Int. J. Mach. Learn. & Cyber. (2014) 5:193–199
DOI 10.1007/s13042-013-0152-x

ORIGINAL ARTICLE

Bayesian Citation-KNN with distance weighting

Liangxiao Jiang · Zhihua Cai · Dianhong Wang · Harry Zhang

Received: 2 October 2012 / Accepted: 1 February 2013 / Published online: 12 February 2013

Abstract Multi-instance (MI) learning is receiving growing attention in the machine learning research field, in which learning examples are represented by a bag of instances instead of a single instance. K-nearest-neighbor (KNN) is a simple and effective classification model in the traditional supervised learning. As its two variants, Bayesian-KNN (BKNN) and Citation-KNN (CKNN) are proposed and are widely used for solving multi-instance classification problems. However, CKNN still applies the simplest majority vote approach among the references and citers to classify unseen bags. In this paper, we propose an improved algorithm called Bayesian Citation-KNN (BCKNN). For each unseen bag, BCKNN firstly finds its *k* references and *q* citers respectively, and then a Bayesian approach is applied to its *k* references and a distance weighted majority vote approach is applied to its *q* citers. The experimental results on several benchmark datasets show that our BCKNN is generally better than previous BKNN and CKNN. Besides, BCKNN almost maintains the same order of computational overhead as CKNN.

L. Jiang (✉) · Z. Cai
Department of Computer Science,
China University of Geosciences, Wuhan 430074, China
e-mail: ljiang@cug.edu.cn

D. Wang
Department of Electronic Engineering,
China University of Geosciences, Wuhan 430074, China

H. Zhang
Faculty of Computer Science, University of New Brunswick,
Fredericton E3B5A3, Canada

Keywords Multi-instance learning · KNN · Bayesian-KNN · Citation-KNN · Bayesian Citation-KNN · Distance weighting

1 Introduction

Multi-instance learning (MI learning) has received much attention in the machine learning research field. MI learning is a variation of the standard supervised learning. In standard supervised learning, each example is an instance represented by an attribute vector, augmented with a class label. The learning task is to build a classifier that predicts the class label of an unseen instance, given its attribute vector. In MI learning, however, each example consists of a bag of instances. Each bag has a class label, but the instances themselves are not labeled. Therefore, the learning task is to build a classifier that predicts the class label of an unseen bag [1]. Recently, some researchers combine multi-instance learning and multi-label learning [2] to propose another machine learning framework: Multi-Instance Multi-label learning (MIML learning) [3, 4]. In this paper, we temporarily focus our attention on the multi-instance learning.

Two different approaches have been adopted to classify an unseen bag of instances in the context of multi-instance problem. The first approach classifies a bag as negative if all the instances in it are negative and positive if at least one instance in it is positive [5]. In contrast, another approach classifies a bag as the maximum label among all instances in it using the simplest majority vote approach [6].

MI learning was original motivated by the drug activity prediction problem where each instance is a possible conformation of a molecule and each bag contains all likely low-energy conformations for the molecule. A molecule is active if it binds strongly to the target protein in at least one of its conformations and is inactive if no conformation binds to the protein. Thus the problem is to predict the label (active or inactive) of molecules based on their conformations [7].

In recent years, in additional to the drug activity prediction problem, a variety of learning problems have been tackled as multi-instance problems. For example, several inductive logic programming problems [8, 9], identifying thioredoxin-fold proteins [10], content-based image retrieval [11, 12], stock market prediction [13], text categorization [14], natural scene classification [15], image categorization [16], etc.

In order to address these multi-instance problems, a large number of algorithms are proposed, which can be broadly divided into two main categories: eager learning and lazy learning [17, 18]. It appears that up to now most of these algorithms belong to eager learning. For example, Dietterich et al. [5] assumed that the classifier could be represented as an axis-parallel rectangle, and proposed four typical algorithms. Among them, the iterated-discrim APR algorithm is the best one. Auer [19] focused on theoretical research and presented the MULTINST algorithm to efficiently learn axis-parallel concept. After that Maron and Lozano-Perez [20] described a new general framework called Diverse Density (DD), Zhang and Goldman [7] proposed an improved algorithm called EM-DD by combining EM and the extended DD. Multi-instance Learning via Embedded Instance Selection (MILES) is a recent MI learning approach presented by Chen et al. [21]. Then, Foulds and Frank [22] revisited the MILES algorithm and presented an empirical study investigating the efficacy of alternative base learners for MILES. Besides, TILDE [23] and RELIC [6] are two of top-down decision tree induction systems for MI learning.

