Key Instance Detection in Multi-Instance Learning

Guoqing Liu LIUGQ@NTU.EDU.SG
Jianxin Wu JXWU@NTU.EDU.SG
School of Computer Engineering, Nanyang Technological Univeristy, Singapore 639798

Zhi-Hua Zhou ZHOUZH@NJU.EDU.CN
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

The goal of traditional multi-instance learning (MIL) is to predict the labels of the bags, whereas in many real applications, it is desirable to get the instance labels, especially the labels of key instances that trigger the bag labels, in addition to getting bag labels. Such a problem has been largely unexplored before. In this paper, we formulate the Key Instance Detection (KID) problem, and propose a voting framework (VF) solution to KID. The key of VF is to exploit the relationship among instances, represented by a citer kNN graph. This graph is different from commonly used nearest neighbor graphs, but is suitable for KID. Experiments validate the effectiveness of VF for KID. Additionally, VF also outperforms state-of-the-art MIL approaches on the performance of bag label prediction.

Keywords: Key instance detection, multi-instance learning, neighborhood relation, voting framework, iterative rejection

1. Introduction

Multi-instance learning (MIL) is a bag-level supervised learning task. The training data is a set of labeled bags, and each bag contains several instances. Formally, we have a dataset $\mathbb{D} = \{(B_i, y_i)\}_{i=1}^{n}$, where $B_i = \{I_j\}_{j=1}^{n_i}$, $y_i \in \{+1, -1\}$ is the label of B_i, n is the number of training bags, n_i is the number of instances in B_i. Each $I_j \in \mathcal{I}$ is an instance, where $\mathcal{I}$ denotes the instance space. In MIL problems, $y_i = +1$ if and only if at least one I_j in B_i is a positive instance of the underlying concept; otherwise, $y_i = -1$. Let B^+ (B^-) denote a positive (negative) bag. The task of classic MIL is to train a classifier that labels new bags, and has already been widely applied in various areas, such as text categorization (Settles et al., 2008), image classification (Chen et al., 2006), and computer-aided medical diagnosis (Fung et al., 2007).

Many real problems, however, ask for much more than bag labeling alone. Rather, *positive instances are expected to be identified*. For example, stock selection was studied by (Maron and Lozano-Pérez, 1998), trying to distinguish three types of stocks: those who perform well for fundamental reasons (positive instances), who perform well because of flukes (negative instances in positive bags), and those who underperform (instances in negative bags). It is obviously desirable if we can label instances, which will explicitly distinguish positive instances (stocks who perform well for fundamental reasons) from all the rest negative instances. In this paper, we study the Key Instance Detection (KID) problem,

whose aim is to detect positive instances (or equivalently, label all instances instead of only bags.) We call the positive instances as *key instances*, to emphasize the fact that they are more important than negative instances in various applications.

Computer vision is another potential application area of the KID problem. Suppose we want to learn a classifier to detect the concept *bride*, and resort to the photo sharing service Flickr for training data. It is easy to find tons of images with the tag *wedding*, but much fewer for brides. If we take image sets tagged with *wedding* as an MIL bag, it is certain that there should be at least one image in a bag with the bride (i.e., a key instance) in it. A high quality KID solver will then automatically find key instances (*bride*), which removes the costly manually labeling step, a key obstacle in current computer vision research and systems.

Key instance detection, however, has not been paid enough attention in the literature. The KID problem is more challenging than bag classification, since we can easily label a bag once all the key instances have been detected. In several MIL methods, instance labels are provided only as the *by-products* while learning bag classification models. EM-DD (Zhang and Goldman, 2002), miSVM (Andrews et al., 2003) and RW-SVM (Wang et al., 2006) are typical examples in this category. These methods usually assign labels to training instances, which will lead to maximized bag classification accuracy. However, so long as one key instance is correctly identified in a bag, labels of other key instances will not change the bag classification accuracy. Thus, these methods are naturally biased to make the prediction for "the most positive instance in a bag" to be correct, instead of emphasizing on correct labels for all (key) instances. The emphasis on *single most positive instance* is explicitly adopted by some MIL methods, for example, in RW-SVM, CkNN-ROI (Zhou et al., 2005b) and KI-SVM (Li et al., 2009). CkNN-ROI (Zhou et al., 2005b) and KI-SVM (Li et al., 2009) were proposed to locate regions of interest (ROI) in image analysis. Empirical study showed that KI-SVM is successful in identifying ROI, that is, the most positive instance in a bag. This emphasis, however, is different from the KID problem, which focuses on finding all key instances. The lack of attention to the KID problem also leads to the rarity of MIL datasets which are accompanied with groundtruth labels for all instances.

In this paper, we directly attack the key instance detection problem, that is, to label all instances instead of bags. We first study the feasibility of KID, and propose a voting framework for it in Section 2, which is based on exploring neighborhood relations among instances. Specifically, we show that a *citer kNN graph*, different from commonly used nearest neighbor relationship graphs, should be used in KID. Two major components in the voting framework, a random walk based voting confidence function and an iterative rejection algorithm that learns a vote vector, are presented in Section 3 and 4, respectively. Algorithmic details are presented in Section 5. Empirical results (Section 6) show that our voting framework has clear advantages in key instance detection, compared to existing MIL methods. As a direct application of KID, we use inferred instance labels to train models for labeling bags, which also outperforms state-of-the-art MIL methods.

For the convenience of presentation, we summarize the notations used in this paper in Table 1.

