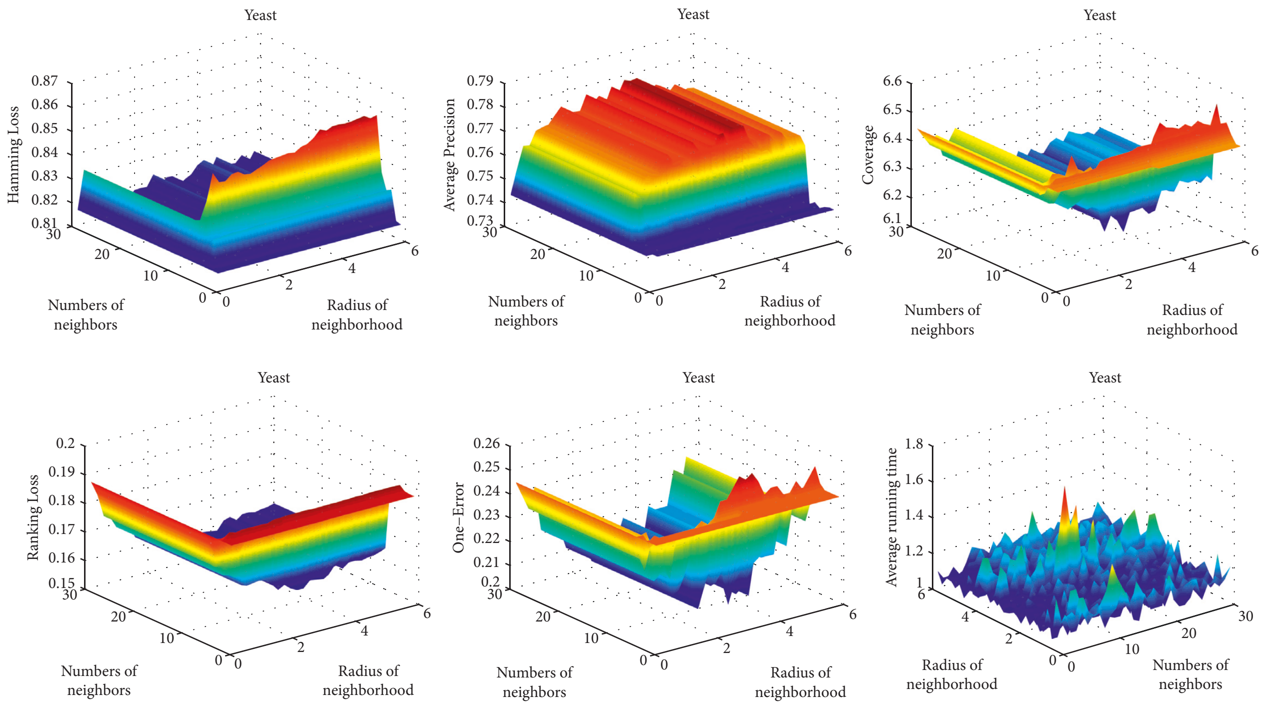

Figure 5: Variation of evaluation metrics with parameter k and radius on dataset Yeast.

Tables6–10 show theclassification accuracy under differentmeasures. The numerical values shown in the tables are the average of ten parallel experiments, where the underlines highlight the best over the other algorithms.

As shown in Table 6, from the Hamming loss corresponding to the algorithms, the proposed algorithm BML-kNN has a slight advantage over the other algorithms. However, it is not significant, the numerical difference is the third significant digit in general.

Table 5: Average running time.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	1.73	3.65	1.78	1.88	1.12
Emotion	0.49	0.26	0.33	0.58	0.11
Genbase	0.16	0.33	0.17	0.23	0.19
Scene	1.12	1.35	1.18	1.16	0.61
Enron	0.84	2.52	0.86	0.89	0.97
Medical	0.34	0.82	0.62	0.78	0.39
Core15k	12.1	46.4	18.2	19.2	15.8
CAL500	1.53	1.87	1.49	1.64	1.43
Mediamill	82.1	90.7	87.0	81.1	76.6
Nus-wide	565	596	529	581	547

Table 6: Hamming loss.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	0.22	0.26	0.28	0.27	0.21
Emotion	0.27	0.29	0.26	0.24	0.23
Genbase	0.04	0.07	0.05	0.06	0.03
Scene	0.09	0.09	0.11	0.07	0.10
Enron	0.06	0.07	0.08	0.07	0.05
Medical	0.01	0.02	0.03	0.03	0.02
Core15k	0.01	0.02	0.02	0.02	0.02
CAL500	0.11	0.13	0.12	0.12	0.09
Mediamill	0.03	0.04	0.05	0.04	0.02
Nus-wide	0.02	0.02	0.03	0.02	0.01