For lazy learning, Wang and Zucker [18] proposed two variants of the K-nearest-neighbor (KNN) algorithm (Bayesian-KNN and Citation-KNN). In this paper, we denote them by BKNN and CKNN respectively. Their experimental results on the drug discovery benchmark datasets show that both BKNN and CKNN are competitive with the best ones conceived in the concept learning framework. After that, Xu [24] proposed a nearest distribution approach to multi-instance learning (MINND). Besides, the KNN algorithm can be used as the basic classifier for some meta multi-instance learning algorithms such as MIBoost [25] and MIWrapper [26].

According to the paper by Wang and Zucker [18], CKNN still applies the simplest majority vote approach among the references and citers to classify unseen bags. In this paper, we proposed an improved algorithm called Bayesian Citation-KNN (BCKNN). For each unseen bag, BCKNN firstly finds its k references and q citers respectively, and then a Bayesian approach is applied to its k references and a distance weighted majority vote approach is applied to its q citers. The experimental results on two drug discovery benchmark datasets validate its effectiveness and efficiency. Besides, in order to adapt KNN to the multi-instance learning, several alternative distance functions and classification approaches are introduced in our paper.

The rest of the paper is organized as follows. In Sect. 2, we briefly introduce the KNN algorithm and its two variants BKNN and CKNN and then discuss how to adapt KNN to the multi-instance learning. Section 3 proposes our improved algorithm called Bayesian Citation-KNN (BCKNN). Section 4 gives the detailed experimental setup and the compared results on several benchmark datasets. Finally, we draw conclusions and outline the main directions for our future work in Sect. 5.

2 Adapting K-nearest-neighbor to the multi-instance learning

K-nearest-neighbor (KNN) has been widely used as a simple and effective classification model in the traditional supervised learning [27]. KNN assumes all instances correspond to points in the m dimensional real space R^m. The nearest neighbors of an unseen instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the attribute vector $\langle a_1(x), a_2(x), \ldots, a_m(x)\rangle$, where $a_j(x)$ denotes the value of the jth attribute A_j of the instance x. Then the distance between two instances x and y is defined as:

$$d(x,y) = \sqrt{\sum_{j=1}^{m} (a_j(x) - a_j(y))^2}. \qquad (1)$$

In KNN, the target function can be either discrete-valued (classification) or real-valued (regression). This paper considers learning discrete-valued target functions of the form $f : R^m \rightarrow C$, where C is a finite set $\{c_1, c_2, \ldots, c_s\}$. For an unseen instance y, KNN needs to find its k-nearest neighbors at first. We assume its k-nearest neighbors are $\{x_1, x_2, \ldots, x_k\}$ and their class labels are respectively $\{c_1, c_2, \ldots, c_k\}$. Then, KNN uses the simplest majority vote approach to determine the class label of y. Therefore, the detailed classification formulation of KNN is

$$c_{KNN}(y) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c_i) \qquad (2)$$

where $\delta(c, c_i) = 1$ if $c = c_i$, and 0 otherwise.

KNN is a typical example of lazy learning, which simply stores training instances at training time and delays its learning process until classification time. In contrast, eager learning generates an explicit model at training time. Due to its simplicity and effectiveness, KNN has been widely used for classification in the traditional supervised learning. In this paper, however, we focus our attention on multi-instance classification problems instead of traditional supervised classification problems. Thus, a natural question is how to adapt KNN to multi-instance classification problems.

According to the observation by Wang and Zucker [18], two practical problems in adapting KNN to multi-instance learning must be addressed, which are the distance measure problem and the classification approach problem. To address the first problem, four distance functions [28–30], such as the minimum distance, the maximum distance, the centroid distance, and the Hausdorff distance, have been proposed.