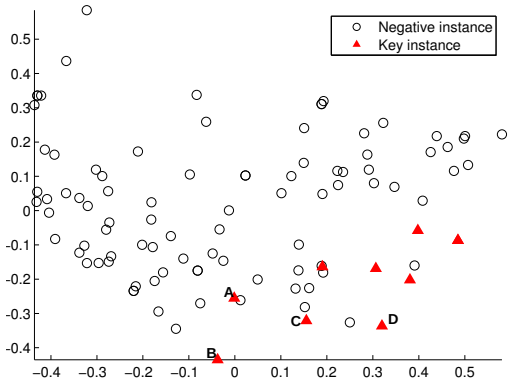

Figure 1: Multi-dimensional scaling (MDS) 2D visualization of the first 100 instances in the COAST dataset, from two bags. MDS approximately preserves pairwise distances (Borg and Groenen, 2005).

Table 1: The Summary of Notations

B_i	THE i-TH TRAINING BAG	y_i	LABEL OF B_i
I_j	THE j-TH TRAINING INSTANCE	I_q	A TESTING INSTANCE
I_j^h	THE h-TH CITER OF I_j	$\mathbb{D}$	THE SET $\{(B_i, y_i)\}$
$\mathbb{I}$	THE SET $\{I_j\}$	$\mathcal{I}$	INSTANCE SPACE
$\mathcal{R}$	THE SET OF REAL NUMBERS	$\mathcal{G}$	CITER KNN GRAPH
kNN_j	THE SET OF k NEAREST NEIGHBORS OF I_j	d_j	THE NUMBER OF CITERS OF I_j
$f(I_j)$	CONFIDENCE OF I_j	$\mathbf{v}$	VOTE VECTOR
$c(I_j)$	VOTING SCORE OF I_j	$\mathbf{v}_B$	BAG LABEL VECTOR
$(\mathbf{C})_{tr}$	TRAINING CONFIDENCE MATRIX	$\mathbf{P}$	TRANSITION MATRIX OF $\mathcal{G}$

2. Feasibility and Solution Framework

It is natural to ask: is KID really feasible? In MIL, only bag labels are provided, beyond the training bags and instances. It seems ill-posed to estimate the instance labels using the bag labels alone.

2.1. KID Feasibility

KID, however, may become feasible if we exploit the relationships among training instances. As shown in Fig. 1, the nearest neighbors of negative instances are mostly negative; and, the probability that a key instance's neighborhood contains other key instances is much higher than the probability that a key instance appearing in the vicinity of a negative one. Thus, *exploiting the neighborhood relationship among instances and the bag labels, it is possible to separate key and negative instances apart.* For a new instance I_q, a simple majority vote within its nearest neighbors in the training set may already lead to a reasonable instance prediction for I_q, as the case in Fig. 1.

It is, however, not trivial to choose a proper neighborhood relation. Different types of neighborhood relation can give quite different performances in a specific task. Maier and Luxburg (2009); Jebara et al. (2009) study this issue on graph-based clustering and semi-supervised learning, respectively. In KID problems, we observed that commonly used neighborhood relations are not suitable. For example, the commonly used ϵ-graph will classify many key instances in Fig. 1 as negative. We argue that *a citer kNN graph*, which we define below, will increase the feasibility of the KID problem.

Let $\mathbb{I} = \{I_j\}_{j=1}^{N}$ denote the set of all training instances, where $N = \sum_{i=1}^{n} n_i$ is the total number of instances in the training set. A neighborhood graph $\mathcal{G}$ has $\mathbb{I}$ as its vertex set, and there is an edge between I_i and I_j if and only if they are related to each other in the neighborhood relation. Commonly used graphs are ϵ-graph and kNN graph (Zhu, 2007; Zhou et al., 2005a). In an ϵ-graph, an undirected edge connects I_i and I_j when $\|I_i - I_j\| \leq \epsilon$. In a kNN graph, a directed edge from I_i to I_j exists when $I_j \in kNN_i$, where kNN_i denotes the set of k nearest instances of I_i within $\mathbb{I}$. Following (Wang and Zucker, 2000), we can name this kNN graph as referencer kNN graph. If we reverse all the directed edges in a referencer kNN graph, we call the resulted graph a *citer kNN graph*, which is suitable for key instance detection.

One common property in MIL is that the number of negative instances (N_-) is much larger than the number of key instances (N_+). Assuming equal number of positive and negative bags and equal size in each bag, it is obvious that $N_+ \leq N_-$ from the definition of MIL. In particular, $N_+ \ll N_-$ when the witness rate is low.[1] Thus, the density of negative instances is much higher than that of the key instances (cf. Fig. 1), and the average distance between negative instances are correspondingly smaller than that between key instances. This asymmetric property makes choosing a distance threshold ϵ very difficult. That is, ϵ-graph is unsuitable for KID. The same property also prefers citer kNN graph to referencer kNN graph. Take the key instance 'A' in Fig. 1 as an example, it relates only to negative instances in a referencer 5-NN graph, since its 5 nearest neighbors are all negative. However, key instance 'B' is related to 'A' in the citer 5-NN graph since 'A' is among the 5-NN of 'B'. Similar observations apply to key instance pair 'C' and 'D'. Because of the disparity in density between negative and positive instances, the citer kNN graph makes it easier to detect key instances.

2.2. Limitations and Future Directions

When the neighborhood relationships are not reliable (e.g., neighbors of most key instances are negative), we expect that even the bag classification task will be difficult. In this case, we do not know whether the KID problem is feasible or not. In this paper, the focus is to promote the KID problem, and to show that a voting framework can successfully detect key instances in many MIL problems. We leave to future work the theoretical analyses of KID's feasibility and performance bounds under different assumptions.

1. Witness rate (WR) is the ratio of the number of positive instances in a positive bag to the size of the bag. Many MIL problems in the real world naturally have positive bags with low witness rate (Zhou et al., 2009; Zhang and Goldman, 2002).

