Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2020, Vol. 16(3)
© The Author(s) 2020
DOI: 10.1177/1550147720911892
journals.sagepub.com/home/dsn
SAGE

A novel multi-label classification algorithm based on *K*-nearest neighbor and random walk

Zhen-Wu Wang[1], Si-Kai Wang[1], Ben-Ting Wan[2] and William Wei Song[2,3]

Abstract
The multi-label classification problem occurs in many real-world tasks where an object is naturally associated with multiple labels, that is, concepts. The integration of the random walk approach in the multi-label classification methods attracts many researchers' sight. One challenge of using the random walk-based multi-label classification algorithms is to construct a random walk graph for the multi-label classification algorithms, which may lead to poor classification quality and high algorithm complexity. In this article, we propose a novel multi-label classification algorithm based on the random walk graph and the *K*-nearest neighbor algorithm (named MLRWKNN). This method constructs the vertices set of a random walk graph for the *K*-nearest neighbor training samples of certain test data and the edge set of correlations among labels of the training samples, thus considerably reducing the overhead of time and space. The proposed method improves the similarity measurement by differentiating and integrating the discrete and continuous features, which reflect the relationships between instances more accurately. A label predicted method is devised to reduce the subjectivity of the traditional threshold method. The experimental results with four metrics demonstrate that the proposed method outperforms the seven state-of-the-art multi-label classification algorithms in contrast and makes a significant improvement for multi-label classification.

Keywords
K-nearest neighbor, multi-label classification, random walk, similarity measurement

Date received: 21 May 2019; accepted: 13 February 2020

Handling Editor: Paolo Barsocchi

Introduction

In the data mining field, the traditional binary classification or multi-classification problems have been explored substantially. However, the multi-label classification (MLC) problem still exists and it has recently attracted increasing research sights due to its wide range of applications, such as text classification,[1,2] gene function classification,[3] social network analysis,[4] and image/video annotation.[5] Furthermore, with the rapid increase of development and applications with wireless sensor networks (WSNs), massive data collected from a large number of monitoring objects[6–12] are analyzed, clustered, and classified with methods like classic *K*-nearest neighbor (KNN), support vector machine (SVM) algorithms,[9,10] and MLC methods.[11,12]

[1]Department of Computer Science and Technology, China University of Mining and Technology, Beijing, China
[2]School of Software and Internet of Things Engineering, Jiangxi University of Finance and Economics, Nanchang, China
[3]Department of Information Systems, Dalarna University, Falun, Sweden

Corresponding author:
William Wei Song, Department of Information Systems, Dalarna University, Falun S-791 88, Sweden.
Email: wso@du.se

Following are a few examples to illustrate advanced data analysis approaches which are applied to WSN data. With a wireless sensor network system set in a room to collect limb motion data, Guraliuc et al.[9] use the KNN and SVM algorithms to classify limb movements, aiming to develop a method for patient motion therapy. Constructing a sensor device over a bed to collect the sleeping posture data of human body, Barsocchi[10] use KNN and SVM algorithms to classify the sleeping postures for bedsore therapy. Belmannoubi et al.[11] adopted the MLC method to simplify and reduce the complexity of the classification task in order to improve the accuracy of zone location in a multi-building, multi-floor indoor environment. Zhang et al.[12] applied the MLC method to detect multiple data faults[13] (regarded as multiple labels) simultaneously in sensor networks because it is difficult to build detection model for each fault type.

The traditional single-label classification (SLC) problem considers that one instance belongs only to one category, whereas in the MLC problem, one instance can be allocated to multiple categories simultaneously. Since SLC is merely a special case, MLC deals with a more difficult and general problem in the data mining domain, focusing on the following two challenges. (1) The number of label sets may be very large for test instances (e.g. being exponentially proportional to the total number of labels).[14] For instance, 10 labels would lead to 2^{10} possible combinations of label sets for each test instance. (2) Due to multiple labels and possible links among them, their correlations become very complex.[15] For instance, on one hand, it is more likely for a piece of news tagged with "entertainment" to have another tag "sport" than "war." On the other hand, in a classification of natural scenes with the set of picture labels {"beach," "field," "autumn leaf," "sunset," "mountain," "city"}, it is less possible that a scenery picture is labeled by both "autumn leaf" and "beach."

Thus, new approaches have been introduced to the MLC algorithms in recent years. For example, considering the application of graph representation to the MLC methods to cope with the above-mentioned problems. However, little progress has been made with these methods and algorithms[15–29] especially in the following aspects. (1) *Complexity*: in order to construct a graph model, a graph-based MLC method must map the instances of an entire training set to graph vertices, with many instances irrelevant to the test instances causing very high requirements for time and space and causing high computational complexity. (2) *Heterogeneity*: for similarity measurement among instances, nearly none of these methods consider the difference between discrete features and continuous features. A measurement method for continuous features is not always applicable to discrete features, such as non-consecutive values (e.g. gender, occupation). Suppose that occupations are expressed by positive integers $\{1, 2, \ldots, n\}$, and the corresponding values of instances x_1, x_2, and x_3 are 1, 2, and 5, respectively. Theoretically, the similarities between them should be the same, but it is more often than not to consider applying a method for computing continuous similarity such as Euclidean distance,[30] which will cause that the distance between the values of x_1 and x_3 is less (hence more similar) than that of the values of x_1 and x_2. (3) *Uncertainty*: when calculating the label sets for the test instances, most of the above-mentioned methods use a probability threshold to determine the prediction labels. The subjectivity of selection of the probability threshold unavoidably leads to an improper setting and overfitting for the label sets.[17,23]

To overcome these problems, we propose a novel graph-based MLC algorithm, which adopts KNN and random walk algorithms, named multi-label classification based on the random walk graph and the K-nearest neighbor algorithm (MLRWKNN). Here, the random walk algorithm is used to explore the interdependent relationships among the labels through the connectivity between vertices on a graph model. In order to construct a random walk graph, the MLRWKNN algorithm creates a vertex set for certain test instances containing only its KNN training instances, not necessarily the entire training set, and an edge set by adopting the correlations among label sets of vertices samples. For the similarity measurement, the MLRWKNN algorithm attempts to differentiate and integrate the discrete and continuous features of the datasets and improves the similarity computation for the features. To deal with the subjectivity problem of label prediction, the MLRWKNN algorithm estimates the label numbers by computing the probability of the test instances belonging to each label, and then ranks the labels in the label set in a descending order according to their prediction probabilities. Furthermore, considering the possible effects of parameter selection on the classification algorithm performance, the MLRWKNN algorithm also discusses the selection principles and the recommended values for K, jump probability α, and adjustment factor σ. The main contributions of this article include the following:

1. A new construction method for graph model is proposed to greatly reduce the time and space complexity of random walk algorithm, and hence to be able to handle large-scale data more easily than traditional methods.
2. The similarity measurement method is improved to express the relationships between the instances more accurately through differentiating and integrating discrete and continuous features.
3. A new prediction method is devised for the label set to reduce the subjectivity of the probability

threshold method which occurs in most graph-based MLC algorithms.

4. We propose the recommended values of algorithm parameters through experimental analysis instead of subjective empirical constants, which possess a great significance in applying this method to similar problems in different domains.

The rest of this article is organized as follows. The background and reviews of the related work about random walk strategy, the KNN algorithm, and graph-based MLC algorithms are discussed in section "Related work." Based on the previous research work, we propose our approach MLRWKNN in section "The principle of the MLRWKNN algorithm," which consists of three components: design of feature similarity computation, construction of random walk graph, and label set prediction. In section "Experimental Results and Analysis," we discuss experiment process, the test data used, and evaluation criteria. We also present comprehensive experimental results and their analysis. We conclude the article in section "Conclusion," indicating our contributions to this research area and our future work in this direction.

Related work

In this section, we first introduce the random walk strategy and KNN algorithm, which will build up a theoretical foundation for our proposed approach, and then we review the graph-based MLC algorithms.

Random walk and the KNN algorithms

Random walk is an algorithm based on graph representation that iteratively explores the global structure of a network to estimate the proximity between two nodes. As mentioned in the previous section, one challenge of the MLC problem is that there are complex relationships among multiple labels. One solution to this problem is to apply the random walk algorithm to accurately describe the correlations among labels using the connectivity between vertices on a random walk graph. The four input parameters in a random walk algorithm are an adjacent matrix $\boldsymbol{P} \in R^{r \times r}$ of the state transition probability (R is the real number set and r is a natural number), an initial probability distribution vector $\boldsymbol{p} \in R^r$, a jump probability α, and a jump-occurrence probability distribution vector $\boldsymbol{u} \in R^r$; the iterative mode of random walk can be recursively described as $\boldsymbol{p}^{k+1} = (1-\alpha)\boldsymbol{P} \times \boldsymbol{p}^k + \alpha \boldsymbol{u}$ $(k = 0, 1, 2, \ldots)$. Generally speaking, each element $\boldsymbol{P}(i,j)$ in $\boldsymbol{P}$, $0 < i \leq r$, $0 < j \leq r$, is represented by the similarity between vertices i and j, and $\boldsymbol{P}(i,j)$ indicates that the walk probability is higher for any vertex i if it is more similar to its

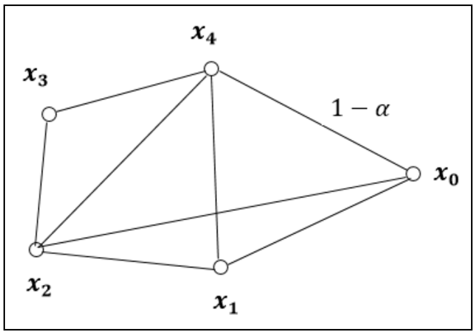

Figure 1. The random walk diagram.