Given two sets of instances $X = \{x_1, x_2, \ldots, x_o\}$ and $Y = \{y_1, y_2, \ldots, y_p\}$, above distance functions can be respectively defined as:

$$D_{\mathrm{Min}}(X,Y) = \operatorname*{Min}_{x\in X}\left\{\operatorname*{Min}_{y\in Y}\{d(x,y)\}\right\} \tag{3}$$

$$D_{\mathrm{Max}}(X,Y) = \operatorname*{Max}_{x\in X}\left\{\operatorname*{Max}_{y\in Y}\{d(x,y)\}\right\} \tag{4}$$

$$D_{cen}(X,Y) = d\left(\frac{1}{o}\sum_{i=1}^{o} x_i, \frac{1}{p}\sum_{j=1}^{p} y_j\right) \tag{5}$$

$$D_H(X,Y) = \mathrm{Max}\left\{\operatorname*{Max}_{x\in X}\left\{\operatorname*{Min}_{y\in Y}\{d(x,y)\}\right\},\ \times \operatorname*{Max}_{y\in Y}\left\{\operatorname*{Min}_{x\in X}\{d(x,y)\}\right\}\right\} \tag{6}$$

where $d(x,y)$ is the standard Euclidean distance between two instances x and y, $\frac{1}{o}\sum_{i=1}^{o} x_i$ and $\frac{1}{p}\sum_{j=1}^{p} y_j$ are the centroids of the instance sets X and Y respectively. In Sect. 4, we design three groups of experiments to validate the effectiveness of different distance functions. The experimental results show that the maximum distance seems to be the worst and the minimum distance seems to be the best among all four distance functions. The more detailed experimental results can be found from Tables 2, 3 and 4 in Sect. 4.

To address the second problem, instead of the simplest majority vote approach, another two alternative classification approaches, namely the Bayesian approach and the citation approach are proposed by Wang and Zucker [18]. They call the resulting algorithms Bayesian-KNN (BKNN) and Citation-KNN (CKNN) respectively.

BKNN uses Eq. 7 to classify y.

$$c_{BKNN}(y) = \arg\operatorname*{Max}_{c\in C} P(c)P(c_1, c_2, \ldots, c_k|c) \tag{7}$$

where C is a finite set $\{c_1, c_2, \ldots, c_s\}$. It appears that up to now almost all MI classification problems are binary classification problems. Namely, c_i is either positive or negative. Thus, the maximal number of combination that $\{c_1, c_2, \ldots, c_k\}$ can take is $k+1$. Therefore, the computational cost of BKNN is not expensive.

In CKNN, the concept of citation is used and the nearest neighbors are extended to the combination of references and citers. Give an unseen bag y, its k-nearest references are $\{x_1, x_2, \ldots, x_k\}$, and its c-nearest citers are $\{x_1, x_2, \ldots, x_q\}$. Please note that, the parameter c is different from the number q. Then, CKNN uses Eq. 8 to classify y.

$$c_{CKNN}(y) = \arg\operatorname*{Max}_{c\in C}\left(\sum_{i=1}^{k}\delta(c,c_i) + \sum_{j=1}^{q}\delta(c,c_j)\right) \tag{8}$$

where $\{c_1, c_2, \ldots, c_k\}$ and $\{c_1, c_2, \ldots, c_q\}$ are the class labels of k references and q citers of y respectively. Their experimental results on two drug discovery benchmark datasets show that both BKNN and CKNN are competitive with the best ones conceived in the concept learning framework [18].

3 Bayesian Citation-KNN with distance weighting

Our research starts from our comments to CKNN. Just as discussed before, CKNN still applies the simplest majority vote approach among the references and citers to classify unseen bags. That is say that the class labels of all references and citers in CKNN are treated equally. That essentially means that each reference and citer is treated equally. However, intuitively, different references and citers should play different roles in classifying the unseen bag, since some of them are more important than others in classification. Thus, a natural way to improve CKNN is to respectively assign different weights to different references and citers according to their distances to the unseen bag. Besides, the Bayesian approach has been proved to be much better than the simplest majority vote approach [18].

Motivated by these facts, in this paper, we propose an improved algorithm called Bayesian Citation-KNN (BCKNN) by combining the Bayesian approach and the distance weighting approach. For each unseen bag, BCKNN firstly finds its k references and q citers respectively, and then a Bayesian approach is applied to its k references and a distance weighted majority vote approach is applied to its q citers.

Therefore, our BCKNN uses Eq. 9 to classify y.

$$c_{BCKNN}(y) = \arg\operatorname*{Max}_{c\in C} \times \left(P(c)P(c_1, c_2, \ldots, c_k|c) + \sum_{j=1}^{q} w_j\delta(c,c_j) \middle/ \sum_{j=1}^{q} w_j\right) \tag{9}$$

where $\{c_1, c_2, \ldots, c_k\}$ and $\{c_1, c_2, \ldots, c_q\}$ are the class labels of k references and q citers of y respectively, and w_j is the weight of the *jth* citer.

