International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **5**, October **2014**

 ISSN 1349-4198
pp. 1861–1871

INSTANCE-BASED K-NEAREST NEIGHBOR ALGORITHM FOR MULTI-INSTANCE MULTI-LABEL LEARNING

Peng Peng, Xinshun Xu* and Xiaolin Wang

School of Computer Science and Technology
Shandong University
No. 1500, Shunhua Road, Jinan 250101, P. R. China
sdupengpeng@gmail.com; *Corresponding author: xuxinshun@sdu.edu.cn
xlwang@sdu.edu.cn

Received October 2013; revised March 2014

Abstract. *Multi-instance multi-label learning (MIML) is proposed to tackle the problem represented by a bag of instances and associated with multiple labels which appears in a wide range of real-world tasks. Transforming the MIML problem into an equivalent problem is a very popular way to solve such learning work. However, such transformation may lose useful information encoded in training examples. In this paper we propose a new lazy learning algorithm – Instance-Based K-Nearest Neighbor (IB-KNN) which transforms MIML to multi-label learning (MLL), but makes full use of information between instances. Specifically, unlike most existing KNN methods for MIML problem which use the distance between bags, IB-KNN uses the distance between instances to discover the neighbor instances. Then, the neighbor instances vote to generate the preliminary results. After that, the problem becomes a standard MLL problem, and KNN method is used again to make the final prediction. We test the proposed algorithm on the COREL image data set and compare it with classical KNN based methods. The results show that IB-KNN achieves better performance.*

Keywords: Multi-instance multi-label learning, Lazy learning, K-Nearest Neighbor, Image annotation

1. **Introduction.** In traditional supervised learning, an object is represented as a single instance and associated with a single label as shown in Figure 1(a). Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the set of class labels. Traditional supervised learning aims to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from a data set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in \mathcal{X}$ represents an instance and $y_i \in \mathcal{Y}$ the label of x_i. Traditional supervised learning has been successful in many areas; however, it may not be appropriate when the object contains a number of instances and is associated with a number of labels. For example, in image classification, an image usually contains several different patches, and each can be represented by a feature vector as an instance; moreover, an image can belong to many categories simultaneously. There is the similar scenario in text categorization task. For example, a document usually consists of multiple sections, each of which can be represented as an instance; in addition, a document usually belongs to multiple topics.

In order to solve the above tasks, multi-instance multi-label learning (MIML) framework [17] has been proposed recently. In MIML framework, each training example is represented as multiple instances and has multiple labels as shown in Figure 1(b). Obviously, a traditional supervised learning problem can be regarded as a problem degenerated from MIML problem. Therefore, some researchers have proposed methods by transforming an MIML learning task into a problem under other learning frameworks.

In this paper, we propose an Instance-Based K-Nearest Neighbor (IB-KNN) algorithm for MIML learning problem, which could make full use of the information of instances. In

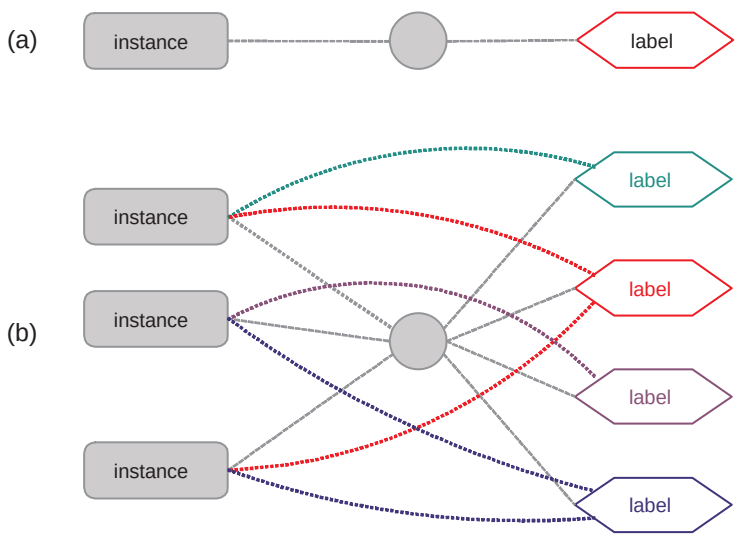


FIGURE 1. (a) Traditional supervised learning; (b) multi-instance multi-label learning

this method, we organize the neighbor information in a new way. Specifically, given a test example, IB-KNN firstly finds neighbors based on instances instead of bags in training data. After that, it generates counting vectors based on these neighbor instances, i.e., to count the number of neighbors of each label; then, in order to consider the "Minor Being Winner" problem in multi-instance learning task, we further proposed a Deep KNN step to process the counting vectors and get the final prediction. Experiments show that IB-KNN outperforms other existing bag level KNN learning algorithms. It has good potentials with application to many areas, e.g., automatic image annotation, image rank optimization, web annotation and other problems which can be represented as MIML learning task.

The rest of this paper is organized as follows. Section 2 reviews related works including multi-instance learning, multi-label learning and K-Nearest Neighbor based algorithms for such problems. Section 3 describes the details of IB-KNN approach. Section 4 gives the experimental results of IB-KNN and several compared state-of-the-arts. Finally, Section 5 concludes this paper.