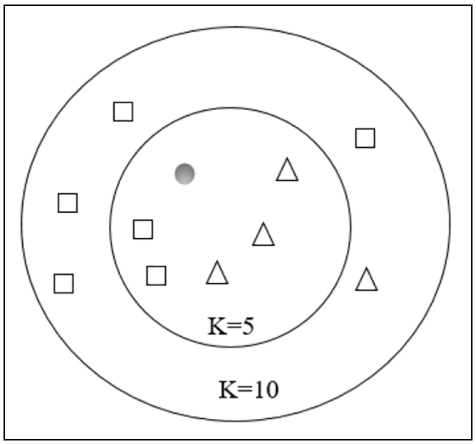

Figure 2. The KNN diagram.

neighbor vertex j. The algorithm assumes that an initial walk would jump to any vertex at the same probability, which indicates that each element in $\boldsymbol{u}$ has the same value (e.g. $1/r$). In addition, the kth element $\boldsymbol{p}(k)$ in the initial vector $\boldsymbol{p}$ is described by the similarity between the start vertex (e.g. x_0) and the vertex k. As described in Figure 1, the basic idea behind the random walk is that a walker traverses a graph from one vertex to a series of vertices, and at any vertex, the walker will walk to its neighbor vertex with the probability $1-\alpha$ and teleport to any other vertex in the graph with the probability α.[17] A probability distribution vector $\boldsymbol{p}^k$ is obtained after the kth walk and the random walk algorithm adopts $\boldsymbol{p}^k$ as the input to the $(k+1)$th walk and performs the iterations over the graph until it reaches the maximum iteration number or $\boldsymbol{p}$ converges.[17]

The KNN algorithm is a lazy learning method which classifies samples according to the idea of "birds of a feather flock together." For certain test instances, it acquires the KNN instances taken from a training set according to a certain similarity measure and votes on the labels of the KNN instances to determine the predicted label for a test instance. As shown in Figure 2, the test instance "●" would be classified as "Δ" when

$K = 5$ according to the voting mechanism. However, it would be classified as "□" when $K = 10$. Obviously, the key problem for the KNN algorithm is how to choose the similarity measurement and K value.

PTMs and AAMs

In general, the MLC algorithms can be classified into two categories: problem transformation methods (PTMs) and algorithm adaptation methods (AAMs). The classical PTMs aim to transform an MLC problem into one or several SLC problems to which existing solutions are available. Here, we brief a few of typical ones. Binary relevance (BR)[4] transforms the MLC problem into a binary classification and generates a separate dataset for each label. Calibrated label ranking (CLR)[31] considers the MLC as a label ranking problem and learns a mapping from instances to rankings over a predefined set of labels. Hierarchy of multi-label classifiers (HOMER)[32] transforms the MLC problem into a tree hierarchy of simpler MLC tasks, in which for each node, labels are split using a balanced clustering algorithm and grouped similar labels into a meta-label, and meta-labels would be predicted by a classifier. Label powerset (LP)[33] attempts to build a single-labeled system, called independent label, for a dataset from possible combinations of labels. RAndom *k*-labELsets (RAkEL)[34] converts the MLC problem into a multiple classification problem, by ranking the votes of sub-classifiers and taking the most relevant labels as the prediction results with a threshold.

The traditional AAMs intend to enhance the SLC algorithms so that they can handle the MLC problem. For example, the well-known multi-label *K*-nearest neighbor (MLKNN)[35] extends the KNN algorithm using the maximum a posteriori (MAP) principle to determine the label set for the unseen instances. Using the maximum margin strategy to deal with multi-label data, the classic Rank-SVM[36] optimizes a set of linear classifiers to minimize the empirical ranking loss and can hence handle nonlinear cases with kernel tricks.

Recently, researchers have been focusing on the introduction of graph representation to the MLC algorithms, which can produce more accurate results with probability-based principles, and elegant representation capability of detecting label correlations. The graph-based MLC methods can be put into two categories, one category focusing on the improvement of the existing MLC algorithms by building corresponding graph models for multi-label datasets, and the other category focusing on the solutions to the MLC problem by combining the SLC algorithms with a graph model. Included in the first category are the improved BR algorithms,[15] the improved classification chain (CC) algorithms,[27,28] and the improved MLKNN algorithm.[29] In the following, we describe each of them succinctly. In Cetiner and Akgul,[15] the label independence issue of the BR algorithm is addressed by first assuming the outputs of each binary classifier as observed nodes of a graphical model, and then determining the final label assignments using the standard powerful Bayesian inference for the unobservable nodes. The Neighbor Pair Correlation Chain Classifier (NPC) algorithm[27] constructs a graph of labels based on the latent Dirichlet allocation (LDA) model and acquires the label correlations using the random walk with the restart strategy. Then, the CC algorithm will solve the label chain selection problem by determining the label correlations to establish the best label chain. In Lee et al.,[28] a directed acyclic graph (DAG) is constructed to maximize the sum of conditional entropies between all parents and children nodes, the highly correlated labels are sequentially ordered in chains obtained from the DAG, and the predictive power could be maximized by utilizing a CC approach with these chains. The instance-ranking method (IR)[29] maps all the training and test instances into a graph, assigns weights to each training instance with the random walk, and uses these weights to calculate the label prior probability in the MLKNN algorithm.

The following algorithms belong to the second category: the random walk-based MLC algorithms,[16,17,19,20,22,23] the MLC algorithm based on Hilbert–Schmidt independence criterion,[24] and the dictionary learning-based DL-MLC algorithm.[25] Specifically, the graph DL-MLC algorithm[25] maps a training set to a graph model and improves the label-consistent K-SVD algorithm (LC-KSVD)[37] with adopting the graph Laplacian regularization. The genome-scale metabolic model (GSMM) method[24] contains three steps: first, it converts the training and test sets to a weighted undirected graph and describes the graph smoothness through a series of transformations including the adjacency matrix, the real label set of training instances, and the predicted label set of test instances; second, the algorithm describes the consistence of label space with the Hilbert–Schmidt independent criterion; third, it builds the MLC classifier through optimizing the smoothness and consistency. Among the MLC algorithms based on random walk, these three algorithms MLRW,[17] ML-RWR,[23] and RW.KNN[19] map all training instances to graph vertex sets, acquire the probability distribution from test instances to training set with random walk, and compute the predicted probability of each label by the probability distribution and label sets of training instances. Even though the above three algorithms all construct the edge sets on the whole training instances, MLRW connects the edges among instances that have the same labels, while ML-RWR connects the edges among instances which have mutual KNN relations, and still RW.KNN considers only unilateral KNN relations. The transductive multi-label learning (TML)[16] method builds a complex partial directed graph by mapping all the training and test instances to graph vertices. Undirected edges link the test instances that have KNN relations and directed edges link the training instances. Based on this graph, TML improves the method proposed in Azran[38] to tackle the MLC problem. Wang et al.[20] constructed a bi-relational graph via combining a data graph and a label graph. The data graph is constructed by all the training and test instances and the random walk with restart strategy is used to compute transition probability from each label vertex to the instance vertex, that is, the label predicted probability.

In summary, for the graph construction problems all the afore-mentioned algorithms use whole training instances (even all the training and test instances) to construct graph vertices, which leads to the embedding of too many vertices unrelated to test instances. Huge graph vertices require considerable computation power in time and space, and sometimes even deteriorate the classification effect.[17] For the similarity measurement problem, the above algorithms adopt the distance reciprocal between feature vectors,[17–19,29] the distance-based Gaussian function method,[16,20,23,24] and the sparse rule operator method.[22] None of these methods consider the difference between discrete features and continuous features. For the parameter selection problem, almost all the algorithms adopt empirical constants and do not implement in-depth experimental analysis on how to choose algorithm parameters. In order to overcome these shortcomings, we propose a novel MLC algorithm based on random walk strategy and KNN algorithm. Section "The principle of the MLRWKNN algorithm" provides a detailed description of the proposed method.

The principle of the MLRWKNN algorithm

In order to outline the MLRWKNN algorithm proposed in this article, some basic conceptions of MLC are discussed here. Suppose that (x, y) represent a multi-label sample where x is an instance and $y \subseteq L$ is its corresponding label set. The total label set L is defined as follows

$$L = \{l_1, l_2, \ldots, l_Q\}, \text{Q is the total number of labels} \quad (1)$$

Suppose that $\boldsymbol{x} = (x^1, x^2, \ldots, x^D) \in X$ is a D-dimensional feature vector corresponding to x, where $X \subseteq R^D$ is the feature vector space and x^d, $d = 1, 2, \ldots, D$, denotes a specific feature, and $\boldsymbol{y} = (y^1, y^2, \cdots, y^Q) \in \{0, 1\}^Q$ is the Q-dimensional label vector corresponding to y, and y^q is described as

$$y^q = \begin{cases} 0, l_q \notin y \\ 1, l_q \in y \end{cases}, \; q = 1, 2, \ldots, \text{Q} \quad (2)$$

Therefore, the multi-label classifier h can be defined as

$$h : X \rightarrow \{0, 1\}^Q \quad (3)$$

Suppose there are m samples in training set X_{train} and n samples in test set X_{test}, they are defined as follows. Let $\boldsymbol{x}_0 \in X_{test}$ denote a certain test instance

$$X_{train} = \left\{(\boldsymbol{x}_i, \boldsymbol{y}_i) | \boldsymbol{x}_i \in X \cap \boldsymbol{y}_i \in \{0, 1\}^Q, i = 1, 2, \ldots, m\right\},$$
$$X_{test} = \left\{(\boldsymbol{x}_i, \boldsymbol{y}_i) | \boldsymbol{x}_i \in X \cap \boldsymbol{y}_i \in \{0, 1\}^Q, i = m+1, m+2, \ldots, m+n\right\} \quad (4)$$