2.3. The Voting Framework

Following this motivation, we propose a voting framework to capture both the citer neighborhood relationship and the bag labels to identify key instances. Let $I_q \in \mathcal{I}$ be an arbitrary instance, the task of KID is to learn $l(I_q)$, where $l : \mathcal{I} \to \{+1, -1\}$ determines the label of an instance. We formulate KID as a voting process: every training instance $I_j \in \mathbb{I}$, based on its own label $l(I_j)$, votes 1 or -1 to a test instance I_q, and $l(I_q)$ is determined by these votes and their confidences. Mathematically, we set $l(I_q) = \text{sign}\,(c(I_q))$, where $c(I_q)$ is the *voting score* of I_q, determined as

$$c(I_q) = f(I_q)^T \mathbf{v}. \tag{1}$$

$\mathbf{v} \in \{1, -1\}^N$ is a vote vector. Each component in $\mathbf{v}$ corresponds to one training instance in $\mathbb{I}$, and $v_j = l(I_j)$, i.e., a positive (negative) training instance will vote 1 (-1), respectively. $f(I_q)$ is the *confidence function*, whose component $f(I_q)_j$ measures the confidence of I_j's vote.

Both $\mathbf{v}$ and $f(I_q)$ play important roles in this framework. On one hand, as shown in Fig. 1, the voting confidence must vary a lot based on the distance between I_j and I_q. The confidence function is a measure to quantify the relationship strength between I_q and I_j. We will formulate $f(I_q)$ based on the citer kNN graph, to be presented in Section 3. On the other hand, the ideal vote vector contains labels for all training instances, which is not available in MIL. We propose an iterative rejection technique in Section 4, so that $\mathbf{v}$ can be estimated from the training data, and subsequently applied to new test instances.

The proposed voting framework can be treated as a type of weighted nearest neighbor (NN) approach. However, two characteristics distinguish it from the classic NN: first, the confidence/weight is globally generated from the proposed citer kNN graph, which is a good representation of neighborhood relation among instances; second, the vote vector is learnt based on the confidence function cooperating with bag labels.

3. Random Walk Confidence Function

Given the training instances $\mathbb{I}$ and the citer kNN graph $\mathcal{G}$, a straightforward way to compute $f(I_q)$ is

$$f(I_q)_j = \begin{cases} 1/d_q & \text{if } I_q \in kNN_j \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where d_q is the number of citers of I_q. However, it is derived by mining only the local structure around I_q. Considering the disparity in density between negative and positive instances, we use random walk to compute $f(I_q)$ globally. A random walk starts from I_q, and consists of a trajectory in $\mathcal{G}$, by taking successive random steps. Compared to the local mining in Eq. 2, random walk on $\mathcal{G}$ makes it possible that a key instance I_q communicates with the other key instances out of its own neighborhood relation. Such global computation of $f(I_q)$ is more suitable for handling the disparity in KID.

The limiting probability that an infinitely dedicated random walk visits an instance I_j is a good choice to globally measure the strength of neighborhood relation between I_j and I_q. A potential way is to formulate $f(I_q)$ based on the transition matrix $\mathbf{P}$ corresponding to $\mathcal{G}$. Let the *training* confidence matrix $\mathbf{C}_{tr} \in \mathcal{R}^{N \times N}$ be defined as $\mathbf{C}_{tr} \equiv$

$(f(I_1), f(I_2), \ldots, f(I_N))^T$, we capture the global structure among instances as

$$\mathbf{C}_{tr} = \mathbf{I} + \mathbf{P} + \mathbf{P}^2 + \cdots = (\mathbf{I} - \mathbf{P})^{-1}. \tag{3}$$

Eq. 3 gives a closed form to compute the confidences of training instances. The limiting probability (i.e., the inverse of matrix in Eq. 3) does not always exist, because the transition matrix is not symmetric. Even if it exists, the computational complexity is $\mathcal{O}(N^3)$, and is inefficient with a large N. More importantly, when I_q is a testing instance, this method cannot be directly applied.

Alternatively, we use a histogram built from the random walk trajectory to compute the confidence. When $I_q \in \mathbb{I}$, i.e., it is a vertex of $\mathcal{G}$, we directly start a random walk from I_q on $\mathcal{G}$. After t steps, we obtain a *random walk histogram* to characterize the behaviors of this random walk. Specifically, a random walk histogram has N cells, with the j-th bin corresponding to the training instance I_j. It counts how many times each I_j is in the trajectory. After normalization, $f(I_q)_j$ is defined to be equal to the j-th bin in the histogram.

When $I_q \notin \mathbb{I}$, it is not in the citer kNN graph $\mathcal{G}$. In this case, we compute $f(I_q)$ with the help of its citers. First, we associate a threshold θ_j with each training instance I_j, equal to the maximum distance between I_j and its k nearest neighbors. That is, $\theta_j = \max \|I_j - x\|, x \in kNN_j$. For any $I_q \notin \mathbb{I}$, we consider an $I_j \in \mathbb{I}$ as a citer of I_q, if and only if $\|I_j - I_q\| \leq \theta_j$. Then, let $\{I_q^i\}_{i=1}^{d_q} \subseteq \mathbb{I}$ denote the set of citers of I_q in the training instances, we start a random walk at each citer on $\mathcal{G}$ independently. So d_q random walk histograms with t steps are generated. Finally, we compute the random walk histogram for I_q by averaging these d_q citer histograms. Equivalently, when $\mathbf{C}_{tr}$ has been obtained, we directly compute the confidence function by $f(I_q) = \frac{1}{d_q}\sum_{i=1}^{d_q} f(I_q^i)$, where each $f(I_q^i)$ can be looked up in $\mathbf{C}_{tr}$.