Table 7: One-error.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	0.25	0.29	0.26	0.28	0.22
Emotion	0.38	0.39	0.36	0.37	0.35
Genbase	0.02	0.02	0.03	0.01	0.02
Scene	0.24	0.31	0.27	0.28	0.22
Enron	0.31	0.34	0.38	0.41	0.39
Medical	0.27	0.28	0.31	0.28	0.26
Core15k	0.74	0.73	0.79	0.76	0.80
CAL500	0.12	0.15	0.17	0.12	0.11
Mediamill	0.17	0.19	0.20	0.18	0.16
Nus-wide	0.57	0.60	0.67	0.58	0.56

Table 8: Coverage.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	6.22	6.31	6.25	6.27	6.16
Emotion	2.31	2.36	2.33	2.37	2.29
Genbase	0.56	0.53	0.56	0.61	0.59
Scene	0.45	0.47	0.49	0.50	0.44
Enron	15.4	15.1	14.4	14.1	13.9
Medical	2.49	2.45	2.61	2.56	2.64
Core15k	115	112	121	113	122
CAL500	113	117	126	123	126
Mediamill	19.0	20.1	20.2	19.6	18.9
Nus-wide	14.0	13.6	13.9	13.8	13.3

Table 9: Average precision.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	0.78	0.73	0.77	0.83	0.87
Emotion	0.62	0.67	0.65	0.69	0.71
Genbase	0.98	0.98	0.97	0.98	0.99
Scene	0.89	0.86	0.83	0.82	0.86
Enron	0.63	0.59	0.62	0.61	0.65
Medical	0.80	0.89	0.83	0.85	0.87
Core15k	0.23	0.21	0.20	0.24	0.19
CAL500	0.46	0.43	0.45	0.47	0.49
Mediamill	0.69	0.67	0.66	0.69	0.72
Nus-wide	0.46	0.47	0.49	0.51	0.52

Table 10: Ranking loss.

Dataset	ML-kNN	NC	BOOSTEXTER	RANK-SVM	BML-kNN
Yeast	0.18	0.19	0.23	0.21	0.25
Emotion	0.25	0.23	0.22	0.23	0.26
Genbase	0.02	0.01	0.03	0.02	0.02
Scene	0.09	0.14	0.11	0.13	0.07
Enron	0.10	0.12	0.13	0.09	0.12
Medical	0.05	0.1297	0.19	0.19	0.14
Core15k	0.14	0.13	0.12	0.17	0.14
CAL500	0.18	0.18	0.19	0.19	0.17
Mediamill	0.06	0.07	0.09	0.08	0.05
Nus-wide	0.12	0.14	0.13	0.12	0.11

Table 7 reports the experimental results of the classification evaluation One-error. Except the datasets Genbase, Medical, Core15k, and CAL500, BML-kNN works slightly better than the other algorithm. From the experimental results, we can also find that BML-kNN has a prominent performance in all numerical data.

The metric Coverage evaluates the average cost to cover all the true labels. The test results on Coverage are listed in Table 8. The experimental results show that NC and BML-kNN outperform the others. However, the algorithms do not differ much in the numerical value of Coverage. Meanwhile, BML-kNN does not have obvious advantages in the symbol dataset Enron, Medical, Core15k, and Genbase.

Different from other evaluation metrics, the higher the average precision, the better the classification effect. According to the evaluation metrics Average precision and Ranking loss as shown in Tables 9 and 10, BML-kNN is slightly better than NC, and NC is better than the others.

5. Conclusions

The proposed algorithm BML-kNN is based on the framework of ML-kNN. ML-kNN only restricts the number of the nearest neighbors, while NC only limits the distance of the nearest neighbors. BML-kNN considers both the two factors at the same time and gives the estimation of test instances based on up to k training instances in the neighborhood. Experimental results illustrate that the classification accuracies of BML-kNN and NC are slightly higher than that of ML-kNN. The calculation speeds of

BML-kNN and ML-kNN are basically equal, while the algorithm NC has higher computational complexity.

The algorithms involved in this paper divide the multilabel classification problem into a series of single-label problems. Specifically, each label is extracted and combined with the feature data to form a single-label binary classification problem. In this way, the single-label problems are independent of each other, and the correlation between labels is not considered. How to learn the label correlation and apply the correlation into the algorithm may be a meaningful topic for future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by National Natural Science Foundation of China (No. 61976027), Liaoning Provincial Department of Education (No. LJ2019011, LJKZ1026, and LJKZ1030) and Liaoning Natural Fundation Guidance Plan (No. 2019-ZD-0502).