Overall description of the MLRWKNN algorithm

The MLRWKNN algorithm aims to solve the MLC problem through constructing the random walk graphs, where the vertex and edge sets are generated by a certain test instance (e.g. $\boldsymbol{x}_0$) and its KNN instances (named $N^K_{x_0}$) in X_{train}. In detail, the MLRWKNN algorithm consists of the following three steps:

1. For each test instance in X_{test}, the MLRWKNN algorithm constructs the random walk graphs for each label. For example, the graph of $\boldsymbol{x}_0$ on l_q can be constructed as follows: the MLRWKNN algorithm maps $\boldsymbol{x}_0$ and its KNN instances ($K = 4$ in Figure 3) in X_{train} to a vertex set; the instances in KNN would be connected by undirected edges if they have the same labels, and $\boldsymbol{x}_0$ would be connected by undirected edges with the instances which have the label l_q. Finally, Q graphs (named $G^q_{x_0}, q = 1, 2, \ldots, Q$) are constructed for $\boldsymbol{x}_0$, as described in Figure 3. The detailed information of the graph construction is described in section "A new construction method of the random walk graph" (equations 8–12).
2. Through random walk operation, the MLRWKNN algorithm computes the probability distribution on the graph vertices. As shown in Figure 3, starting from $\boldsymbol{x}_0$ on all the above graphs, a traveler will walk to its neighbor vertex at the probability $1 - \alpha$ and teleport to any other vertex at the probability α, and insofar Q stable probability distribution vectors are obtained. Through computing the prior probability for each label as the weight to each Q vector, the MLRWKNN algorithm produces a summation of the weighted Q vectors, which is

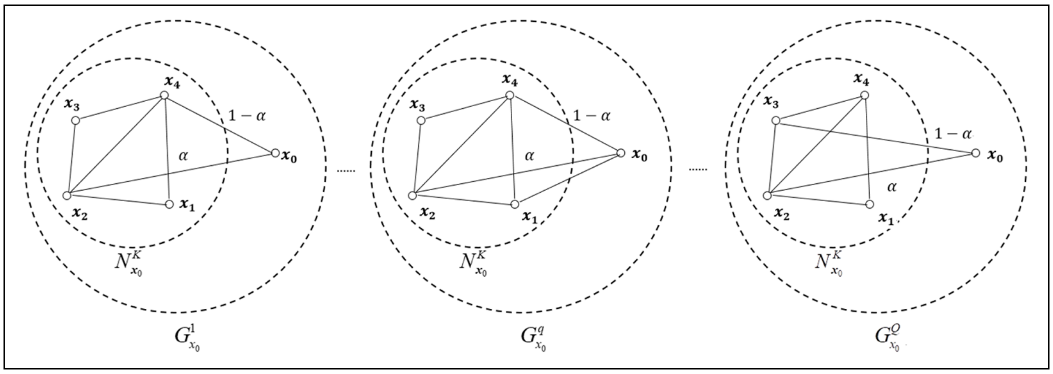

Figure 3. The construction of random walk graphs.

considered the final probability distribution vector. Equations 13–18 in section "A new construction method of the random walk graph" describe the procedure of random walk and the generation of probability distribution vector.

3. Through using the above obtained probability distribution vector, the MLRWKNN algorithm predicts the probability of each label belonging to $\boldsymbol{x}_0$ and sorts all these probabilities in a descending order, and then the number of the labels of $\boldsymbol{x}_0$ is calculated, and finally the predicted labels are generated by choosing higher probability labels. Section "Label Sets Prediction" gives detailed information about the prediction of label set.

In order to elaborate the MLRWKNN algorithm at a great detail, we will discuss the design of similarity measurement, the construction of random walk graph, and the prediction of label set in the following sections. Similarity measurement, as the basis of KNN algorithm, is discussed in section "Design of proposed similarity," and sections "A new construction method of the random walk graph" and "Label sets prediction" give the detailed description of the above three steps.

Design of proposed similarity

Considering the difference between discrete features and continuous features, our MLRWKNN calculates the similarities for discrete features and continuous features, respectively, and then combines them with a linear weighted process. Specifically, given an instance $\boldsymbol{x} = (x^1, x^2, \ldots, x^D) \in \chi_{train}$, suppose that there are D_1 discrete features and $D - D_1$ continuous features. For two instances, $\boldsymbol{x_i} = (x_i^1, x_i^2, \ldots, x_i^{D_1+1}, x_i^{D_1+2}, \ldots, x_i^D)$ and $\boldsymbol{x_j} = (x_j^1, x_j^2, \ldots, x_j^{D_1+1}, x_j^{D_1+2}, \ldots, x_j^D)$, their similarity based on discrete features is defined as follows

$$NomSim(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{\left|\left\{\left(x_i^k, x_j^k\right) | x_i^k = x_j^k, k = 1, 2, \ldots, D_1\right\}\right|}{D_1} \tag{5}$$

In order to determine the value range of similarity based on a continuous feature, MLRWKNN adopts the Gaussian kernel function to handle the similarity of continuous features

$$NumSim(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp\left[-\frac{\left(dist\left((x_i^{D_1+1}, x_i^{D_1+2}, \ldots, x_i^D), \left(x_j^{D_1+1}, x_j^{D_1+2}, \ldots, x_j^D\right)\right)\right)^2}{2\sigma^2}\right] \tag{6}$$

where $dist((x_i^{D_1+1}, x_i^{D_1+2}, \ldots, x_i^D), (x_j^{D_1+1}, x_j^{D_1+2}, \ldots, x_j^D))$ represents the similarity of continuous features (such as Euclidean distance), and σ is the spread factor in the Gaussian kernel function.[39] Refer to Section "Experimental analysis of the selection of the σ value" for a detailed definition and explanation of σ. The final similarity between x_i and x_j is defined as

$$Similarity(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{D_1}{D} NomSim(\boldsymbol{x_i}, \boldsymbol{x_j}) + \frac{D - D_1}{D} NumSim(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{7}$$

A new construction method of the random walk graph

In general, only a few training instances play a decisive role in predicting the label sets of test instances, whereas the other training instances not only complicate the random walk graph and interfere with the classification results. In this article, we propose a novel construction method of the random walk graph by adopting KNN instances of $\boldsymbol{x}_0$ in X_{train}, and the KNN instances $N^K_{x_0}$ is defined as follows

$$N^K_{x_0} = \{\boldsymbol{x} | \boldsymbol{x} \text{ belongs to the } K \text{nearest neighbors of } \boldsymbol{x}_0 \text{ in } X_{train}\} \tag{8}$$

Let $G^q_{x_0}$ represent the KNN graph of $\boldsymbol{x}_0$, based on l_q. $G^q_{x_0}$ is defined as follows

$$G^q_{x_0} = \langle V^q_{x_0}, E^q_{x_0} \rangle, \text{such that} \tag{9}$$

$$V^q_{x_0} = V_{x_0} \cup \{\boldsymbol{v}_0 | \boldsymbol{v}_0 = \boldsymbol{x}_0\} \text{ and}$$
$$E^q_{x_0} = E_{x_0} \cup \left\{ e_{0i} | \exists \boldsymbol{x} \in N^K_{x_0}, s.t.\, l_q \in y \right\} \tag{10}$$

where

1. The vertex set of instances V_{x_0} is defined as

$$V_{x_0} = \left\{ v_i | \exists \boldsymbol{x}_i^{'} \in N^K_{x_0}, s.t. \boldsymbol{v}_i = \boldsymbol{x}_i^{'}, i = 1, 2, \ldots, K \right\} \tag{11}$$

2. $\boldsymbol{v}_0$ represents the vertex of $\boldsymbol{x}_0$;
3. The edge set $\{e_{0i} | \exists \boldsymbol{x} \in N^K_{x_0}, s.t. l_q \in y\}$ links $\boldsymbol{v}_0$ with other vertices in V_{x_0};
4. E_{x_0}, the edge set among vertices in V_{x_0}, is defined as

$$E_{x_0} = \left\{ e_{ij} | \exists \boldsymbol{x}_i', \boldsymbol{x}_j', s.t. \boldsymbol{v}_i = \boldsymbol{x}_i', \boldsymbol{v}_j = \boldsymbol{x}_j' \textit{ and } y_i \cap y_j \neq \varnothing \right\} \tag{12}$$

Since there is no need to construct the KNN for every training instance in X_{train}, MLRWKNN avoids many *sort* operations (the KNN for each training instance needs to sort m training instances). Based on $G^q_{x_0}$, the iterative mode of random walks is described as

$$\boldsymbol{p}_q^{k+1} = (1-\alpha) \boldsymbol{P}^q_{x_0} \times \boldsymbol{p}_q^k + \alpha \boldsymbol{u} \tag{13}$$

The terms in this formula are explained as follows: (1) $\boldsymbol{p}_q^k \in R^{K+1}$ $(k = 0, 1, 2, \ldots)$ is the probability distribution vector after k times of random walks, whose ith element $p_q^k(i)$ represents the probability of the $(i-1)$th vertex of $G^q_{x_0}$, and (2) $\boldsymbol{P}^q_{x_0} \in R^{(K+1)\times(K+1)}$ is the adjacent matrix of the state transition probability of $G^q_{x_0}$, whose element $\boldsymbol{P}^q_{x_0}(i,j)$, the probability of a random walk from $\boldsymbol{v}_i$ to $\boldsymbol{v}_j$, is defined as follows

$$\boldsymbol{P}^q_{x_0}(i,j) = \frac{similarity(v_i, v_j)}{\sum_{k=0}^{K} similarity(v_i, v_k)} \delta\left(e_{i,j} \in E^q_{x_0}\right),$$
$$where\, \delta\left(e_{i,j} \in E^q_{x_0}\right) = \begin{cases} 0, e_{i,j} \notin E^q_{x_0} \\ 1, e_{i,j} \in E^q_{x_0} \end{cases} \tag{14}$$