This random walk based framework has a probabilistic interpretation. Let X be a discrete random variable in the sample space $\mathbb{I}$. Each instance generates a distribution, characterized by the global structure among the training instances. Formally, for any $I_q \in \mathcal{I}$,

$$\Pr(X = I_j) = f(I_q)_j, \quad j = 1, \ldots, N. \tag{4}$$

Furthermore, we define a *label function* $g : \mathbb{I} \to \{+1, -1\}$, which is a function of the random variable X, corresponding to the instance labels, that is, $v_j = g(X = I_j)$. Then, the voting mechanism Eq. 1 can be rewritten as

$$c(I_q) = \sum_{j=1}^{N} g(I_j)\Pr(X = I_j) = E(g(X)). \tag{5}$$

Hence, the voting score $c(I_q)$ is the expectation of the label function of X following the distribution generated by I_q.

4. Learning the Vote Vector

The next step is to learn or estimate the vote vector $\mathbf{v}$. $\mathbf{v}$ is unknown, and the only relevant inputs are the bag labels. Let us define a bag label vector $\mathbf{v}_B \in \{1, -1\}^N$ for the training

instances, where $\mathbf{v}_{Bj}$ is 1 if $I_j \in \mathbb{I}$ is in a positive bag, and -1 if otherwise. We will then estimate $\mathbf{v}$, using $\mathbf{v}_B$ as its initial value, and iteratively refine the estimation using the neighborhood relation explored in $\mathbf{C}_{tr}$.

The initial values $\mathbf{v}_B$ is useful for its own sake. We first study the behaviors of

$$\tilde{c}(I_j) = (\mathbf{C}_{tr})_{j:}\mathbf{v}_B, \tag{6}$$

in positive and negative training bags, respectively, where $I_j \in \mathbb{I}$, and $(\mathbf{C}_{tr})_{j:}$ denotes the j-th row of the training confidence matrix. This term, $\tilde{c}(I_j)$, can be used as a basis for finding out negative instances in the positive training bags, and filtering out borderline instances in the negative training bags. Thus, we use $\tilde{c}(I_j)$ and an iterative procedure to estimate the $\mathbf{v}$.

Negative Instances in Positive Bags. If a training instance I_j satisfies both $I_j \in B^+$ and $c(I_j) \leq 0$, the voting framework should classify it as a negative instance. Although $c(I_j)$ is unknown, an upper bound is provided by $\tilde{c}(I_j)$. Specifically, it is easy to check that $\mathbf{v}_B \succeq \mathbf{v}$, and consequently $\mathbf{C}_{tr}\mathbf{v}_B \succeq \mathbf{C}_{tr}\mathbf{v}$. Hence, for any $I_j \in \mathbb{I}$, $\tilde{c}(I_j) \leq 0$ ensures that $c(I_j) \leq 0$. Using this relationship, we can use $\tilde{c}(I_j)$ to change such instances' labels in $\mathbf{v}$ from $+1$ to -1.

Algorithm 1 Iterative Rejection on Training Bags

Input: training instances $\mathbb{I}$ and $\mathbf{v}_B$, the number of neighbors k, the number of random walk steps t
Let $\tilde{\mathbb{I}} = \mathbb{I}$, $\tilde{\mathbf{v}} = \mathbf{v}_B$
repeat
  Initialize $noChange = true$.
  Let $N = |\tilde{\mathbb{I}}|$
  Construct the citer kNN graph $\mathcal{G}$ from $\tilde{\mathbb{I}}$
  for $j = 1$ **to** N **do**
    Generate t-steps random walk histogram of $I_j \in \tilde{\mathbb{I}}$, and compute $(\mathbf{C}_{tr})_{j:}$
    Compute $\tilde{c}(I_j) = (\mathbf{C}_{tr})_{j:}\tilde{\mathbf{v}}$
    if $\tilde{v}_j = +1$ and $\tilde{c}(I_j) \leq 0$ **then**
      Drop instance I_j out of $\tilde{\mathbb{I}}$
      Delete the j-th row of $\tilde{\mathbf{v}}$
      $noChange = false$
    end if
    if $\tilde{v}_j = -1$ and $\tilde{c}(I_j) > 0$ **then**
      Drop instance I_j out of $\tilde{\mathbb{I}}$
      Delete the j-th row of $\tilde{\mathbf{v}}$
      $noChange = false$
    end if
  end for
until $noChange$ is $true$

Borderline Instances in Negative Bags. In real MIL problems, the distributions of positive and negative instances are usually overlapped (cf. Fig. 1). Some negative bags may contain instances that are closely related to key instances in $\mathcal{G}$. Such relationships are harmful to KID. In the proposed voting framework, a mathematical definition of borderline instances within a negative bag B^- is naturally given, based on $\tilde{c}(\cdot)$, as: *I_j is a borderline instance, if $I_j \in B^-$ and $\tilde{c}(I_j) > 0$*. Although we should not change the borderline instances' label from -1 to $+1$ in $\mathbf{v}$, removing such instances from the graph $\mathcal{G}$ will be helpful in finding key instances in the positive bags.

Iterative Rejection. With the help of $\mathbf{v}_B$ and $\tilde{c}(I_j)$, we can then estimate $\mathbf{v}$, which is equivalent to identifying key instances in positive training bags. Based on the above analyses, an Iterative Rejection (IR) algorithm is proposed.

The IR algorithm is presented in details in Algorithm 1. It begins with the training data $\tilde{\mathbb{I}} = \mathbb{I}$. After constructing the citer kNN graph $\mathcal{G}$ from $\tilde{\mathbb{I}}$, $\mathbf{C}_{tr}$ is computed from the t-steps random walk histograms of training instances. For each training instance I_j, $\tilde{c}(I_j)$ is obtained following Eq. 6. Following the discussions in this section, lines 10 to 14 and 15 to 19 detect negative instances in each B^+ and borderline instances in each B^-, respectively. These instances are removed from $\tilde{\mathbb{I}}$. We repeat the rejection process until $\tilde{\mathbb{I}}$ is not changed, i.e., all instances of positive (negative) bags in $\tilde{\mathbb{I}}$ have $\tilde{c}(I_j) > 0$ ($\tilde{c}(I_j) \leq 0$). After convergence, an instance in $\mathbb{I}$ is positive, if and only if it is contained by a positive bag in $\tilde{\mathbb{I}}$. The vote vector $\mathbf{v}$ can be equivalently obtained from $\tilde{\mathbf{v}}$.