To our knowledge, a large number of proposals can be used to define the weights w_j $(j = 1, 2, \ldots, q)$. Among them, the simplest proposal is to define their weights according to their distances to the unseen bag. That is to say, any functions which are inversely proportional to their distances can be used to define weights of these citers. Generally speaking, citers are weighted according to the inverse of their distances from the unseen bag, with less weight being assigned to citers that are further from the unseen bag. Thus, just for simplicity, we define three alternative simple weight functions as follows:

$$w_j = \frac{1.0}{1.0 + d_j} \tag{10}$$

$$w_j = \frac{1.0}{1.0 + d_j^2} \tag{11}$$

$$w_j = e^{-d_j^2}. \tag{12}$$

In Sect. 4, we design another group of experiments to compare three different distance weighting functions. Experimental results show that the distance weighting function defined by Eq. 11 is a little better than another two distance weighting functions defined by Eqs. 10 and 12. More detailed experimental results can be found from Table 5 in Sect. 4. More important, seen from Table 7 in Sect. 4, our BCKNN is generally better than BKNN and CKNN. Please also note that our new proposed BCKNN almost maintains the same order of computational overhead as previous CKNN. The experimental results from Tables 8 and 9 in Sect. 4 validate our conclusion.

4 Experimental conditions, methods, results

Because our proposed BCKNN is an improved algorithm of previous BKNN and CKNN, we conduct our compared experiments on the same datasets in the paper by Wang and Zucker [14]. We download them from the main website of Prof. E. Frank (http://www.cs.waikato.ac.nz/~eibe/multi_instance/). Each bag in them represents a molecule, and the main difference between these two datasets is that Musk2 contains molecules that have more possible conformations than Musk1. Thus, the learning task is to predict whether the molecule emits a musky odour. Some detailed descriptions of these two datasets are shown in Table 1.

We implement our improved algorithm BCKNN, KNN and the previous BKNN and use the implementation of CKNN in Weka 3.5.7 [31]. Please note that, in our implementation, the Laplace correction is used to smooth the estimated class membership probabilities.

Table 1 Descriptions of two musk datasets

Dataset	Instances	Attributes	Bags	Positives	Negatives
Musk1	476	166	92	47	45
Musk2	6,598	166	102	39	63

Table 2 Classification accuracy comparisons for KNN with different k values and different distance functions

Datasets	Distance functions	k = 1	k = 2	k = 3	k = 4	k = 5
Musk1	Minimum distance	85.78	90.11	84.78	84.78	80.33
	Maximum distance	66.22	66.22	68.33	68.44	68.44
	Centroid distance	84.56	84.56	82.44	83.67	79.22
	Hausdorff distance	80.44	85.78	83.67	84.78	73.67
Musk2	Minimum distance	76.27	81.18	78.18	81.27	75.36
	Maximum distance	69.36	69.45	69.27	67.27	65.36
	Centroid distance	73.55	79.36	78.27	73.27	74.27
	Hausdorff distance	78.27	81.18	73.18	74.18	71.09

In the first three groups of experiments, we respectively use KNN, BKNN, and CKNN with different k values to validate the effectiveness of different distance functions. In CKNN, we set c to $k + 2$, which is the same as the setting of the paper by Wang and Zucker [18]. Besides, in order to save the running time of experiments, we use the tenfold cross validation test instead of the leave-one-out test to predict classification accuracy of each algorithm on each datasets. Note that runs with various algorithms are carried out on the same training sets and evaluated on the same test sets. In particular, the cross-validation folds are the same for all the experiments on each dataset.

The detailed experimental results are shown in Tables 2, 3 and 4. From the experimental results, we can see that, generally speaking, the minimum distance seems to be the best among all four distance functions and when $k = 2$ the classification accuracies of related algorithms are the highest. This conclusion is consistent with that of the paper by Wang and Zucker [18].