In equation 13, $\alpha \in [0, 1]$ is a constant and $\boldsymbol{u} \in R^{K+1}$ is the jump probability vector. Assuming a walk jumps from a vertex to other vertices at the same probability $(1/(K+1))$ when starting from any arbitrary vertices, we define $\boldsymbol{u}$ as follows

$$\boldsymbol{u} = \frac{\mathbf{I}_{K+1}}{K+1} \tag{15}$$

where I_{K+1} is the $(K+1)$-dimension constant vector, each of whose elements has the same value 1. Note that for equation 13, the convergent probability distribution vector $\boldsymbol{p}_q^*$ must satisfy

$$\boldsymbol{p}_q^* = (1-\alpha) \boldsymbol{P}^q_{x_0} \times \boldsymbol{p}_q^* + \alpha \boldsymbol{u} \tag{16}$$

When the random walk procedure ends, the MLRWKNN algorithm generates Q stable probability distribution vectors and the final probability distribution vector is defined as

$$\boldsymbol{p}^* = \sum_{q=1}^{Q} \boldsymbol{p}_L(q) * \boldsymbol{p}_q^* \tag{17}$$

where $\boldsymbol{p}_L \in R^Q$, whose qth element $\boldsymbol{p}_L(q)$ represents the prior probability of the qth label, is defined as

$$\boldsymbol{p}_L(q) = \frac{\left|\left\{ \boldsymbol{x} \in X_{train} | l_q \in y \right\}\right|}{m} \tag{18}$$

Note that the symbol (*) is an ordinary multiplication operator. In summary, for a given random walk graph, the MLRWKNN algorithm obtains the probability distribution from $\boldsymbol{v}_i$ to other vertices by one random walk operation, and then uses the above probabilities as the starting probabilities for each vertex. The algorithm repeats the above process to obtain new probabilities until it reaches a given number of walk rounds or the probability distribution which remains unchanged. The last probabilities are regarded as the final walk probabilities from $\boldsymbol{v}_i$ to other vertices.

Label sets prediction

The MLRWKNN algorithm predicts the cardinal number $len(y_0')$ of a label set for a test instance $\boldsymbol{x}_0$ with the probabilities that $\boldsymbol{x}_0$ belongs to each label, sorts these probabilities in a descending order, and selects the first $len(y_0')$ labels to form the predicted label set of $\boldsymbol{x}_0$. Suppose $\boldsymbol{p}_{x_0} \in R^Q$ is the predicted probability vector for which $\boldsymbol{x}_0$ belongs to each label. The qth element $\boldsymbol{p}_{x_0}(q)$, representing the probability that $\boldsymbol{x}_0$ belongs to l_q, is defined as

$$\boldsymbol{p}_{x_0}(q) = \sum_{k=1}^{K} \boldsymbol{p}^*(k) * \delta\left(l_q \in y_k'\right) \tag{19}$$

where y_k' is the label sets of the training instances corresponding to $\boldsymbol{v}_k$ and $\delta(l_q \in y_k') = \begin{cases} 0, l_q \notin y_k' \\ 1, l_q \in y_k' \end{cases}$. The predicted label set is defined as follows

$$y_0' = \left\{ l_q \in L | rank_{p_{x_0}}(q) \leqslant len\left(y_0'\right) \right\} \tag{20}$$

Algorithm 1. The steps of the MLRWKNN algorithm

```
Input: $X_{train}$, $\boldsymbol{x}_0$, $K$, $\alpha$, $\sigma$
Output: $p^*$
Steps:
1. Compute the similarities of instances in $X_{train}$ and the similarities between $\boldsymbol{x}_0$ and instances in $X_{train}$ using equation 7;
2. Construct $N_{x_0}^K$ using equation 8;
3. Based on equation 9, construct $G_{x_0}^q$, $q = 1, 2, \ldots, Q$;
4. Execute random walks on $G_{x_0}^q$ based on equation 13 until the algorithm satisfies equation 16 and obtains $p_q^*$, $q = 1, 2, \ldots, Q$;
5. Compute $p_L$ according to equation 18;
6. Compute $p^*$ using equation 17;
7. Compute $p_{x_0}$ based on equation 19;
8. Compute $length(y_0')$ using equation 21;
9. Predict $y_0'$ according to equation 20.
```

where $rank_{p_{x_0}}(q)$ is the descending order of $\boldsymbol{p}_{x_0}(q)$ of $\boldsymbol{p}_{x_0}$ and $len(y_0')$ is defined as follows

$$len(y_0') = \sum_{q=1}^{Q} \boldsymbol{p}_{x_0}(q) \sum_{i=1}^{m} \frac{|y_i|\delta(l_q \in y_i)}{|\{(x, y) \in X_{train} | l_q \in y\}|} \quad (21)$$

where r represents the largest integer that is not greater than the real number r.

In summary, first, the MLRWKNN algorithm computes $N_{x_0}^K$ according to the proposed similarity measure and constructs $G_{x_0}^q$ for the instance x_0. Then, random walks are executed on $G_{x_0}^q$ until a convergence probability distribution is obtained. Finally, the label set is predicted by calculating the length of the label set. The MLRWKNN algorithm is shown in Algorithm 1.

Convergence proof of MLRWKNN algorithm

We maintain this statement that a simple random walk on a graph is a discrete-time Markov chain over the nodes (i.e. vertexes).[40]

Theorem 1. The MLRWKNN algorithm is convergent.

Proof:

1. Because vector $\boldsymbol{u}$ does not contain zero elements and $0 < \alpha < 1$, it is possible that the MLRWKNN algorithm can randomly move to any vertex in $G_{x_0}^q$, starting from any vertex. Therefore, the adjacent matrix $\boldsymbol{P}_{x_0}^q$ is irreducible.
2. A walk from any vertex can in principle return to any given vertex, including the vertex itself, in consecutive steps due to the existence of strictly positive vector $\boldsymbol{u}$. Therefore, the whole random walk procedure is aperiodic.
3. Obviously, it is possible to traverse a vertex again in a positively recurrent number of walk steps after it is traversed for the first time since being a Markov chain; random walks possess the feature of finiteness.
4. From the above three points, it can be observed that the MLRWKNN algorithm is ergodic, and hence convergent.[17] That is, there exists a vector $\boldsymbol{p}_q^*$ which satisfies equation 16.

Experimental results and analysis

In this section, we first introduce experimental environment and datasets; then discuss the experimental analysis of the construction of the random walk graph, the similarity measure method, and the parameter selection of the MLRWKNN algorithm; and finally make a comprehensive experimental comparison of the seven algorithms.

Experiment environment and datasets

Six datasets, which are extensively used by researchers to conduct experiments and evaluate the MLC algorithms, include Flags, Genbase, Medical, Scene, Yeast, and Mediamill (for detailed information about these public datasets, refer to the URL: http://mulan.sourceforge.net/datasets-mlc.html). They cover different application domains such as texts, images, biologic data, and video data, and the number of labels varies from 6 to 101 as described in Table 1. Seven state-of-the-art MLC algorithms were chosen for performance comparison, including BRKNN, CLR, HOMER, LP, RAkEL, MLKNN, and Rank-SVM, among which binary relevance KNN (BRKNN), CLR, HOMER, LP, and RAkEL belong to the category of PTMs, and MLKNN and Rank-SVM to that of AAMs. For the objectivity of comparisons, 10-fold cross validation was adopted and the average values of 10 experimental

Table 1. Description multi-label datasets.

Datasets	Instance number	Label number	Continuous feature number	Discrete feature number	Density	Field
Flags	194	7	10	9	0.485	Image
Genbase	662	27	0	1186	0.046	Biology
Medical	978	45	0	1449	0.028	Text
Scene	2407	6	294	0	0.179	Image
Yeast	2417	14	103	0	0.303	Biology
Mediamill	5000[a]	101	120	0	0.043	Video

[a]Mediamill contains 43,907 items of video data. We used the first 5000 video items in our experiments for comparison purpose.

repetitions were considered as the final values for every evaluation criterion.