5. Implementation

To further improve the efficiency of KID, an ensemble is introduced in the proposed framework. Training bags are randomly partitioned into s disjoint subsets $\mathbb{I}_i$, $i = 1, \ldots, s$, and $\bigcup_i \mathbb{I}_i = \mathbb{I}$. Using each $\mathbb{I}_i$ as the training instance set, we obtain a voting score $c_i(\cdot)$ following Algorithm 1. The final decision $c(I_q)$ is defined as

$$c(I_q) = \min\left(c_1(I_q), \ldots, c_s(I_q)\right), \tag{7}$$

that is, an instance I_q is labeled to be positive, if and only if it is positive in all s detections. We initialize $s = 1$, and gradually increase s in the set $\{1, 2, 5, 10\}$. If a larger s does not reduce the number of detected key instances in the training set by more than h, the larger s value is used. We aim to obtain the best efficiency, while without hurting the KID accuracy. For the threshold h, we always $h = 20\%$ according to the experiences in our experiments.

In Algorithm 2, the proposed voting framework of KID, referred to as VF, is briefly summarized, where we give different approaches to label training and testing instances.

Labeling Training Instances with Voting Score. As described in Lines 3-10 in Algorithm 2, we label training instances (i.e., learn the vote vector) based on the voting framework. During ensembling (c.f. Lines 5-9 in Algorithm 2), we compute the voting score of $I_q \in \mathbb{I}_{-j}$ from Eq. 1, where $\mathbb{I}_{-j} = \bigcup_{i \neq j} \mathbb{I}_i$. Specifically, $f(I_q)$ is derived in the way presented in Section 3, and

$$c(I_q) = \frac{1}{d_q} \sum_{i=1}^{d_q} f(I_q^i)^T \mathbf{v} = \frac{1}{d_q} \sum_{i=1}^{d_q} c(I_q^i). \tag{8}$$

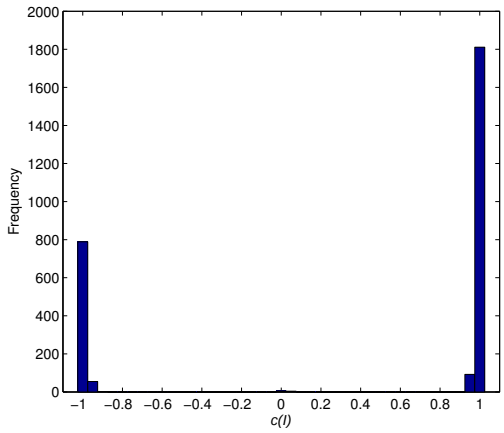

Figure 2: The histogram of $c(I)$ in the first iteration of the IR algorithm on the COMP.GRAPHICS dataset.

When t uses a large value (e.g., 1000), we empirically find that for $I \in \mathbb{I}$, $c(I) \approx \text{sign}(c(I))$. The absolute values of $c(I)$ almost always concentrate around 1 in the training set. Fig. 2 shows a histogram of $c(I)$ on the COMP.GRAPHICS dataset in Section 6. Hence, Eq. 8 is approximated by $c(I_q) = \frac{1}{d_q}\sum_{i=1}^{d_q} \text{sign}(c(I_q^i)) = \frac{1}{d_q}\sum_{i=1}^{d_q} v_i$, in which v_i is the i-th component in the estimated vote vector $\mathbf{v}$. This approach further improves efficiency of VF.

Labeling Testing Instances with SVM. Till now, the voting framework only exploits the neighborhood relationships and bag labels. After obtaining instance label predictions in the training set, supervised learning methods can be used to learn a model that distinguishes key and negative instances. The final step in our voting framework converts the training data $\mathbb{D}$ to a set of instances $\mathbb{I}$ labeled by the vote vector. Then KID becomes a standard supervised learning. We train a RBF kernel SVM classifier (c.f. Line 11 in Algorithm 2), and finally label testing instances using the learned SVM model.

Algorithm 2 The Voting Framework for KID

1: **Input:** training bags $\mathbb{D} = \{(B_i, y_i)\}_{i=1}^n$, the ensemble parameter s
2: **Training:**
3: Obtain training instances set $\mathbb{I}$ and $\mathbf{v}_B$ based on $\mathbb{D}$
4: Randomly partition $\mathbb{I}$ into s disjoint subsets $\mathbb{I}_j$, $j = 1, \ldots, s$; correspondingly, each $\mathbf{v}_B^j$ is segmented from $\mathbf{v}_B$ based on $\mathbb{I}_j$.
5: **for** $j = 1$ **to** s **do**
6: Run Algorithm 1 with input $\mathbb{I}_j$ and $\mathbf{v}_B^j$
7: Label $\mathbb{I}_{-j}$ using Eq. 8's approximation
8: Generate $c_j(\cdot)$
9: **end for**
10: Generate $c(\cdot)$ using Eq. 7, and further obtain $\mathbf{v}$
11: Train a SVM instance classifier using $\mathbb{I}$, and $\mathbf{v}$

6. Experimental Results

We validate the proposed VF method on two series of KID datasets. One previous MIL dataset with groundtruth instance labels is the Reuters text, constructed in (Zhou et al., 2009). In addition, we constructed a new KID dataset series for scene recognition, which is generated from images of the MIT scene dataset (Oliva and Torralba, 2001). The data contains images of 8 scene types, from which we constructed 8 MIL problems. Given a specific scene type, one MIL problem treats images from this type as key instances, and all other images as negative ones. A negative bag has 50 instances; a positive bag has 5 key instances and 45 negative ones. For each scene type, we created a MIL dataset with 50 positive and 50 negative bags. Every instance is a CENTRIST representation of an image (Wu and Rehg, 2011).