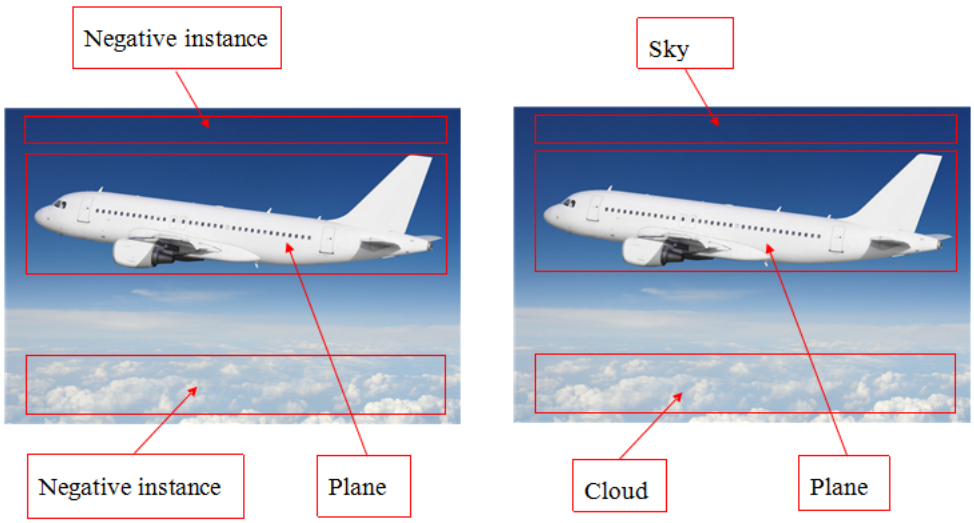
2. **Related Work.** Before MIML learning, multi-instance learning (MIL) and multi-label learning (MLL) have been proposed. MIL, proposed by Dietterich et al. [4], studies the problem where a real-world object described by a number of instances is associated with one class label. For multi-instance learning, many methods have been proposed. For instance, Maron and Tomás [9] proposed the DD algorithm which aims to detect a point with the maximum diverse density; Zhang and Goldman [13] proposed EM-DD algorithm which combines DD and EM algorithms together. A lazy learning method named Citation-KNN was proposed by Wang and Wang [11]. Citation-KNN is a variant of the K-Nearest Neighbor algorithm (KNN) which uses Hausdorff distance to measure the distance between bags and adds citation information to determine the label.

Multi-label learning tasks are also ubiquitous in real-world problems. In MLL, each sample is represented as a single instance associated with a set of labels. For this task, Zhang and Zhou [15] proposed ML-KNN method which is also derived from the classical KNN algorithm. This method firstly identifies the top K nearest neighbors of an instance in the training set and then gains statistical information based on the labels of these neighbor instances. For an unseen instance, the label set is determined by the

maximum a posteriori (MAP) principle. Besides the ML-KNN, BOOSTEXTER [10], ADTBOOST.MH [3], RANK-SVM [6] are also popular methods for the MLL tasks.

From the definitions of MIL and MLL, we can find that both of them can be treated as special cases of MIML. Thus, some methods have been proposed to tackle the MIML problem by degenerating it into MIL or MLL problem, e.g., MIMLBOOST and MIMLSVM [12]. In the first step of MIMLBOOST, the MIML task is transformed into MIL task by changing each MIML sample to several multi-instance bags with single-label. After that, it employs the MIBOOSTING algorithm [12] for the transformed MIL problem. MIMLSVM firstly converts each MIML sample into one instance with multi-label and uses MLSVM algorithm [2] to solve the transformed MLL task. Lazy learning is an important type of machine learning methods for many tasks, e.g., classification, and clustering [1]. Thus, lazy learning based methods have also been proposed to solve MIML problems. For instance, Jin et al. [7] proposed an algorithm to learn a distance metric from multi-instance multi-label data, and combined this algorithm with Citation-KNN for MIML. Specifically, this method optimizes the Mahalanobis distance between bags to minimize the distance between bags in the same classes, and maximizes the distance between bags from different classes. Based on the optimized distance, their method finally employs Citation-KNN to MIML task. In addition, inspired by the idea of Citation-KNN, Zhang [14] proposed MIML-KNN algorithm. Different from the Citation-KNN using minimal Hausdorff distance, this method employs average Hausdorff metric [16] to measure the distance. However, the computing of minimal or average Hausdorff distance may lose some useful information contained in instances. In order to take advantage of such information, in this paper, we propose the instance-level K-Nearest Neighbor algorithm for the multi-instance multi-label learning problem, which is detailed in the following section.

3. **Algorithm.** For MIML learning, given a data set $\mathcal{D} = \{(X_i, Y_i) \mid i = 1, 2, \ldots, M\}$ where $X_i \subseteq \mathcal{X}$ contains all instance vectors in the ith bag. Each $X_i = \{x_j^{(i)} \mid j = 1, 2, \ldots, N_i\}$, $x_j^{(i)} \in \mathcal{X}$. $Y_i \subseteq \mathcal{Y}$ contains the labels of the ith bag. Without loss of generality, Y_i can be treated as a column vector, e.g., $Y_i = (y_1^{(i)}, y_2^{(i)}, \ldots, y_L^{(i)})^T$ where



(a) Single-label image (b) Multi-label image

FIGURE 2. Multi-instance multi-label image has less negative instances than single-label image.