In the fourth group of experiments, we use our proposed BCKNN with different weights to compare three different distance weighting functions. In BCKNN, we set k to 2 and c to $k + 2$, which is same to the setting of the paper by Wang and Zucker [18]. Besides, the minimum distance

Table 3 Classification accuracy comparisons for BKNN with different k values and different distance functions

Datasets	Distance functions	k = 1	k = 2	k = 3	k = 4	k = 5
Musk1	Minimum distance	85.78	90.11	89.11	84.78	86.89
	Maximum distance	66.22	66.22	66.44	67.33	67.22
	Centroid distance	84.56	83.44	82.44	83.67	80.33
	Hausdorff distance	80.44	85.78	80.44	83.67	81.44
Musk2	Minimum distance	76.27	81.18	78.18	80.27	76.36
	Maximum distance	69.36	69.45	64.27	67.27	71.27
	Centroid distance	73.55	79.36	71.45	74.45	71.27
	Hausdorff distance	78.27	81.18	75.27	74.18	72.36

Table 4 Classification accuracy comparisons for CKNN with different k values and different distance functions

Datasets	Distance functions	k = 1	k = 2	k = 3	k = 4	k = 5
Musk1	Minimum distance	87.00	89.11	88.11	84.00	84.89
	Maximum distance	67.11	67.22	67.33	68.44	69.56
	Centroid distance	85.67	87.89	86.89	84.78	83.56
	Hausdorff distance	83.56	84.67	85.89	83.56	82.44
Musk2	Minimum distance	81.18	82.36	88.09	85.27	80.27
	Maximum distance	69.27	70.36	70.36	68.27	67.36
	Centroid distance	82.09	81.36	81.36	80.27	75.27
	Hausdorff distance	79.27	79.18	76.27	74.18	76.27

defined by Eq. 3 is used. The detailed experimental results are shown in Table 5. From the experimental results, we can see that the distance weighting function defined by Eq. 11 is a little better than another two distance weighting functions defined by Eqs. 10 and 12.

Finally, we have designed another group of experiments to validate our improved algorithm BCKNN. Thus, we compared our BCKNN with previous BKNN and CKNN. In this group of experiments, we use the fine-tuned experience parameters from previous four groups of experiments: The values of k and c in related algorithms to 2 and 4 respectively. Besides, the minimum distance defined by Eq. 3 and the distance weighting function defined by Eq. 11 are used.

We add two other train direction prediction datasets (denoted by *eastwest* and *westeast*) and three mutagenicity prediction datasets (denoted by *muta-atoms, muta-bonds* and *muta-chains*) into this group of experiments. At the same time, we choose other two related multi-instance learning algorithms (MIBoost [25] and MIWrapper [26]) for competitors. We use the implementation of MIBoost and MIWrapper with the nearest neighbor algorithm as the basic classifier) in Weka 3.5.7 [31]. The detailed description of them is shown in Table 6.

Table 7 shows the detailed compared results in terms of classification accuracy. In the same way, we also observe their CPU training and test time (the averaged CPU time in seconds, all experiments are carried out on a PC with Microsoft Windows XP professional 2002 service pack 3 with Intel (R) Core (TM) 2 Quad CPU Q8400 (2.66 GHz) and 3 GB memory). The detailed experimental results are

Table 5 Classification accuracy comparisons for BCKNN with different weights

Datasets	No weights	Weighted by Eq. 10	Weighted by Eq. 11	Weighted by Eq. 12
Musk1	90.11	92.22	92.22	90.00
Musk2	82.27	82.18	83.18	84.18

Table 6 Descriptions of two train direction prediction datasets and three mutagenicity prediction datasets

Dataset	Instances	Attributes	Bags	Positives	Negatives
Eastwest	213	24	20	10	10
Westeast	213	24	20	10	10
Muta-atoms	1,618	10	188	125	63
Muta-bonds	3,995	16	188	125	63
Muta-chains	5,349	24	188	125	63

Table 7 Classification accuracy comparisons for BCKNN versus BKNN, CKNN, MIBoost, and MIWrapper

Datasets	BCKNN	BKNN	CKNN	MIBoost	MIWrapper
Musk1	92.22	90.11	89.11	84.56	85.67
Musk2	83.18	81.18	82.36	76.27	77.27
Eastwest	65	70	45	55	50
Westeast	65	65	45	40	50
Muta-atoms	75.44	69.59	75.96	84.04	83.51
Muta-bonds	72.78	67.46	74.39	82.46	82.46
Muta-chains	74.5	71.9	74.47	84.68	85.73
Average	75.45	73.61	69.47	72.43	73.52

Table 8 CPU Training time comparisons for BCKNN versus BKNN, CKNN, MIBoost, and MIWrapper