References

[1] M. L. Zhang and Z. H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[2] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[3] S. J. Huang, W. Gao, and Z. H. Zhou, "Fast multi-instance multi-label learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2614–2627, 2019.

[4] Y. Zhu, K. M. Ting, and Z. H. Zhou, "Multi-label learning with emerging new labels," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1901–1914, 2018.

[5] B. Wu, F. Jia, W. Liu, B. Ghanem, and S. Lyu, "Multi-label learning with missing labels using mixed dependency graphs," *International Journal of Computer Vision*, vol. 126, no. 8, pp. 875–896, 2018.

[6] M. L. Zhang, Y. K. Li, X. Y. Liu, and X. Geng, "Binary relevance for multi-label learning: an overview," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.

[7] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[8] M. L. Zhang and Z. H. Zhou, "ML-KNN: a lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[9] R. E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[10] F. D. Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision tree from texts and data," in *Machine Learning and Data Mining in Pattern Recognition. MLDM 2003*, pp. 35–49, Springer, Berlin, Heidelberg, , Lecture Notes in Computer Science, 2003.

[11] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 681–687, Vancouver, British Columbia, Canada, December 2002.

[12] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.

[13] V. Losing, B. Hammer, and H. Wersing, "Tackling heterogeneous concept drift with the Self-Adjusting Memory (SAM)," *Knowledge and Information Systems*, vol. 54, no. 1, pp. 171–201, 2018.

[14] M. Roseberry and A. Cano, "Multi-label kNN classifier with Self adjusting memory for drifting data streams," in *Proceedings of Machine Learning Research*, pp. 23–37, 2018.

[15] S. Zhang, "KNN-CF approach: incorporating certainty factor to kNN classification," *IEEE Intelligent Informatics Bulletin*, vol. 11, no. 1, pp. 24–33, 2010.

[16] S. Zhang, "Shell-neighbor method and its application in missing data imputation," *Applied Intelligence*, vol. 35, no. 1, pp. 123–133, 2011.

[17] H. Liu and S. Zhang, "Noisy data elimination using mutual k-nearest neighbor for classification mining," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1067–1074, 2012.

[18] A. K. Ghosh, "On optimum choice of k in nearest neighbor classification," *Computational Statistics & Data Analysis*, vol. 50, no. 11, pp. 3113–3123, 2006.

[19] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel kNN algorithm with data-driven k parameter computation," *Pattern Recognition Letters*, vol. 109, pp. 44–54, 2018.

[20] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN Classification," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 3, pp. 1–19, 2017.

[21] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018.

[22] S. Zhang, "Cost-sensitive KNN classification," *Neurocomputing*, vol. 391, pp. 234–242, 2020.

[23] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[24] J. Song, J. Zhao, F. Dong, J. Zhao, Z. Qian, and Q. Zhang, "A novel regression modeling method for PMSLM structural design optimization using a distance-weighted KNN algorithm," *IEEE Transactions on Industry Applications*, vol. 54, no. 5, pp. 4198–4206, 2018.

[25] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh et al., "Effects of distance measure choice on K-nearest neighbor classifier performance: a review," *Big Data*, vol. 7, no. 4, pp. 221–248, 2019.

[26] Y. Yi, J. Wang, W. Zhou, Y. Fang, J. Kong, and Y. Lu, "Joint graph optimization and projection learning for dimensionality reduction," *Pattern Recognition*, vol. 92, pp. 258–273, 2019.

[27] S. Zhang, X. Li, Q. Lin, and K. C. Wong, "Nature-inspired compressed sensing for transcriptomic profiling from random composite measurements," *IEEE Transactions on Cybernetics*, vol. 51, no. 9, pp. 4476–4487, 2021.

[28] Y. Yi, J. Wang, W. Zhou, C. Zheng, J. Kong, and S. Qiao, "Non-negative matrix factorization with locality constrained

adaptive graph," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 2, pp. 427–441, 2020.

[29] L. Cui, Y. Zhang, R. Zhang, and Q. H. Liu, "A modified efficient kNN method for antenna optimization and design," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 10, pp. 6858–6866, 2020.

[30] Z. Yu, C. Bian, G. Liu, S. Zhang, K. C. Wong, and X. Li, "Elucidating transcriptomic profiles from single-cell RNA sequencing data using nature-inspired compressed sensing," *Briefings in Bioinformatics*, vol. 22, no. 5, Article ID bbab125, 2021.

[31] Q. H. Hu, D. R. Yu, and Z. X. Xie, "Neighborhood classifiers," *Expert Systems with Applications*, vol. 34, no. 2, pp. 866–876, 2008.