$y_l^{(i)} = 1$ indicating bag X_i is assigned to class c_l, otherwise $y_l^{(i)} = 0$. Here, N_i is the number of instances in X_i and L is the number of all possible labels. For a new bag $\widetilde{X}_t$, the task is to predict its label vector Y_t.

The traditional assumption for MIL is that the labels of key instances trigger the bag labels [8]. As shown in Figure 2(a) whose label is "plane", the key instance containing plane determines the image label, while other instances, e.g., those containing clouds and sky, are negative instances. Therefore, the Hausdorff distance between bags may be suitable. However, in Figure 2(b), when it is considered as a multi-label image, both cloud and sky become its labels. Here most of the instances are positive, so most of the instances will be beneficial to label prediction. Based on above observation, we can find that, for MIML problem, most instances contain useful information. Therefore, using instances as the basic voting unit may be better than using bags. Motivated by this, unlike some existing KNN methods which consider bag as the basic voting unit, our approach makes all instances in the bag participate in voting.

3.1. **Counting vector generation algorithm.** In IB-KNN, the neighbors of every instance are firstly found out; then these neighbors vote to generate the statistic vector of labels which we call it "counting vector". Thus, the counting vector contains statistic label information of all the nearest neighbors of the instances in the bag.

Specifically, for each instance x_i, the distance function between two instances is defined as:

$$Idis(x_i, x_j) = \|x_i - x_j\| \tag{1}$$

Assume that $x_j^{(m)}$ is one nearest neighbor of ${x_i}^{(n)}$ where m and n represent the identifier of the bag. Let Y_m be the label vector of the mth bag. For representation convenience, we use Y_{x_j} to represent the label vector of the bag that instance x_j belongs to. Let $\mathcal{K}_{x_i}^1$ be the set of nearest neighbors of ${x_i}^{(n)}$. Then, we define the following function $IKL(\cdot)$ to sum all the label vectors of bags that the neighbors of instance $x_i^{(n)}$ belong to:

$$IKL({x_i}^{(n)}) = \sum_{j=1}^{K_1} Y_{x_j} \tag{2}$$

$$x_j \in \mathcal{K}_{x_i}^1 \text{ and } \mathcal{K}_{x_i}^1 = \{x_j | j = 1, 2, \ldots, K_1\} \tag{3}$$

where K_1 is the number of neighbors of $x_i^{(n)}$.

To calculate the counting vector of a bag X_n, we firstly sum $IKL(\cdot)$ for all instances in X_n which means that every instance makes a contribution to the counting vector of the bag by its $IKL(\cdot)$. Let $\widehat{\delta}_X$ denote the counting vector which is an L-dimensional vector. Then, the counting vector of bag X_n is calculated by:

$$\widehat{\delta}_{X_n} = \sum_{i=1}^{N} IKL\left({x_i}^{(n)}\right) \tag{4}$$

To make the counting vector represent the percentage of votes for each possible label, we further normalize the counting vector as follows:

$$\delta_{X_n} = \frac{\widehat{\delta}_{X_n}}{\sum_l^L \left(\widehat{\delta}_{X_n}^l\right)} \tag{5}$$

3.2. **Deep KNN.** Based on the counting vector, we can make prediction for a bag. However, Paper [11] shows that there is a contradiction in MIL: Minor Being Winner (e.g., for unseen bags whose three nearest neighbors are two positive and one negative, if it is predicted as negative rather than positive, the overall prediction accuracy will be much higher). In traditional supervised learning, the contradiction will not happen. The contradiction in multiple-instance learning may be explained by the fact that positive bags contain both "true positive instances" and "false positive instances", and the latter may attract negative bags. Apparently, in MIML, such contradiction still exists. So the number of votes may not correctly reflect the probability of labels. In order to consider this problem, we firstly generate counting vectors for all training bags; then find the nearest neighbors from these counting vectors for every test bags. After that, we analyze the entropy of the neighbors' counting vectors to generate their weights. Based on this, we weaken the weight of the counting vector which may bring more noise and strengthen those more regular. Finally, these nearest neighbors vote again to decide the final prediction with different weights. We call this method "Deep KNN".

Let $\mathcal{D}$ be the training data set. For each $(X_i, Y_i) \in \mathcal{D}$, $i = 1, 2, \ldots, n$, we calculate the counting vector δ_{X_i} according to Equation (3). Then, these counting vectors constitute a new features set $\mathcal{D}_\Delta = \{\delta_{X_1}, \delta_{X_2}, \ldots, \delta_{X_n}\}$. The distance of two bags can be measured by calculating the Euclidean distance between the counting vectors:

$$Bdis(X_i, X_j) = \|\delta_{X_i} - \delta_{X_j}\| \tag{6}$$

Suppose that $\widetilde{X}_t$ is a new test bag and $\delta_{\widetilde{X}_t}$ is its counting vector. $\mathcal{K}^2_{\widetilde{X}_t}$ is further defined as a set which contains K_2 nearest neighbors of $\delta_{\widetilde{X}_t}$ in $\mathcal{D}_\Delta$. It can be represented as:

$$\mathcal{K}^2_{\widetilde{X}_t} = \{\delta_{X_j} | j = 1, 2, \ldots, K_2 \text{ and } \delta_{X_j} \in \mathcal{D}_\Delta\} \tag{7}$$

where δ_{X_j} is the jth nearest neighbor of $\delta_{\widetilde{X}_t}$.