6.1. Experiments on Key Instance Detection

As the witness rate of KID task is usually very low (for example, 10% on every dataset in our experiments), classifying instances is imbalanced. In such imbalanced setup, accuracy as the evaluation of KID becomes ill-posed, while precision and recall based measures are preferred in literature of imbalanced classification (He, 2009). Furthermore, KID wants to detect a large fraction of key instances, and only a small number of false positives. Hence, it is natural to use the *precision* and *recall* metrics,

$$precision = tp/(tp + fp), \tag{9}$$

$$recall = tp/(tp + fn), \tag{10}$$

where tp, fp and fn are the number of true positives (key instances), false positives, and false negatives in the instance level, respectively.

We compare with miSVM and KI-SVM, all with the RBF kernel. The parameter $C \in \{2^{-15}, 2^{-13}, \ldots, 2^{15}\}$ and RBF parameter $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ are selected by 3-fold cross-validation on the training data. In VF, k is selected from $\{2, 10, 30, 50\}$, and t is fixed to be 1000. In addition, we also compare VF (i.e., using the citer kNN graph) with VF using the referencer kNN graph (VFr).

Tables 2 and 3 show the average *precision* and *recall* of each method using 10-fold cross-validation on MIT scene and Reuters text datasets, respectively.

As shown in Table 2, KI-SVM gives higher precision than miSVM, but its recall is too low. This might be explained by the fact that KI-SVM is designed to find the *single most positive instance* in a bag, and our observations confirm this argument that KI-SVM can provide comparable accuracies on the "most positive instances" with the proposed VF. Differently, VF is designed to do its best in keeping all key instances. Indeed, it has very high recall rate (0.82). Furthermore, its precision (0.85) is also much higher than both miSVM and KI-SVM.

In Table 3, miSVM outperforms KI-SVM on Reuters text data: both precision and recall of miSVM (0.55 and 0.72) are higher than KI-SVM (0.44 and 0.31). Similarly, VF provides the best results again: although it is comparable with miSVM on recall, its precision is improved by a large margin, from 0.55 to 0.75.

Table 2 and 3 also report the results from VFr, which has the same setup as VF except using the referencer kNN graph to mine neighborhood relations. As we have analyzed in

Table 2: Key instance detection performances of various methods on MIT scene data.

Dataset	miSVM		KI-SVM	
Name	Precision	Recall	Precision	Recall
coast	0.51±0.12	0.69±0.11	0.70±0.24	0.28±0.17
forest	0.87±0.07	0.89±0.07	0.65±0.20	0.36±0.12
highway	0.96±0.06	0.62±0.08	0.79±0.18	0.31±0.18
insidecity	0.68±0.15	0.71±0.08	0.71±0.17	0.24±0.11
mountain	0.56±0.21	0.74±0.11	0.81±0.19	0.37±0.12
opencountry	0.28±0.15	0.53±0.13	0.52±0.27	0.12±0.08
street	0.51±0.17	0.57±0.16	0.52±0.41	0.05±0.04
tallbuilding	0.66±0.11	0.76±0.09	0.83±0.11	0.46±0.16
Ave. value	0.63±0.13	0.69±0.10	0.69±0.22	0.28±0.12
Dataset	VF		VFr	
Name	Precision	Recall	Precision	Recall
coast	0.88±0.08	0.82±0.08	0.88±0.08	0.82±0.08
forest	0.95±0.06	0.88±0.07	0.93±0.06	0.86±0.05
highway	0.90±0.08	0.73±0.10	0.88±0.08	0.77±0.07
insidecity	0.92±0.12	0.75±0.12	0.91±0.12	0.79±0.14
mountain	0.75±0.17	0.91±0.06	0.75±0.16	0.82±0.06
opencountry	0.77±0.12	0.79±0.09	0.75±0.10	0.75±0.08
street	0.76±0.17	0.76±0.10	0.75±0.16	0.81±0.06
tallbuilding	0.87±0.10	0.91±0.06	0.87±0.10	0.80±0.10
Ave. value	0.85±0.11	0.82±0.09	0.84±0.11	0.80±0.08

Section 2.1, the asymmetric property makes the proposed citer kNN graph more suitable for KID tasks. VFr gives inferior performance over VF on both datasets. Especially, on Reuters text data, the average precision and recall of VFr are 0.65 and 0.32, which are much lower than 0.75 and 0.72 of VF, respectively. For ϵ-graph, when we set $\epsilon = 0.001$ on the alt.atheism data of Reuters text, 94.12% of key instances have no neighbor, while 84.21% of negative instances have more than 2500 neighbors. Obviously, this makes the ϵ-graph unsuitable for KID.

The F-score is a harmonic mean of precision and recall:

$$F - \text{score} = \frac{precision \times recall}{(1-\alpha) \times precision + \alpha \times recall}, \tag{11}$$

where $\alpha \in [0, 1]$ controls the relative importance of recall and precision. It is a succinct summary of a method's KID performance. In Table 4, we report the average F-score with $\alpha = 0.7$ of various methods on both datasets. $\alpha = 0.7$ emphasizes recall more than precision, which is suitable for evaluating key instance detection. On MIT scene, the F-score of VF is 0.83, much higher than that of miSVM (0.67), and KI-SVM (0.34). Similarly, VF also outperforms miSVM and KI-SVM on Reuters text. Besides, VF gives higher F-scores than VFr on both datasets.

Table 3: Key instance detection performances of various methods on Reuters text data.