Evaluation criteria

So far, a series of criteria have been introduced to evaluate the MLC algorithms from different perspectives, and these measurement criteria can be divided into two categories: bipartition-based criteria and ranking-based criteria.[14,41] The former concentrates on whether a label is correctly predicted, while the latter on whether a relevant label is ranked before an irrelevant label. In general, no algorithm can produce a best performance for all the criteria. Appropriate criteria should be set up in accordance with the optimal objective for a proposed method. As our MLRWKNN algorithm focuses on the ranking problem, four ranking-based criteria have been determined to validate our method: Ranking Loss (RL), One Error (OE), Coverage (Cove), and Average Precision (AP). Let f denote the function of a predicted probability and y' represent the predicted label set of a test instance x_0. The predicted probabilities for which x_0 belongs to each label are sorted in a descending order and $rank_f(x_i, l)$ represents the corresponding ranking of the label l. Let $\overline{y_i}$ be the complement of y_i in L. RL computes the average number of times when irrelevant labels are ranked before the relevant labels

$$RL(h) = \frac{1}{n}\sum_{i=m+1}^{m+n}\frac{1}{|y_i||\overline{y_i}|}|\{(l,l')|l \in y_i \cap l' \in \overline{y_i}, f(x_i,l) \leq f(x_i,l')\}| \tag{22}$$

OE calculates the average number of times that the top-ranked label is irrelevant to the test instance

$$OE(h) = \frac{1}{n}\sum_{i=m+1}^{m+n}\left\{l \notin y_i | l = \arg\max_{l \in L} rank_f(x_i,l)\right\} \tag{23}$$

Coverage $Cove$ reckons the average number of steps that in the ranked list to find all the relevant labels of the test instance

$$Cove(h) = \frac{1}{n}\sum_{i=m+1}^{m+n}\max_{l \in y_i} rank_f(x_i,l) - 1 \tag{24}$$

AP evaluates the degree that labels before the relevant labels are still relevant labels

$$AP(h) = \frac{1}{n}\sum_{i=m+1}^{m+n}\frac{1}{|y_i|}\sum_{l \in y_i}\frac{|\{l' \in y_i | f(x_i,l) \leq f(x_i,l')\}|}{rank_f(x_i,l)} \tag{25}$$

Results and analysis

Analysis on the proposed similarity measurement. Most of the graph-based MLC algorithms use the similarity measurement methods for the continuity features (e.g. Euclidean distance) to compute instance distances, which may be unsuitable for the discrete features. Based on the MLRWKNN algorithm, the proposed similarity is compared with the Gaussian kernel of the Euclidean distance method[16,20,23,24] and the reciprocal of Euclidean distance.[17–19,29] Flags and Genbase datasets were chosen for a comparison study. As described in Table 1, the Genbase dataset only has the feature of discreteness whereas the Flags dataset has both the features of discreteness and continuity. Table 2 shows the performance of the MLRWKNN algorithm under different similarities measurements. The experimental results show that the proposed similarity measurement metric has apparent advantages over the other two measurements in terms of the four criteria.

In terms of the reciprocal of the Euclidean distance, RL, OE, Coverage, and AP on the Genbase dataset were improved by 98.24%, 97.83%, 66.81%, and 14.99%, respectively. The improvement of those of Flags were by percentage 14.56%, 3.39%, 5.73%, and 2.86%, respectively. For the Gaussian kernel of the Euclidean distance, the corresponding performance on

Table 2. Performance comparison under different similarity measurement.

Datasets	Similarity measure	Ranking Loss	One Error	Coverage	Average Precision
Flags	Gaussian kernel of Euclidean distance	0.2212	0.2018	3.7737	0.8123
	Reciprocal of Euclidean distance	0.2157	0.1974	3.7632	0.8158
	The proposed similarity	0.1843	0.1907	3.5474	0.8391
Genbase	Gaussian kernel of Euclidean distance	0.0028	0.0281	0.4307	0.9794
	Reciprocal of Euclidean distance	0.0170	0.2075	1.0909	0.8625
	The proposed similarity	0.0003	0.0045	0.3621	0.9918

Table 3. Performance improvement based on the proposed similarity.

Datasets	Compared similarities	Ranking Loss	One Error	Coverage	AveragePrecision
Flags	Gaussian kernel of Euclidean distance	16.68%	5.5%	6%	3.3%
	Reciprocal of Euclidean distance	14.56%	3.39%	5.73%	2.86%
Genbase	Gaussian kernel of Euclidean distance	89.29%	83.99%	15.93%	1.27%
	Reciprocal of Euclidean distance	98.24%	97.83%	66.81%	14.99%

Table 4. Different graph construction modes comparison for MLRWKNN and ML-RWR.

Datasets	Algorithms	Ranking Loss	One Error	Coverage	Average Precision
Medical	ML-RWR	0.0346	0.3070	2.1659	0.7775
	MLRWKNN	0.01073	0.2577	4.6289	0.7636
Yeast	ML-RWR	0.1661	0.2279	6.2650	0.7673
	MLRWKNN	0.1618	0.2216	6.2087	0.7685

MLRWKNN: multi-label classification based on the random walk graph and the *K*-nearest neighbor algorithm; ML-RWR: multi-label classification based on random walk with restart model.

Genbase were 89.29%, 83.99%, 15.93%, and 1.27% improved, respectively; those of Flags were 16.68%, 5.5%, 6%, and 3.3% improved, respectively. The results in Table 3 show that the proposed similarity can reflect the distances between instances more accurately and it can improve the classification affects greatly.

Analysis of the proposed random walk graph. In order to illustrate the advantages of the MLRWKNN algorithm, we choose the ML-RWR[23] algorithm which improves the MLRW[17] algorithm and adopts the KNN idea. We use the same datasets as ML-RWR, that is, the Medical and Yeast datasets, for the performance comparison, and the experimental results are described in Table 4.

As shown in Table 4, MLRWKNN produced almost the same classification performance as the ML-RWR algorithm. However, compared with the proposed MLRWKNN algorithm, ML-RWR constructs random walk graphs on the whole training set, which leads to the large space and time requirements. Assume the training set of the ML-RWR algorithm is X_{train}. There are $m + 1$ vertices in its random walk graph, and edge set construction needs $m + 1$ times of sorting operations, but in the MLRWKNN algorithm, the corresponding vertex number is $K + 1(K << m)$ and only one sorting operation is executed on X_{train} for each label, see section "A new construction method of the random walk graph."

Analysis of parameters selection. There are three parameters, K (see equation 6), adjustment factor σ(see equation 8), and jump probability α (see equation 13), that we need to select for the MLRWKNN algorithm. A suitable value is selected for these parameters is crucial since either too large or too small of the value to be selected will cause problems for the algorithm. Too small a value is selected for the parameter K will lead to the problem of over-fitting. On the contrary, too large K will cause the instances' influence related to x_0 to decrease for label prediction. This situation will not only increase the computation complexity in time and space but also cause erroneous classification results.[42] For the parameter σ, too small of it will lead to over-fitting and too large will produce a failure of instances classification.[39] For the parameter α, too small of it will

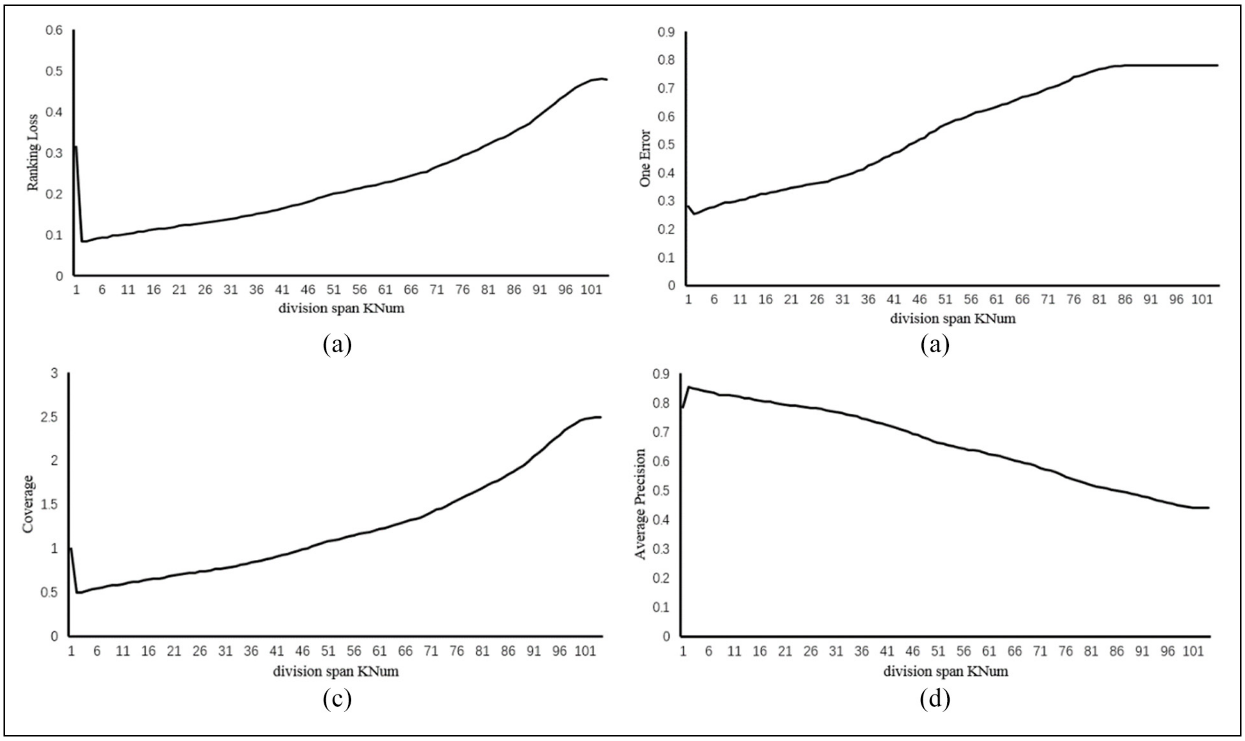

Figure 4. The relationship between different *K* values and evaluation criteria on the Scene dataset.

cause the label prediction to be over-sensitive to the change in the state transition matrix while too large will cause the convergence speed of the random walk to decrease.[19] In order to determine an accurate value for the above parameters, in the following, we use the Scene dataset as sample data to illustrate the impact of the parameter determination on the classification performance of the MLRWKNN algorithm. The optimal settings are also provided.

Experimental analysis of the selection of the parameter K. Different from adopting fixed constant values, we dynamically determine an optimal K value for different datasets during the model training procedure. Specifically, the gradual refinement method was adopted to determine the K value. First, different K values divide the training set into n parts (for instance, $n = 100$) and $K = 1 + (m - 1/n - 1)(KNum - 1)$, where $KNum$ represents the division span and $KNum = 1, 2, \ldots, n$. Then, an approximate interval that contains the best K value is chosen. By repeating the above process, the best K value is confirmed.