It is well known that different neighbors usually make different contribution to predicting the labels of $\widetilde{X}_t$. Thus, it is expected that we can get more accurate results if we could get the weight of each neighbor according to its importance for predicting.

To get such weights, we introduce the entropy in information theory into the computing process, which can usually be used as a measure of the number of specific ways in which a system may be arranged, e.g., disorder; the higher the entropy is, the higher the disorder is. For a counting vector, if the votes are concentrated on several labels, its prediction will be more credible and it also means that it has a smaller entropy. For each δ_{X_j} in $\mathcal{K}^2_{\widetilde{X}_t}$, its entropy is calculated by:

$$H_{\delta_{X_j}} = -\frac{1}{\ln L} \sum_{l=1}^{L} \delta^l_{X_j} \ln(\delta^l_{X_j}), \ \delta_{X_j} \in \mathcal{K}^2_{\widetilde{X}_t} \tag{8}$$

where $\delta^l_{X_j}$ is the lth value of δ_{X_j}. Then, for $\widetilde{X}_t$, all the weights of its neighbors in $\mathcal{K}^2_{\widetilde{X}_t}$ can be represented as a vector:

$$W_{\widetilde{X}_t} = \left(w_{\delta_{X_1}}, w_{\delta_{X_2}}, \ldots, w_{\delta_{X_{K_2}}}\right) \tag{9}$$

where

$$w_{\delta_{X_j}} = \frac{1 - H_{\delta_{X_j}}}{K_2 - \sum_j^{K_2} H_{\delta_{X_j}}} \tag{10}$$

Let $\mathcal{L}^2_{\widetilde{X}_t}$ be the matrix in which each column is the label vector of its corresponding nearest neighbor, i.e.,

$$\mathcal{L}^2_{\widetilde{X}_t} = \left(Y_{X_1}, Y_{X_2}, \ldots, Y_{X_{K_2}}\right) \tag{11}$$

Once we have $\mathcal{L}^2_{\widetilde{X}_t}$ and $W_{\widetilde{X}_t}$, the label probability of $\widetilde{X}_t$ can be calculated by:

$$P(\widetilde{X}_t) = \mathcal{L}^2_{\widetilde{X}_t} * {W_{\widetilde{X}_t}}^{\mathrm{T}} \tag{12}$$

Based on this probability, we can further predict the labels of $\widetilde{X}_t$. For example, we can firstly get a threshold; then, the label will be assigned to $\widetilde{X}_t$ if its corresponding probability is larger than the threshold and vice versa.

4. **Experiments.** To show the performance of IB-KNN, we test it on COREL image data set. In our experiments, only the top 20 most popular keywords are considered because many keywords only annotate a few images. Consequently, we get a total of 3,947 training images and 444 test images. The features we use are the same as those proposed in [5].

4.1. **Evaluation metrics.** The performance is evaluated by five metrics [10], which are popular for multi-label models; specifically, they are *hamming loss, one-error, coverage, ranking loss and average precision*. For a given test MIML data set $\mathcal{D} = \{(X_i, Y_i) \mid 1 \leq i \leq p\}$, they are described as follows.

(1) ***Hamming loss:***

It evaluates the mean of how many times a label pair is misclassified.

$$hloss_D\left(h\right) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|\mathcal{Y}|}|h(X_i)\Delta Y_i| \tag{13}$$

where $h(X_i)$ returns the predicted label. The operator Δ calculates the number of difference between two sets. The smaller the value of $hloss_D\left(h\right)$ is, the better the performance is.

(2) ***One-error:***

It evaluates how many times of top-ranked label is annotated with wrong label.

$$one\text{-}error_D(h) = \frac{1}{p}\sum_{i=1}^{p}\langle[\arg\max_{y\in\mathcal{Y}} h(X_i, y)] \notin Y_i\rangle \tag{14}$$

where for predicating $\pi, \langle\pi\rangle$ equals 1 if π holds and 0 otherwise. The smaller the value of $one\text{-}error_D(h)$ is, the better the performance is.

(3) ***Coverage:***

It evaluates how far in rank list we need to coverage all true labels.

$$coverage_D(h) = \frac{1}{p}\sum_{i=1}^{p}\max_{y\in\mathcal{Y}} rank_h(X_i, y) - 1 \tag{15}$$

where $rank_h(X_i, y)$ is the predictable rank index of label y for example X_i. The smaller the value of $coverage_D(h)$ is, the better the performance is.