Dataset	miSVM		KI-SVM	
Name	Precision	Recall	Precision	Recall
alt.atheism	0.53±0.19	0.76±0.24	0.37±0.26	0.36±0.19
comp.graphics	0.61±0.28	0.74±0.11	0.38±0.26	0.37±0.19
comp.os.ms-windows.misc	0.55±0.33	0.58±0.21	0.39±0.37	0.14±0.15
comp.sys.ibm.pc	0.62±0.22	0.59±0.20	0.39±0.31	0.19±0.17
comp.sys.mac.hardware	0.78±0.17	0.62±0.22	0.32±0.33	0.12±0.12
comp.windows.x	0.55±0.28	0.74±0.18	0.40±0.30	0.14±0.16
misc.forsale	0.59±0.22	0.55±0.18	0.03±0.05	0.08±0.12
rec.autos	0.43±0.18	0.74±0.21	0.39±0.25	0.21±0.19
rec.motorcycles	0.40±0.30	0.76±0.19	0.71±0.41	0.34±0.20
rec.sport.baseball	0.46±0.21	0.80±0.14	0.63±0.26	0.34±0.11
rec.sport.hockey	0.45±0.31	0.84±0.21	0.83±0.32	0.34±0.18
sci.crypt	0.63±0.31	0.71±0.22	0.36±0.18	0.37±0.20
sci.electronics	0.95±0.08	0.85±0.13	0.39±0.26	0.30±0.20
sci.med	0.56±0.11	0.78±0.18	0.57±0.20	0.41±0.21
sci.space	0.37±0.27	0.76±0.16	0.30±0.21	0.33±0.22
soc.religion.christian	0.34±0.25	0.75±0.18	0.39±0.17	0.39±0.19
talk.politics.guns	0.52±0.31	0.61±0.26	0.36±0.22	0.31±0.16
talk.politics.mideast	0.73±0.28	0.78±0.15	0.66±0.28	0.47±0.24
talk.politics.misc	0.65±0.18	0.62±0.23	0.54±0.19	0.60±0.21
talk.religion.misc	0.30±0.23	0.74±0.25	0.38±0.25	0.30±0.20
Ave. value	0.55±0.24	0.72±0.19	0.44±0.25	0.31±0.18
Dataset	VF		VFr	
Name	Precision	Recall	Precision	Recall
alt.atheism	0.73±0.24	0.60±0.24	0.58±0.43	0.32±0.30
comp.graphics	0.66±0.12	0.74±0.14	0.95±0.16	0.40±0.18
comp.os.ms-windows.misc	0.79±0.33	0.73±0.16	0.55±0.50	0.12±0.13
comp.sys.ibm.pc	0.85±0.16	0.67±0.22	0.68±0.47	0.27±0.23
comp.sys.mac.hardware	0.78±0.19	0.65±0.20	0.70±0.48	0.35±0.31
comp.windows.x	0.69±0.20	0.70±0.18	0.27±0.44	0.13±0.31
misc.forsale	0.66±0.26	0.66±0.14	0.85±0.34	0.29±0.17
rec.autos	0.78±0.16	0.71±0.19	0.79±0.15	0.56±0.26
rec.motorcycles	0.70±0.27	0.70±0.21	0.42±0.40	0.18±0.24
rec.sport.baseball	0.73±0.16	0.69±0.14	0.84±0.23	0.39±0.17
rec.sport.hockey	0.79±0.20	0.85±0.16	0.83±0.31	0.52±0.29
sci.crypt	0.79±0.23	0.77±0.20	0.40±0.52	0.13±0.19
sci.electronics	0.96±0.08	0.83±0.13	0.90±0.17	0.51±0.22
sci.med	0.73±0.10	0.74±0.24	0.54±0.38	0.38±0.27
sci.space	0.86±0.12	0.75±0.16	0.76±0.33	0.36±0.31
soc.religion.christian	0.84±0.18	0.62±0.14	0.64±0.30	0.42±0.20
talk.politics.guns	0.53±0.18	0.73±0.21	0.66±0.46	0.26±0.28
talk.politics.mideast	0.72±0.27	0.66±0.26	0.58±0.44	0.21±0.21
talk.politics.misc	0.72±0.20	0.75±0.22	0.65±0.40	0.42±0.34
talk.religion.misc	0.67±0.22	0.78±0.18	0.49±0.48	0.22±0.20
Ave. value	0.75±0.19	0.72±0.19	0.65±0.37	0.32±0.24

Table 4: Average F-score ($\alpha = 0.7$) of various methods.

Dataset	miSVM	KI-SVM	VF	VFr
MIT scene	0.67	0.34	0.84	0.83
Reuters text	0.65	0.34	0.73	0.50

AUC-PR (Area Under Precision-Recall Curve) (Davis and Goadrich, 2006) measures the precision-recall performances in the entire parameter range. We also evaluate KID methods using this metric. Table 5 reports the average AUC-PR. VF gives the highest AUC-PR (0.87 and 0.67) on scene and text data, respectively. One interesting observation is that although KI-SVM has the lowest average F-score, it has higher AUC-PR than miSVM on both datasets .

Table 5: Average AUC-PR of various methods.

Dataset	miSVM	KI-SVM	VF	VFr
MIT scene	0.55	0.71	0.87	0.85
Reuters text	0.41	0.42	0.67	0.59

Finally, Table 6 shows the average running time (in seconds) used by each method. We can see that VF is much faster than both miSVM and KI-SVM. Thus, VF has better scalability.

Table 6: Average running time of various methods.

Dataset	miSVM	KI-SVM	VF
MIT scene	381.31	>1000	211.89
Reuters text	61.43	>1000	33.27

6.2. Experiments on Bag Classification

An accurate instance label prediction method must lead to good bag classification. Thus, the performance on classifying bags should also be used as an important measurements of key instance detection performance. Once VF obtains labels for instances in testing bags, we can use the MIL definition to determine labels for test bags. In this section, we compare VF with state-of-the-art MIL bag classification methods.