From Figure 4, for the Scene dataset, it can be observed that a gradual increase of the K value improves the experiment results in terms of the four evaluation criteria, that is, RL, OE, Coverage, and AP. If K continues to increase to the optimal value, the evaluation results will become worse or remain stable, which indicates that an appropriate K values can be reached for a best algorithm performance. It is noted that for different measurement metrics, the optimal K values are different. We select an optimal K value through synthetizing all the measurement metrics in the model training procedure.

Experimental analysis of the selection of the σ value. For convenience, we determine an optimal σ in terms of the parameter u in the interval of [0, 1], $\sigma = \min_{x_i, x_j \in N_{x_0}^K, i \neq j} similarity(x_i, x_j) + u * (\max_{x_i, x_j \in N_{x_0}^K, i \neq j} similarity(x_i, x_j) - \min_{x_i, x_j \in N_{x_0}^K, i \neq j} similarity(x_i, x_j))$. Suppose that we take n groups of data in $[0, 1]$, then $u = (uNum - 1)/n$, where $uNum$ represents the division span, taking a value from $(1, 2, \ldots, n)$. Figure 5 shows the classification performance of the MLRWKNN algorithm with different values of σ on the Scene dataset.

As shown in Figure 5, when u increases, the classification performance gradually decreases. After reaching its lowest performance value, the classification performance will gradually improve as u continues to increase. This indicates that, in general, an optimal σ value can be determined when it is close to a minimum or maximum distance between two training instances. The experiments on the other five datasets show the same results.

Experiment analysis of the selection of the α value. We assign n (e.g. $n = 101$) different values to $\alpha \in [0, 1]$ in

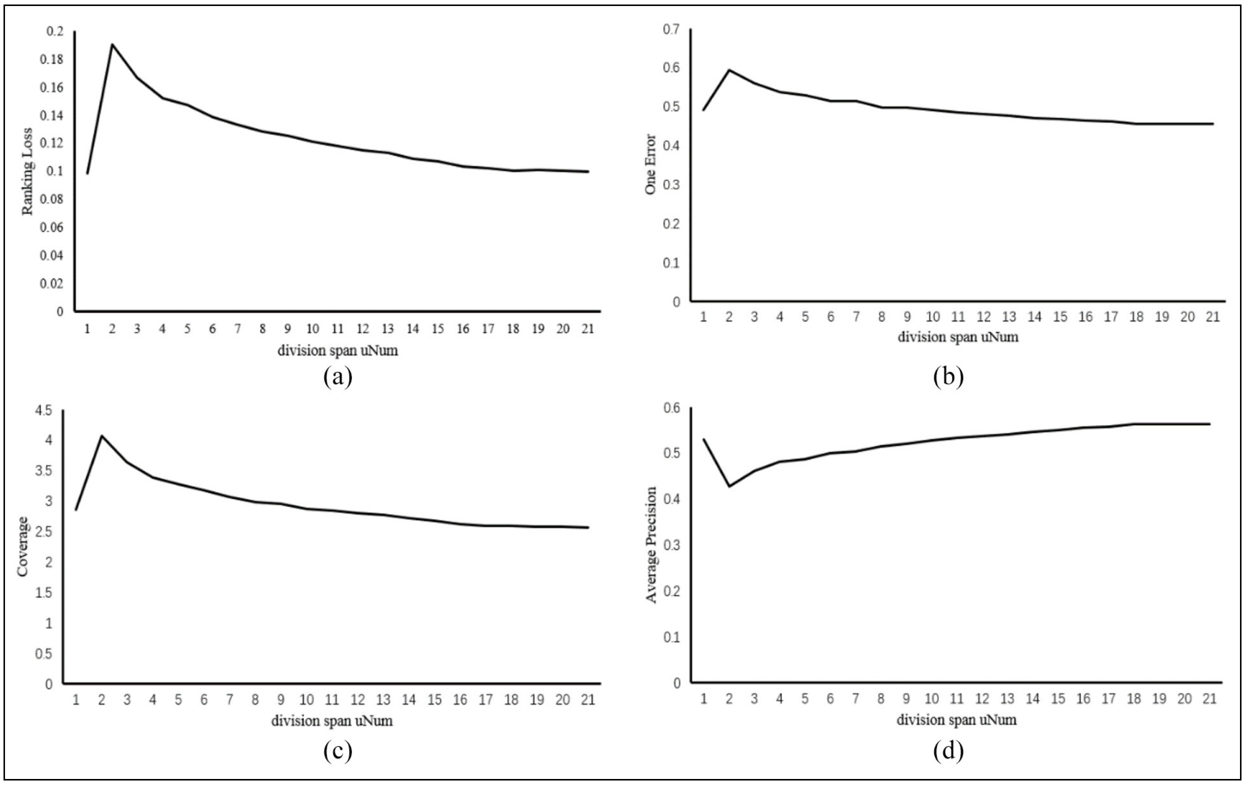

Figure 5. The relationship between different σ values and evaluation criteria on Scene dataset.

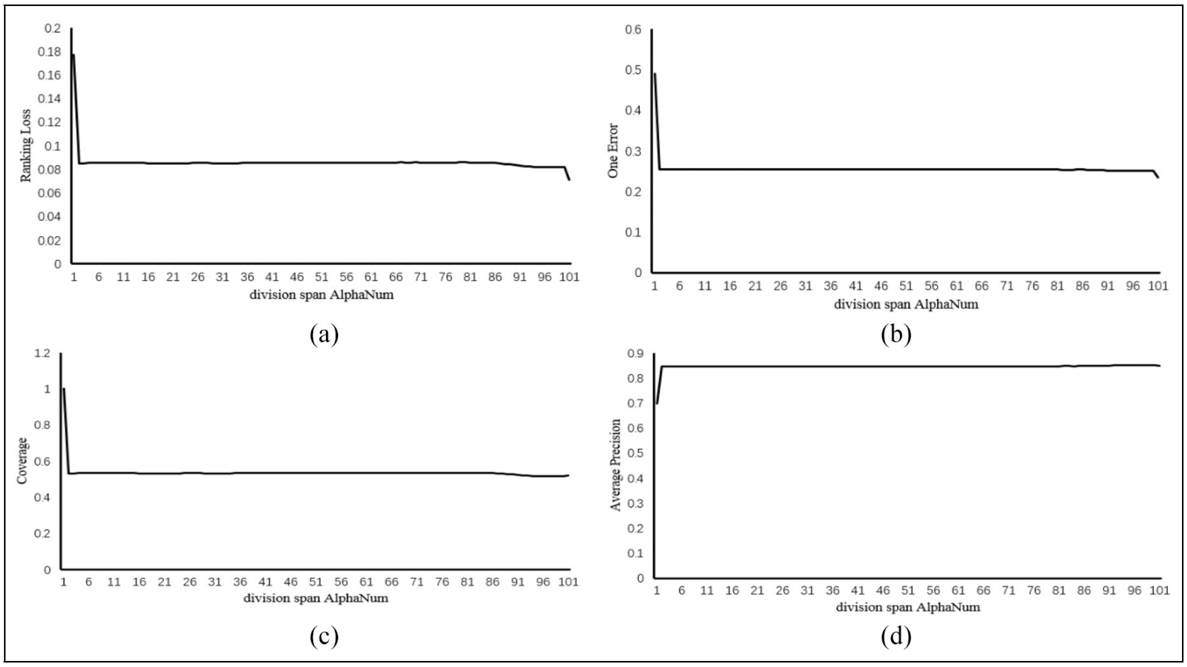

Figure 6. The relationship between different α values and evaluation criteria on Scene dataset.

order to test and determine a most suitable one for selection, and let $\alpha = (AlphaNum - 1)/n$, where $AlphaNum = 1, 2, \ldots, n$ represents the division span. The classification performance of MLRWKNN is shown in Figure 6 with different α values on the Scene dataset.

Table 5. The optimal K values on different datasets.

	Yeast	Genbase	Flags	Scene	Medical	Mediamill
K	32	3	23	29	4	4

From Figure 6, it can be observed that different α values have little effect on the classification results. The experiments on the other five datasets produced the same results. However, the curves of each subgraph jump at both ends. This occurs because when random walks only transfer between vertices with edges, the walks only randomly jump between vertices when $\alpha = 0$.

Briefly, from the above analysis, we can observe that an optimal K value for the algorithm is dynamically determined during the model training procedure. An optimal σ value is generally selected near a minimum or maximum distance between two training instances. The values of α have little influence on the classification results, so in our experiments, we set to 0.15, as in Wang et al.[18]

Comparison of experimental results of seven algorithms. We select seven most cited MLC algorithms, that is, BRKNN, CLR, HOMER, LP, RAkEL, MLKNN, and Rank-SVM, for a comparison study in contrast to the proposed MLRWKNN algorithm. As the MLRW algorithm adopts the continuous feature-based similarity measurement and uses m for the parameter K, the discussions in section "Analysis on the proposed similarity measurement" to section "Analysis of parameters selection" have indicated that it is only a special case of MLRWKNN under the non-optimal graph construction mode, the non-optimal similarity measurement, and the non-optimal parameters selection, so there is no need to include the MLRW in this comparison study.

The experimental datasets are described in Table 1, and the optimal values of the parameter K on different datasets are described in Table 5. The parameter σ adopts the maximum distance between the training instances, which is 0.15. The experimental results are shown in Tables 6–11, with each dataset in a table (note that the numbers in the parentheses represent the ranking of the eight algorithms under the corresponding criteria).