(4) ***Ranking loss:***

It evaluates the number of misordered label pairs.

$$rloss_D(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i||\overline{Y}_i|}\cdot|\mathcal{R}_i| \tag{16}$$

$\mathcal{R}_i = \{(y_1, y_2)|h(X_i, y1) \leq h(X, y_2), (y_1, y_2) \in Y_i \times \overline{Y}_i\}$, where $\overline{Y}_i$ is the complementary set of Y_i in $\mathcal{Y}$. The smaller the value of $rloss_D(h)$ is, the better the performance is and max value is 1.

(5) ***Average precision:***

It evaluates the average precision of labels ranked above a particular label $y \in Y_i$.

$$avgprec_D(h) = \frac{1}{p}\sum_{i=1}^{p} \times \sum_{y \in Y_i} \frac{|\mathcal{P}_i|}{rank_h(X_i, y)} \tag{17}$$

where $\mathcal{P}_i = \{\, |y'| \, rank_h(X_i, y') \leq rank_h(X_i, y),\ y' \in Y_i|\}$. The bigger the value of $avgprec_D(h)$ is, the better the performance is.

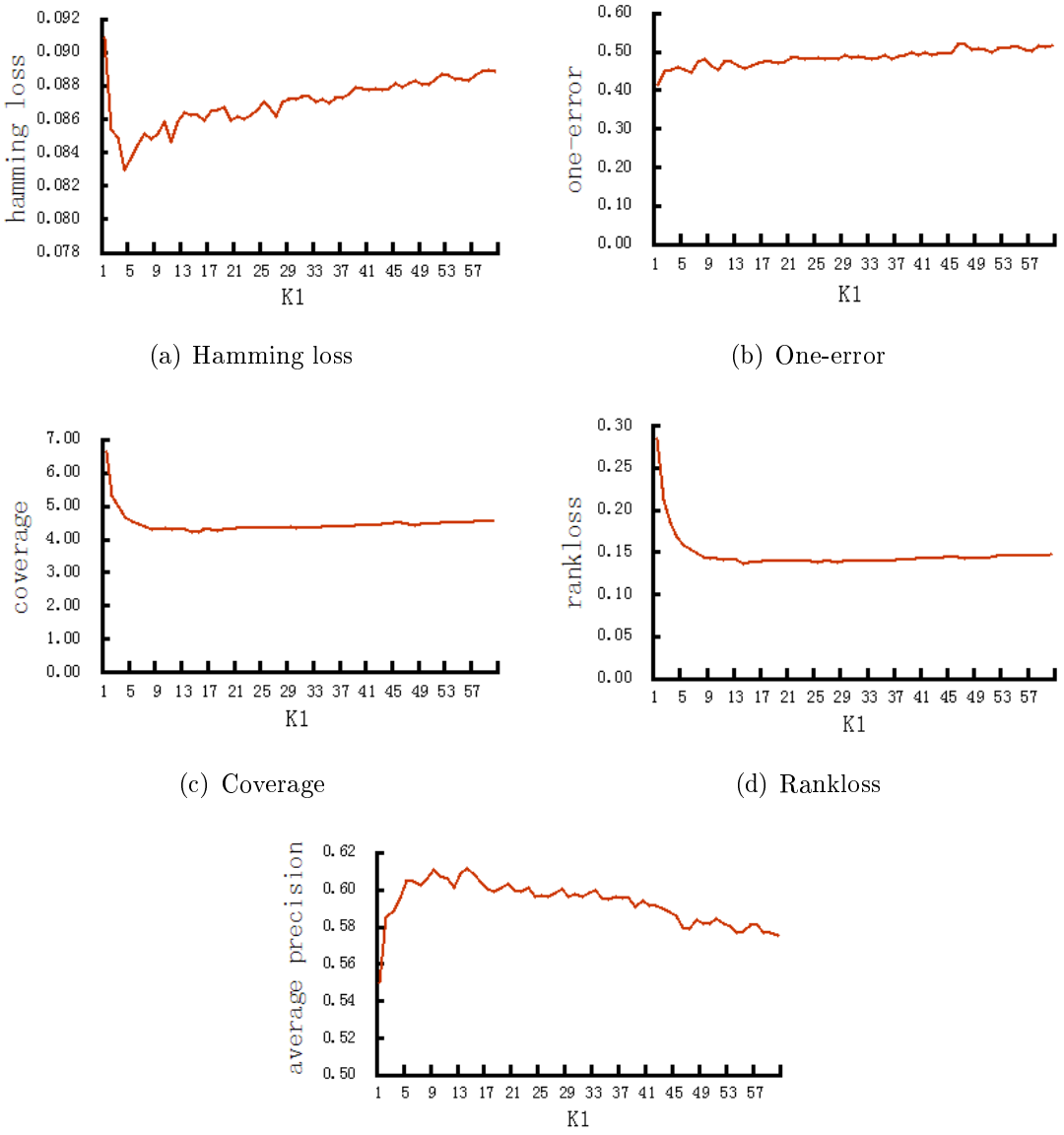
TABLE 1. Experimental results of IB-KNN without Deep KNN step and Citation-KNN methods on the COREL data set (Citation-KNN with ML means Citation-KNN with metric learning).