Datasets	BCKNN	BKNN	CKNN	MIBoost	MIWrapper
Musk1	0.64	0.64	0.68	0.37	0.06
Musk2	116.53	125.21	116.26	48.78	0.77
Eastwest	0.02	0.02	0.02	0.02	0
Westeast	0.02	0.02	0.02	0.04	0
Muta-atoms	0.55	0.55	0.56	2.09	0
Muta-bonds	4.85	4.85	4.85	17.02	0.01
Muta-chains	12.43	12.32	12.31	39.96	0.02
Average	19.291	20.516	19.243	15.47	0.12

Table 9 CPU test time comparisons for BCKNN versus BKNN, CKNN, MIBoost, and MIWrapper

Datasets	BCKNN	BKNN	CKNN	MIBoost	MIWrapper
Musk1	0.16	0.07	0.15	0.06	0.07
Musk2	27.09	14.36	27.73	8.09	8.13
Eastwest	0	0	0.01	0	0
Westeast	0	0	0	0	0.01
Muta-atoms	0.13	0.06	0.13	0.23	0.03
Muta-bonds	1.09	0.55	1.18	1.9	0.19
Muta-chains	2.75	1.38	2.89	4.53	0.46
Average	4.46	2.346	4.584	2.12	1.27

shown in Tables 8 and 9 respectively. From the experimental results, we can see that:

1. BCKNN (92.22 %) is better than previous BKNN (90.11 %) and CKNN (89.11 %) on Musk1. On Musk2, BCKNN (83.18 %) also outperforms BKNN (81.18 %) and CKNN (82.36 %). These results validate the effectiveness of our improved solution.
2. The CPU training and test time of BCKNN (19.291 and 4.46) is almost equal to previous CKNN (19.243 and 4.584). These results show that our BCKNN almost maintains the same order of computational overhead as previous CKNN.
3. Generally speaking, on two train direction prediction datasets, our BCKNN is much better than other two related competitors (MIBoost and MIWrapper) while this strength is reversed on three mutagenicity prediction datasets.

5 Conclusion and future work

In this paper, we revisit the well-known algorithms called Bayesian-KNN (BKNN) and Citation-KNN (CKNN). After the discussion of two practical problems in adapting KNN to multi-instance classification problems, we propose an improved algorithm called Bayesian Citation-KNN (BCKNN) by combining the Bayesian approach and the distance weighting approach. For each unseen bag, BCKNN firstly finds its k references and q citers respectively, and then a Bayesian approach is applied to its k references and a distance weighted majority vote approach is applied to its q citers. The experimental results on several benchmark datasets show that our BCKNN is generally better than previous BKNN and CKNN. Besides, BCKNN almost maintains the same order of computational overhead as CKNN.

The same as BKNN and CKNN, our BCKNN also uses standard Euclidean distance to define the difference between each pair of instances. Thus, using some new proposed distance measures, such as affinity-based distance function [32], to scale up the accuracy of our BCKNN is one of the main directions for our future work. Besides, in the current version of BCKNN, k references are treated equally in the Bayesian approach. Thus, how to apply the distance weighting approach to the references and get the distance weighted Bayesian approach is another main direction for our future work. Finally, investigating and researching the ranking performance of our BCKNN in terms of AUC [33–35] will also be included in our future work.

Acknowledgments We thank Prof. E. Frank and Miguel Garcia Torres for their kindly help. We also thank anonymous reviewers for their valuable comments and suggestions. This work was partially supported by the National Natural Science Foundation of China (No. 61203287), the Program for New Century Excellent Talents in University (No. NCET-12-0953), the Provincial Natural Science Foundation of Hubei (No. 2011CDA103), and the Fundamental Research Funds for the Central Universities.

References

1. Zhou ZH (2004) Multi-instance learning: a survey. Technical Report, AI Lab, Department of Computer Science and Technology, Nanjing University, Nanjing
2. Zhang ML, Zhou ZH (2007) ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognit 40:2038–2048
3. He J, Gu H, Wang Z (2012) Bayesian multi-instance multi-label learning using Gaussian process prior. Mach Learn 88(1–2):273–295
4. Zhou ZH, Zhang M-L, Huang S-J, Li Y-F (2012) Multi-instance multi-label learning. Artif Intell 176(1):2291–2320
5. Dietterich TG, Lathrop RH, Lozano-Perez T (1997) Solving the multiple instance problem with axis-parallel rectangles. Artif Intell 89(1–2):31–71
6. Ruffo G (2000) Learning single and multiple decision trees for security applications. PhD Dissertation, Department of Computer Science, University of Turin, Turin
7. Zhang Q, Goldman SA (2002) EM-DD: an improved multiple-instance learning technique. Adv Neural Inf Process Syst 14:1073–1080