In summary, as observed from Tables 6–11, the LP algorithm demonstrates a poor performance on all the six datasets among these eight algorithms. The other six algorithms show a low performance on some datasets. More specifically, the HOMER algorithm

Table 6. Performance comparisons of eight algorithms on Yeast.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.1778 (3)	0.2309 (3)	6.5245 (4)	0.7599 (3)
CLR	0.1783 (4)	0.2412 (4)	6.7008 (5)	0.7459 (4)
HOMER	0.3287 (7)	0.2871 (6)	9.2457 (7)	0.6259 (7)
LP	0.3977 (8)	0.5143 (8)	9.3607 (8)	0.5733 (8)
MLKNN	0.1652 (2)	0.2292 (2)	6.2324 (2)	0.7658 (1)
Rank-SVM	0.1928 (5)	0.2521 (5)	6.3554 (3)	0.7217 (5)
RAkEL	0.2143 (6)	0.2946 (7)	7.5086 (6)	0.7144 (6)
MLRWKNN	0.1634 (1)	0.2270 (1)	6.2166 (1)	0.7647 (2)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

Table 7. Performance comparisons of eight algorithms on Flags.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.1978 (3)	0.1853 (1)	3.7082 (4)	0.8280 (3)
CLR	0.1901 (2)	0.1958 (4)	3.6216 (2)	0.8324 (2)
HOMER	0.2895 (6)	0.3703 (6)	4.1221 (6)	0.763 (6)
LP	0.5011 (7)	0.5587 (7)	4.9495 (7)	0.6407 (7)
MLKNN	0.2012 (4)	0.1905 (2)	3.6705 (3)	0.8255 (4)
Rank-SVM	0.7052 (8)	0.7618 (8)	5.5303 (8)	0.4987 (8)
RAkEL	0.2332 (5)	0.2255 (5)	3.7824 (5)	0.8118 (5)
MLRWKNN	0.1843 (1)	0.1907 (3)	3.5474 (1)	0.8391 (1)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

Table 8. Performance comparisons of eight algorithms on Genbase.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.0052 (4)	0.0166 (8)	0.4367 (4)	0.9820 (8)
CLR	0.0083 (8)	0.0015 (1)	0.6481 (8)	0.9914 (3)
HOMER	0.0049 (3)	0.0091 (3)	0.4636 (5)	0.9875 (5)
LP	0.0076 (7)	0.0106 (5)	0.4997 (6)	0.9871 (6)
MLKNN	0.0062 (6)	0.0136 (6)	0.5559 (7)	0.9864 (7)
Rank-SVM	0.0060 (5)	0.0152 (7)	0.2727 (1)	0.9937 (1)
RAkEL	0.0026 (2)	0.0091 (3)	0.3429 (2)	0.9912 (4)
MLRWKNN	0.0003 (1)	0.0045 (2)	0.3621 (3)	0.9918 (2)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

Table 9. Performance comparisons of eight algorithms on Scene.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.0889 (3)	0.2522 (3)	0.5314 (3)	0.8496 (3)
CLR	0.1011 (5)	0.3020 (6)	0.5912 (5)	0.8209 (6)
HOMER	0.2345 (8)	0.4595 (8)	1.2685 (8)	0.6946 (8)
LP	0.2121 (7)	0.3984 (7)	1.1550 (7)	0.7308 (7)
MLKNN	0.0774 (1)	0.2243 (1)	0.4744 (1)	0.8662 (1)
Rank-SVM	0.1039 (6)	0.2863 (5)	0.6266 (6)	0.8275 (5)
RAkEL	0.0998 (4)	0.2672 (4)	0.5854 (4)	0.8378 (4)
MLRWKNN	0.0853 (2)	0.2463 (2)	0.5117 (2)	0.8522 (2)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

Table 10. Performance comparisons of eight algorithms on the Mediamill dataset.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.0632 (3)	0.1570 (2)	20.9691 (2)	0.7390 (2)
CLR	0.1041 (5)	0.9322 (8)	21.4783 (3)	0.2398 (8)
HOMER	0.2526 (7)	0.4261 (6)	57.7752 (8)	0.4975 (6)
LP	0.3388 (8)	0.5851 (7)	57.6239 (7)	0.3510 (7)
MLKNN	0.0360 (2)	0.1594 (3)	12.4433 (1)	0.7503 (1)
Rank-SVM	0.0712 (4)	0.2252 (5)	29.1329 (5)	0.6290 (4)
RAkEL	0.1149 (6)	0.1924 (4)	35.6374 (6)	0.6917 (3)
MLRWKNN	0.0091 (1)	0.0374 (1)	27.4020 (4)	0.5220 (5)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

Table 11. Performance comparisons of eight algorithms on the Medical dataset.

Algorithms	Ranking Loss	One Error	Coverage	Average Precision
BRKNN	0.0474 (4)	0.3067 (8)	2.9697 (4)	0.7686 (6)
CLR	0.0503 (5)	0.2774 (7)	2.9536 (3)	0.7606 (7)
HOMER	0.1199 (7)	0.2108 (3)	6.6711 (7)	0.7866 (4)
LP	0.1398 (8)	0.2373 (5)	7.7768 (8)	0.7501 (8)
MLKNN	0.0395 (3)	0.2577 (6)	2.6012 (2)	0.8062 (3)
Rank-SVM	0.0274 (2)	0.1939(2)	1.7347(1)	0.8697(1)
RAkEL	0.0780 (6)	0.1841 (1)	4.4026 (6)	0.8299 (2)
MLRWKNN	0.0103 (1)	0.2186 (4)	4.1320 (5)	0.7704 (5)

BRKNN: binary relevance KNN; CLR: calibrated label ranking; HOMER: hierarchy of multi-label classifiers; LP: label powerset; MLKNN: multi-label KNN; Rank-SVM: rank support vector machine; RAkEL: RAndom *k*-labELsets; MLRWKNN: multi-label classification based on random walk graph and KNN algorithm.

performs not well on the datasets of Yeast, Flags, Scene and Mediamill, the Rank-SVM algorithm receives the worst results on the Flags dataset, while the MLKNN algorithm is not good with the Genbase dataset. In terms of the four assessment criteria, the proposed MLRWKNN algorithm outperforms all the other seven algorithms with the best comprehensive performance on the datasets of Yeast, Flags and Genbase, and achieves the above-average performance on the datasets of Scene, Mediamill and Medical. This comparison result demonstrates that the proposed algorithm achieves a stable and significant classification effect on the six commonly used datasets in contrast to the other most-cited seven algorithms.

Conclusion

The MLC problem is an important, significant, and widely influencing research problem in the field of data mining, impacting a wide range of applications in real world domains. The graph-based MLC algorithms have, in recent years, received an increasing research attention and introducing the random walk-based methods into the solutions to the MLC problem has also become a hot research topic. In this article, we propose a novel approach, the MLRWKNN algorithm, and research direction to tackle this problem through integrating the random walk methods in the graph-based MLC algorithms. Our major contributions to the field of MLC in this article include the following three aspects. First, a new paradigm is proposed for a graph model that constructs a vertex set via the KNN training instances for certain test instances and determines the label correlations of the vertices samples to build the edge set. This new paradigm can reduce the overhead complexity in time and space compared with other graph-based models. Second, a novel label set prediction method is developed by introducing the similarity measurement. This method overcomes the problem of subjectivity of determining thresholds in the traditional methods and performs the similarity computation more accurately via differentiating and integrating discrete and continuous features. Third, we consider the influences of the parameter selection and determination on the classification performance and suggest the guidelines of the selection principles and recommended values for the algorithm parameters. A good number of experiments have been conducted and significant comparisons have been made on the six datasets among the seven algorithms and ours, in order to evaluate the proposed similarity measurement, the new construction method for graph model, and the MLRWKNN algorithm. The experiment results demonstrate that the proposed MLRWKNN algorithm produce a much better result than the other seven state-of-the-art MLC algorithms.

Future work: there are efforts required to improve and extend the proposed method and algorithm. First, in this work, we considered only a linear integration of the similarity metric for the two continuous and discrete datasets. To explore a further theoretical similarity computation, it will be interesting and useful to investigate a nonlinear combination style for the proposed similarity measurement which takes the dataset features of continuity and discreteness into account. Second, it is a challenge to tackle the issues of an adaptive adjustment mechanism for the determination of algorithm parameters, in order to enhance the accuracy and automation of the MLC methods and tools. Third, deep learning approaches[43] have been recently widely explored and considered to have a strong impact on various application domains. Integrating deep learning or rule learning methods in the MLC methods[44,45] is attracting researchers and some interesting results have been observed. We believe this is another significant direction for our future work. Fourth, a newly published paper proposed a semi-supervised learning (SSL) method based on random walk,[40] which leverages the notation of "landing probabilities" of class-specific random walks and aims at a great improvement in computational complexity and scalability for large graphs. This article highlights another way of investigation of MLC problems. As a part of our next step of MLRWKNN, we consider contrasting this method to our method on KNN and random walk and making comparison study of algorithm performance through experiments with the datasets used in Berberidis et al.[40]

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by National Science Foundation of China (Grant Nos 61302157 and 61363075), in part by National High Technology Research and Development Program ("863"Program) of China (Grant 2012AA12A308), in part by Yue Qi Young Scholars Program of China University of Mining & Technology, Beijing (Grant 800015Z1117), in part by Department of Science and Technology of Jiangxi Province of China (Grant Nos 20161BBG70078 and KJLD12023), and in part by Department of Education of Jiangxi Province of China (Grant No. GJJ180270). Finally, the authors are in debt to the anonymous reviewers with their constructive comments.