Evaluation metrics	Compared Algorithm		
	IB-KNN without Deep KNN	Citation-KNN with ML	Citation-KNN without ML
Hamming loss	**0.086**	N/A	N/A
One-error	**0.457**	0.570	0.633
Coverage	**4.221**	5.574	6.000
Ranking loss	**0.137**	N/A	N/A
Average precision	**0.611**	0.535	0.490

4.2. **Instance level and bag level comparison.** Most previous algorithms try to find neighbors on bag level. In order to show that instance level KNN based method has an inherent advantage over bag level KNN based method, in this section, we firstly compare IB-KNN without Deep KNN step with bag level KNN methods, i.e., Citation-KNN [11] and Citation-KNN with metric learning [7]. Citation-KNN is a classic bag level KNN method for multi-instance learning which uses not only neighbors information but also citation information; Citation-KNN with metric learning [7] introduces distance metric learning for optimization into the Citation-KNN. For fair comparison, we cite the result in [7] (in which only One-error Coverage and Average precision are considered, and the same features are used). The results are listed in Table 1, from which, we can see that even without the Deep KNN step IB-KNN obtains better results than the bag level methods.

Note that there is a parameter K_1 to be considered in IB-KNN without Deep KNN step. We use cross-validation to determine its best value $K_1 = 14$. To further investigate the impact of K_1 on the performance of IB-KNN, Figure 3 shows the result on COREL data set with different K_1. We can find that, for different evaluation metrics, the best result appears in the different K_1. In general, IB-KNN with $K_1 = 14$ obtains the best average precision on all metrics.

4.3. **Impact of Deep KNN.** To show the impact of Deep KNN step on the performance of IB-KNN, Table 2 shows the comparison results between IB-KNN with Deep KNN step and without Deep KNN step. In Section 4.2, we find that 14 is an appropriate value for K_1 when this is no Deep KNN step. However, with Deep KNN, there is the other parameter K_2 that needs to be considered. Thus, we have to find an appropriate combination of K_1 and K_2 so that the IB-KNN can obtain good performance. We also use the cross-validation to determine their values, and find that when $K_1 = 52$ and $K_2 = 38$, IB-KNN obtains the best results. In Table 2, the results of IB-KNN with another pair of parameters are also listed, i.e., $K_1 = 14$ and $K_2 = 43$. From this table, we can find that IB-KNN with Deep KNN step obtains better performance than that without Deep KNN step. This also confirms that IB-KNN benefits from the Deep KNN step.