8. De Raedt L (1998) Attribute-value learning versus inductive logic programming: the missing links. Lecture Notes Artif Intell 1446:1–8
9. Zucker JD, Chevaleyre Y (2001) Solving multiple-instance and multiple-part learning problems with decision trees and rule sets, application to the mutagenesis problem. Lecture Notes Artif Intell 2056:204–214
10. Wang C, Scott S, Zhang J, Tao Q, Fomenko D, Gladyshev V (2004) A study in modeling low-conservation protein superfamilies. Technical report, Department of Computer Science, University of Nebraska-Lincoln, Lincoln
11. Yang C, Lozano-Perez T (2000) Image database retrieval with multiple-instance learning techniques. In: Proceedings of the IEEE International Conference on Data Engineering, pp 233–243
12. Zhang Q, Goldman SA, Yu W, Fritts J (2002) Content-based image retrieval using multiple-instance learning. In: Proceedings of 19th International Conference on Machine Learning, pp 682–689
13. Maron O (1998) Learning from ambiguity. Department of Electrical and Computer Science, Massachusetts Institute of Technology, Cambridge
14. Andrews S, Tsochantaridis I, Hofmann T (2003) Support vector machines for multiple-instance learning. Adv Neural Inf Process Syst 15:561–568
15. Maron O, Ratan AL (1998) Multiple-instance learning for natural scene classification. In: Proceedings of 15th International Conference on Machine Learning, pp 341–349
16. Chen Y, Wang JZ (2004) Image categorization by learning and reasoning with regions. J Mach Learn Res 5:913–939
17. Aha DW (ed) (1997) Lazy learning. Kluwer Academic Publishers, Dordrecht
18. Wang J, Zucker J-D (2000) Solving the multiple-instance problem: a lazy learning approach. In: Proceedings of 17th International Conference on Machine Learning, pp 1119–1125
19. Auer P (1997) On learning from multi-instance examples: empirical evaluation of a theoretical approach. In: Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, pp 21–29
20. Maron O, Lozano-Perez T (1998) A framework for multiple-instance learning. In: Advances in Neural Information Processing Systems, vol 10. MIT Press, Cambridge
21. Chen Y, Bi J, Wang JZ (2006) MILES: multiple-instance learning via embedded instance selection. IEEE PAMI 28(12):1931–1947
22. Foulds JR, Frank E (2008) Revisiting multiple-instance learning via embedded instance selection. In: Proceedings of 21st Australasian Joint Conference on Artificial Intelligence. Springer, Auckland, pp 300–310
23. Blockeel H, De Raedt L (1998) Top-down induction of first order logical decision trees. Artif Intell 101:285–297
24. Xu X (2001) A nearest distribution approach to multiple-instance learning. Department of Computer Science, University of Waikato, Hamilton
25. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann Press, San Francisco, pp 148–156
26. Frank ET, Xu X (2003) Applying propositional learning algorithms to multi-instance data. Technical Report, Department of Computer Science, University of Waikato, Hamilton
27. Dhurandhar A, Dobra A (2012) Probabilistic characterization of nearest neighbor classifier. Int J Mach Learn Cybern. doi: 10.1007/s13042-012-0091-y
28. Peuquet DJ (1992) An algorithm for calculating minimum euclidean distance between two geographic features. Comput Geosci 18(8):989–1001
29. Edgar GA (1995) Measure, topology, and fractal geometry. 3rd print, Springer, Berlin
30. Chen X, Doihara T, Nasu M (1995) Spatial relations of distance between arbitrary object s in 2D/3D geographic spaces based on the hausdorff metric. LIESMARS'95, Wuhan
31. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco
32. Bhattacharya G, Ghosh K, Chowdhury AS (2012) An affinity-based new local distance function and similarity measure for kNN algorithm. Pattern Recognit Lett 33(3):356–363
33. Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310
34. Jiang L, Li C, Cai Z (2009) Learning decision tree for ranking. Knowl Inf Syst 20(1):123–135
35. Liang G, Zhu X, Zhang C (2012) The effect of varying levels of class distribution on bagging for different algorithms: an empirical study. Int J Mach Learn Cybern. doi:10.1007/s13042-012-0125-5