ORCID iD

Zhen-Wu Wang https://orcid.org/0000-0002-8620-4623

References

1. Burkhardt S and Kramer S. Online multi-label dependency topic models for text classification. *Mach Learn* 2018; 107: 859–886.
2. Nanculef R, Flaounas I and Cristianini N. Efficient classification of multi-labeled text streams by clashing. *Expert Syst Appl* 2014; 41: 5431–5450.
3. Vateekul P, Kubat M and Sarinnapakorn K. Hierarchical multi-label classification with SVMs: a case study in gene function prediction. *Intell Data Anal* 2014; 18: 717–738.
4. Keikha MM, Rahgozar M and Asadpour M. Community aware random walk for network embedding. *Knowl-Based Syst* 2018; 148: 47–54.
5. Markatopoulou F, Mezaris V and Patras I. Implicit and explicit concept relations in deep neural networks for multi-label video/image annotation. *IEEE T Circ Syst Vid* 2019; 29: 1631–1644.
6. Claudio G, Alessio M, Paolo B, et al. User movements forecasting by reservoir computing using signal streams produced by mote-class sensors. In: *Proceedings of the international conference on mobile lightweight wireless systems*, Bilbao, 9–10 May 2011, pp.151–168. Berlin: Springer.
7. Claudio G, Davide B, Alessio M, et al. Predicting user movements in heterogeneous indoor environments by reservoir computing. In: *Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI)*, Barcelona, 16–22 July 2011, pp.1–6. AAAI Press.
8. Antonio AL, Paolo B, Mario GCA, et al. Sleep behavior assessment via smartwatch and stigmergic receptive fields. *Pers Ubiquit Comput* 2018; 22(2): 227–243.
9. Guraliuc AR, Paolo B, Francesco P, et al. Limb movements classification using wearable wireless transceivers. *IEEE T Inf Technol B* 2011; 15(3): 474–480.
10. Barsocchi P. Position recognition to support bedsores prevention. *IEEE J Biomed Health* 2013; 17(1): 53–59.
11. Belmannoubi S, Touati H and Snoussi H. Stacked autoencoder for scalable indoor localization in wireless sensor networks. In: *Proceedings of the 15th international wireless communications and mobile computing conference*, Tangier, Morocco, 24–28 June 2019, pp.1245–1250. New York: IEEE.
12. Zhang Z, Li S and Li Z. Data fault detection algorithm based on multi-label classification in sensor network. *Appl Res Comput* 2014; 31(12): 3788–3791, 3817 (in Chinese).
13. Ni K, Ramanathan N, Nabil H, et al. Sensor network data fault types. *ACM T Sensor Network* 2009; 5(3): 1–29.
14. Zhang ML and Zhou ZH. A review on multi-label learning algorithms. *IEEE T Knowl Data En* 2014; 26(8)): 1819–1837.
15. Cetiner M and Akgul YS. A graphical model approach for multi-label classification. In: *Proceedings of the 29th international symposium on computer and information sciences*, Krakow Old City, 27–28 October 2014, pp.61–67. Cham: Springer.
16. Jiang Y, She Q, Li M, et al. A transductive multi-label text categorization approach. *J Comput Res Dev* 2008; 45: 1817–1823 (in Chinese).
17. Zheng W, Wang CK, Liu Z, et al. A multi-label classification algorithm based on random walk model. *Chin J Comput* 2010; 8: 1418–1426 (in Chinese).
18. Wang C, Zheng W, Liu Z, et al. Using random walks for multi-label classification. In: *Proceedings of the 20th ACM international conference on information and knowledge management*, Glasgow, 24–28 October 2011, pp.2197–2200. New York: ACM.
19. Xia X, Yang X, Li S, et al. RW.KNN: a proposed random walk KNN algorithm for multi-label classification. In: *Proceedings of the 4th workshop on workshop for Ph.D. students in information & knowledge management (PIKM '11)*, Glasgow, 24–28 October 2011, pp.87–90. New York: ACM.
20. Wang H, Huang H and Ding C. Image annotation using bi-relational graph of images and semantic labels. In: *Proceedings of the IEEE Computer Society conference on computer vision and pattern recognition*, Providence, RI, 20–25 June 2011, pp.793–800. New York: IEEE.
21. Song K, Feng S, Gao W, et al. Build emotion lexicon from microblogs by combining effects of seed words and emoticons in a heterogeneous graph. In: *Proceedings of the 26th ACM conference on hypertext & social media (HT '15)*, Guzelyurt, Cyprus, 1–4 September 2015, pp.283–292. New York: ACM.
22. Yao L, Sheng QZ, Ngu AHH, et al. Multi-label classification via learning a unified object-label graph with sparse representation. *World Wide Web* 2016; 19: 1125–1149.
23. Liu J and Yang J. Multi-label classification using random walk with restart. In: *Proceedings of the 2017 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC)*, Nanjing, China, 12–14 October 2017, pp.206–212. New York: IEEE.
24. Zhang CG and Zhang XH. Graph-based semi-supervised multi-label learning method. In: *Proceedings of the 2013 international conference on mechatronic sciences, electric engineering and computer (MEC'13)*, Shenyang, China, 20–22 December 2013, pp.1021–1025. New York: IEEE.
25. Yankelevsky Y and Elad M. Graph-constrained supervised dictionary learning for multi-label classification. In: *Proceedings of the 2016 IEEE international conference on the science of electrical engineering (ICSEE'16)*, Eilat, 16–18 November 2016, pp.1–5. New York: IEEE.
26. Fu B, Wang Z, Xu G, et al. Multi-label learning based on iterative label propagation over graph. *Pattern Recogn Lett* 2014; 42: 85–90.
27. Jiang G, Liu X and Shi Z. Multi-label image annotation based on neighbor pair correlation chain. In: *Proceedings of the machine learning and data mining in pattern recognition (MLDM 2012); Lecture Notes in Computer Science*, vol. 7376, no. 61035003. Berlin: Springer, 2012; pp.345–354. Berlin: Springer.
28. Lee J, Kim H, Kim N, et al. An approach for multi-label classification by directed acyclic graph with label correlation maximization. *Inform Sciences* 2016; 351: 101–114.
29. Xia X, Yang X, Li S, et al. Instance-ranking: a new perspective to consider the instance dependency for classification. In: Washio T and Luo J (eds) *Proceedings of the emerging trends in knowledge discovery and data mining (PAKDD 2012); Lecture notes in computer science*. Berlin: Springer, 2013, pp.112–123.

30. Song WW, Lin CL, Forsman A, et al. A Euclidean similarity measurement approach for hotel rating data analysis. In: *Proceedings 2017 2nd IEEE international conference on cloud computing and big data analysis*, Chengdu, China, 28–30 April 2017, pp.293–298. New York: IEEE.
31. Fürnkranz J, Hüllermeier E, LozaMencía E, et al. Multilabel classification via calibrated label ranking. *Mach Learn* 2008; 73: 133–153.
32. Tsoumakas G, Katakis I and Vlahavas I. Effective and efficient multilabel classification in domains with large number of labels. In: *Proceedings of the ECML/PKDD 2008 workshop on mining multidimensional data (MMD'08)*, Antwerp, 15–19 September 2008, pp.30–44. Springer.
33. Read J, Pfahringer B and Holmes G. Multi-label classification using ensembles of pruned sets. In: *Proceedings of the 8th IEEE international conference on data mining*, Pisa, 15–19 December 2008, pp.995–1000. New York: IEEE.
34. Tsoumakas G, Katakis I and Vlahavas I. Random k-labelsets for multilabel classification. *IEEE T Knowl Data En* 2011; 23: 1079–1089.
35. Zhang ML and Zhou ZH. ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recogn* 2007; 40: 2038–2048.
36. Elisseeff A and Weston J. A kernel method for multilabelled classification. In: *Proceedings of the 15th annual neural information processing systems conference*, Vancouver, BC, Canada, 3–8 December 2001, pp.681–687. New York: ACM.
37. Jiang ZL, Lin Z and Davi LS. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In: *Proceedings of the 2011 IEEE conference on computer vision and pattern recognition*, Colorado Springs, CO, 20–25 June 2011, pp.1697–1704. New York: IEEE.
38. Azran A. The rendezvous algorithm: multiclass semisupervised learning with Markov random walks. In: *Proceedings of the 24th international conference on machine learning*, Corvallis, OR, 20–24 June 2007, pp.49–56. New York: ACM.
39. Wang WJ, Xu Z, Lu W, et al. Determination of the spread parameter in the Gaussian kernel for classification and regression. *Neurocomputing* 2003; 55: 643–663.
40. Berberidis D, Nikolakopoulos AN and Giannakis GB. Adaptive diffusions for scalable learning over graphs. *IEEE T Signal Proces* 2019; 67(5): 1307–1321.
41. Tsoumakas G, Katakis I and Vlahavas I. Mining multilabel data. In: Rokach L and Maimon OZ (eds) *Data mining and knowledge discovery handbook*. New York: Springer, 2010, pp.667–685.
42. Li H. *Statistical learning method*. 1st ed. Beijing, China: Tsinghua University Press, 2012.
43. LeCun Y, Bengio Y and Hinton G. Deep learning. *Nature* 2015; 521: 436–444.
44. Nam J, Kim Y, Mencía EL, et al. Learning context-dependent label permutations for multi-label classification. In: *Proceedings of the 36th international conference on machine learning*, Long Beach, CA, 9–15 June 2019, pp.1–10. PMLR.
45. Mencía EL, Fürnkranz J, Hüllermeier E, et al. Learning interpretable rules for multi-label classification. In: Jair Escalante H, Escalera S, Guyon I, et al. (eds) *Explainable and interpretable models in computer vision and machine learning*. Cham: Springer-Verlag, 2018, pp.81–113.