(a) Hamming loss

(b) One-error

(c) Coverage

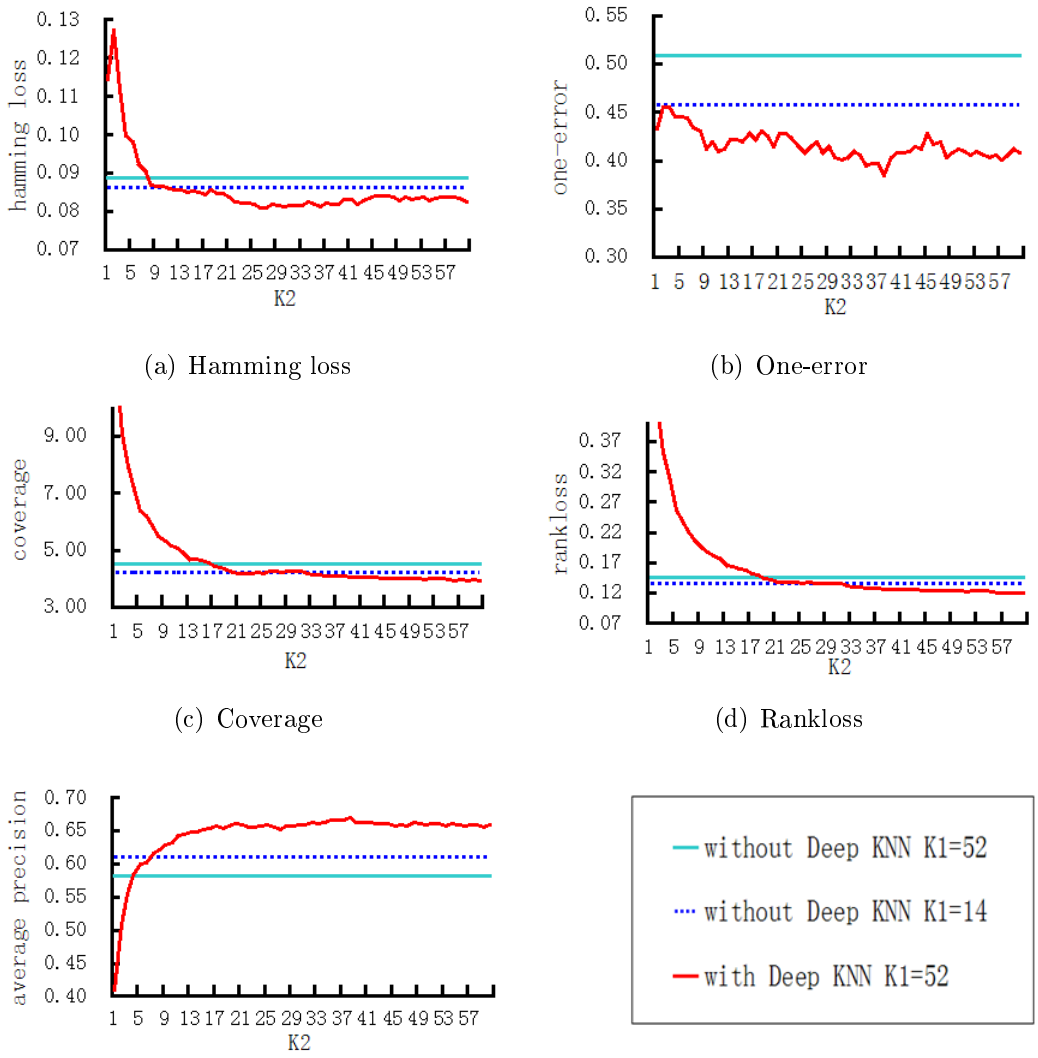
(d) Rankloss

(e) Average precision

FIGURE 3. The performance of IB-KNN without Deep KNN changes with the different value of K_1 on COREL.

TABLE 2. Experimental results of IB-KNN with and without Deep KNN on the COREL image data set.

Evaluation metrics	Compared Algorithms			
	IB-KNN with Deep KNN		IB-KNN without Deep KNN	
	$K_1 = 52$, $K_2 = 38$	$K_1 = 14$, $K_2 = 43$	$K_1 = 14$,	$K_1 = 52$
Hamming loss	**0.082**	0.084	0.086	0.089
One-error	**0.385**	0.430	0.457	0.509
Coverage	**4.088**	4.135	4.220	4.502
Ranking loss	**0.127**	0.128	0.137	0.145
Average precision	**0.669**	0.643	0.611	0.582



(a) Hamming loss

(b) One-error

(c) Coverage

(d) Rankloss

(e) Average precision

FIGURE 4. The performance of IB-KNN with Deep KNN changes with different values of K_2 and $K_1 = 52$ on COREL data set.

To further investigate the impact of K_2 on the performance, we show the results of IB-KNN with different values of K_2 when $K_1 = 52$ in Figure 4. From this figure, we can see that with the increasing of K_2, the values of different evaluation metrics (except one error) rapidly increase and become stable. When K_2 is in an appropriate range of values, IB-KNN with Deep KNN step always exceeds the best result of IB-KNN without Deep KNN step. This further shows that IB-KNN can benefit much from Deep KNN step.

In addition, we compare IB-KNN (with Deep KNN step) with other MIML methods, such as MIMLBOOST and MIMLSVM. The results are listed in Table 3. From this table, we can also find that IB-KNN gets the best performance on all of the five metrics.

In order to further show the performance of IB-KNN, in Figure 5 we list the top 8 images got by IB-KNN and traditional KNN. In this experiment, the traditional KNN uses Hausdorff distance between bags to find the neighbors of a bag. From this figure, we can see that, compared with traditional KNN, IB-KNN can find more meaningful neighbors for the test examples.

TABLE 3. Experimental results of different algorithms on the COREL image data set

Evaluation metrics	Compared Algorithm			
	IB-KNN	Citation-KNN	MIMLBOOST	MIMLSVM
Hamming loss	**0.082**	N/A	0.103	0.090
One-error	**0.385**	0.570	0.462	0.439
Coverage	**4.088**	5.574	7.130	5.975
Ranking loss	**0.127**	N/A	0.304	0.224
Average precision	**0.669**	0.535	0.590	0.614