In our experiments, miSVM and miGraph (Zhou et al., 2009) are used as baselines. Table 7 reports the average accuracies of various methods on MIT scene. VF greatly improves the bag accuracy to 0.95 from 0.81 of miSVM and 0.77 of miGraph. Similar results can be observed in Table 8, where the accuracies of bag classification on the Reuters text data are presented. The average accuracy of VF is 0.83, outperforms both miSVM (0.75) and miGraph (0.81). VF's superior performance demonstrates a direct application of key instance detection: a good key instance detection method is able to improve bag classification (that is, multi-instance learning).

Table 7: Bag classification performances of various methods on MIT scene data.

Dataset	miSVM	miGraph	VF
coast	0.80±0.16	0.89±0.09	0.98±0.12
forest	0.92±0.12	0.89±0.10	0.95±0.09
highway	0.95±0.13	0.80±0.15	0.97±0.10
insidecity	0.85±0.21	0.73±0.18	0.98±0.11
mountain	0.82±0.16	0.72±0.13	0.95±0.16
opencountry	0.59±0.11	0.64±0.13	0.93±0.14
street	0.76±0.15	0.61±0.13	0.87±0.14
tallbuilding	0.81±0.17	0.88±0.11	0.98±0.12
Ave. value	0.81±0.15	0.77±0.13	0.95±0.12

7. Conclusion

In this paper, we promote the study of key instance detection (KID), a problem that has impact in many application domains. We proposed a voting framework (VF) as our solution to KID. VF mines relations among all instances to form a citer kNN graph, and use them to define confidences of votes of training instances. The bag labels are used within an iterative rejection algorithm to determine what vote ($+1$ or -1) should be casted. The effectiveness and efficiency of VF have been demonstrated by its superior performance in predicting instance labels. Empirical results also confirm that a good key instance detection method is able to improve bag classification. In the future, we will further analyze the feasibility of KID in adverse scenarios, its performance bounds, and improved KID solutions. It is also critical to create more MIL datasets with groundtruth instance labels.

Acknowledgments

This research was partially supported by the National Fundamental Research Program of China (2010CB327903) and the National Science Foundation of China (61073097, 61105043).

Table 8: Bag classification performances of various methods on Reuters text data.

data	miSVM	miGraph	VF
alt.atheism	0.77	0.83	0.85
comp.graphics	0.80	0.83	0.84
comp.os.ms-windows.misc	0.73	0.69	0.72
comp.sys.ibm.pc	0.71	0.78	0.79
comp.sys.mac.hardware	0.81	0.77	0.83
comp.windows.x	0.76	0.81	0.88
misc.forsale	0.80	0.71	0.79
rec.autos	0.75	0.82	0.81
rec.motorcycles	0.72	0.78	0.88
rec.sport.baseball	0.74	0.88	0.88
rec.sport.hockey	0.70	0.93	0.91
sci.crypt	0.75	0.79	0.79
sci.electronics	0.94	0.94	0.94
sci.med	0.77	0.84	0.85
sci.space	0.68	0.83	0.85
soc.religion.christian	0.66	0.80	0.84
talk.politics.guns	0.72	0.75	0.77
talk.politics.mideast	0.77	0.84	0.79
talk.politics.misc	0.77	0.74	0.80
talk.religion.misc	0.64	0.78	0.78
Ave. value	0.75	0.81	0.83

References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 557–584, Cambridge, MA, 2003. MIT Press.

I. Borg and P. Groenen. *Modern Multidimensional Scaling: theory and applications*. Springer-Verlag, New York, 2005.

Y. Chen, J. Bi, and J. Z. Wang. MILES: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12): 1931–1947, 2006.

J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *the 23rd International Conference on Machine Learning*, pages 233–240, Cambridge, MA, 2006. MIT Press.

G. Fung, M. Dundar, B. Krishnappuram, and R. B. Rao. Multiple instance learning for computer aided diagnosis. In *Advances in Neural Information Processing Systems 19*, pages 425–432, Cambridge, MA, 2007. MIT Press.

H. He. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *the 26th International Conference on Machine Learning*, pages 441–448, 2009.

Y.-F. Li, J.T. Kwok, I.W. Tsang, and Z.-H. Zhou. A convex method for locating regions of interest with multi-instance learning. In *the 20th European Conference Machine Learning*, pages 15–30, 2009.

M. Maier and U. Luxburg. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems 20*, pages 1025–1032, Cambridge, MA, 2009. MIT Press.

O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*, pages 570–576, Cambridge, MA, 1998. MIT Press.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

B. Settles, M. Craven, and S. Ray. Multiple instance active learning. In *Advances in Neural Information Processing Systems 20*, pages 1289–1296, Cambridge, MA, 2008. MIT Press.

D. Wang, J. Li, and B. Zhang. Multiple-instance learning via random walk. In *the 17th European Conference Machine Learningg*, pages 473–484, 2006.

J. Wang and J.-D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *the 17th International Conference on Machine Learning*, pages 1119–1125, 2000.

J. Wu and J.M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1489–1501, 2011.

Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 14*, pages 1073–1080, Cambridge, MA, 2002. MIT Press.

D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems 17*, pages 1633–1640, Cambridge, MA, 2005a. MIT Press.

Z.-H. Zhou, X.-B. Xue, and Y. Jiang. Locating regions of interest in cbir with multiinstance learning techniques. In *the 18th Australian Joint Conference on Artificial Intelligence*, pages 92–101, 2005b.

Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li. Multi-instance learning by treating instances as non-i.i.d. samples. In *the 26th International Conference on Machine Learning*, pages 1249–1256, 2009.

X. Zhu. Semi-supervised learning tutorial. Technical report, Department of Computer Sciences University of Wisconsin, Madison, USA, 2007.