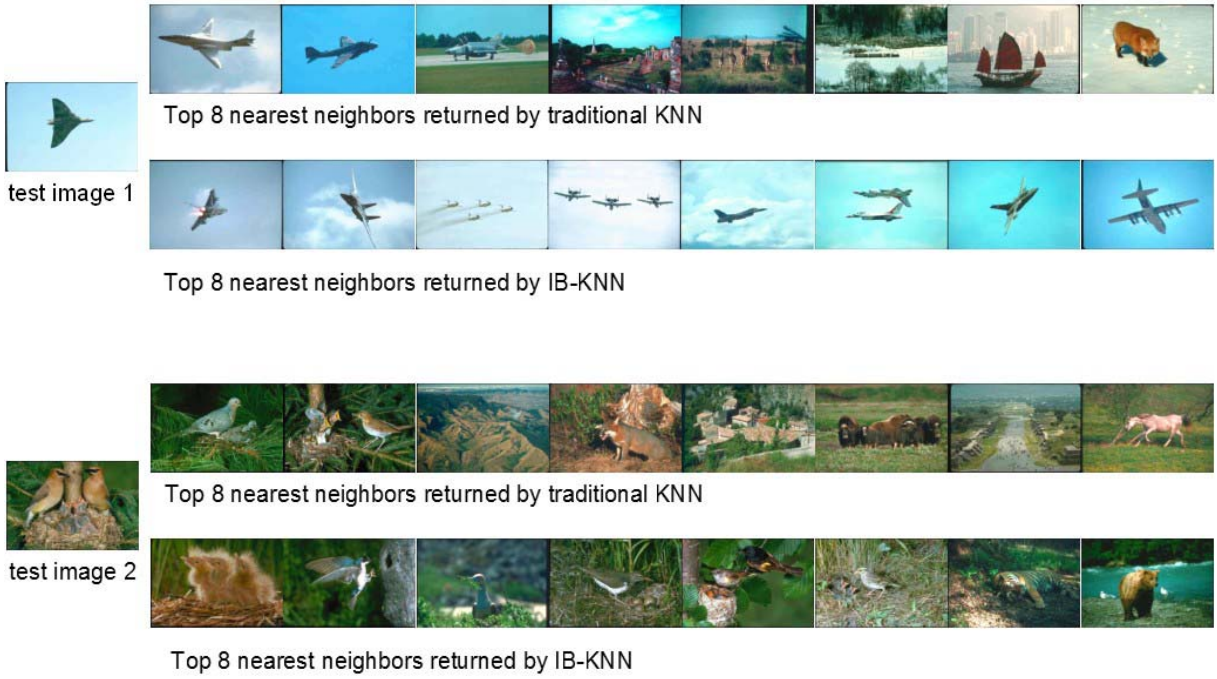


FIGURE 5. Top 8 images found by IB-KNN and traditional KNN

5. **Conclusions.** In this paper, we propose a lazy learning method IB-KNN for MIML problem. Different from some existing algorithms which use Hausdorff distance to find nearest neighbors of bags, IB-KNN uses Euclidean distance to find the nearest neighbors of instances. With the information of these instance nearest neighbors, we introduce the Deep KNN step into IB-KNN which can tackle the "Minor Being Winner" problem in multi-instance learning, to some extent. In this step, we use entropy theory to analyze the degree of confusion and assign different weights for counting vectors for revoting. From the experimental results on the real-world data set, we find that IB-KNN can obtain better results than some popular methods.

Acknowledgment. We would like to thank the reviewers for carefully reading our manuscript and giving detailed comments and suggestions that have been helpful to improve the manuscript. This research is partially supported by National Natural Science Foundation of China (61173068), Program for New Century Excellent Talents in University and DNSLAB, China Internet Network Information Center (K201206007).

REFERENCES

[1] I. Babaoglu, O. Findik, E. Ulker and N. Aygul, A novel hybrid classification method with particle swarm optimization and k-nearest neighbor algorithm for diagnosis of coronary artery disease using exercise stress test data, *International Journal of Innovative Computing, Information and Control*, vol.8, no.5(B), pp.3467-3475, 2012.

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

[3] F. De Comité, R. Gilleron and M. Tommasi, Learning multi-label alternating decision trees from texts and data, *Proc. of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp.35-49, 2003.

[4] T. G. Dietterich, R. H. Lathrop and T. Lozano-Pérez, Solving the multiple instance problem with axis-parallel rectangles, *Artificial Intelligence*, vol.89, no.1, pp.31-37, 1997.

[5] P. Duygulu, K. Barnard, J. F. de Freitas and D. A. Forsyth, Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary, *Proc. of European Conference on Computer Vision*, pp.349-354, 2002.

[6] A. Elisseeff and J. Weston, A kernel method for multi-labelled classification, *Advances in Neural Information Processing Systems 13*, pp.681-687, 2001.

[7] R. Jin, S. Wang and Z. H. Zhou, Learning a distance metric from multi-instance multi-label data, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp.896-902, 2009.

[8] G. Q. Liu, J. X. Wu and Z. H. Zhou, Key instance detection in multi-instance learning, *Proc. of Asian Conference on Machine Learning*, pp.1-16, 2012.

[9] O. Maron and L.-P. Tomás, A framework for multiple-instance learning, *Advances in Neural Information Processing Systems 10*, pp.570-576, 1998.

[10] E. S. Robert and S. Yoram, BoosTexter: A boosting-based system for text categorization, *Machine Learning*, vol.39, no.2, pp.135-168, 2000.

[11] J. Wang and J. D. Wang, Solving multi-instance problem: A lazy learning approach, *Proc. of International Conference on Machine Learning*, pp.1119-1125, 2000.

[12] X. Xu and E. Frank, Logistic regression and boosting for labeled bags of instances, *Proc. of Pacific Asia Conference on Knowledge Discovery and Data Mining*, pp.272-281, 2004.

[13] Q. Zhang and S. A. Goldman, EM-DD: An improved multiple-instance learning technique, *Advances in Neural Information Processing Systems 14*, pp.1073-1080, 2002.

[14] M. L. Zhang, A k-nearest neighbor based multi-instance multi-label learning algorithm, *Proc. of IEEE International Conference on Tools with Artificial Intelligence*, vol.2, pp.207-212, 2010.

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

[16] M. L. Zhang and Z. H. Zhou, Multi-instance clustering with applications to multi-instance prediction, *Applied Intelligence*, vol.31, no.1, pp.47-68, 2009.

[17] Z. H. Zhou and M. L. Zhang, Multi-instance multi-label learning with application to scene classification, *Advances in Neural Information Processing Systems 14*, pp.1609-1616, 2007.