MILD: Multiple-Instance Learning via Disambiguation

Wu-Jun Li, Dit-Yan Yeung

Abstract

In multiple-instance learning (MIL), an individual example is called an *instance* and a *bag* contains a single or multiple instances. The class labels available in the training set are associated with bags rather than instances. A bag is labeled positive if at least one of its instances is positive; otherwise, the bag is labeled negative. Since a positive bag may contain some negative instances in addition to one or more positive instances, the true labels for the instances in a positive bag may or may not be the same as the corresponding bag label and, consequently, the instance labels are inherently ambiguous. In this paper, we propose a very efficient and robust MIL method, called MILD (Multiple-Instance Learning via Disambiguation), for general MIL problems. First, we propose a novel disambiguation method to identify the true positive instances in the positive bags. Second, we propose two feature representation schemes, one for instance-level classification and the other for bag-level classification, to convert the MIL problem into a standard single-instance learning (SIL) problem that can be solved by well-known SIL algorithms, such as support vector machine. Third, an inductive semi-supervised learning method is proposed for MIL. We evaluate our methods extensively on several challenging MIL applications to demonstrate their promising efficiency, robustness and accuracy.

Index Terms

Multiple-instance learning, learning from ambiguity, CBIR, object recognition, co-training, drug activity prediction

Wu-Jun Li and Dit-Yan Yeung are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Hong, China. E-mail: liwujun@cse.ust.hk; dyyeung@cse.ust.hk

Manuscript received xxxx xx, 2008; revised xxxx xx, xxxx.

1 INTRODUCTION

1.1 Multiple-Instance Learning

The task of drug activity prediction [1] is to classify aromatic molecules according to whether or not they are "musky". Here, the *same* molecule can manifest several different steric configurations (i.e., molecular shapes), each with very different energy properties. A molecule is said to be musky if it binds itself to a particular receptor when it is in *one or some* of its configurations, although it cannot bind itself to the receptor in its other configurations. When a molecule cannot bind itself to the receptor in any of its configurations, it is said to be non-musky.

A new learning paradigm called *multiple-instance learning* (MIL) was proposed in [1] to model learning problems like drug activity prediction. In an MIL problem [1], an individual example is called an *instance* and a *bag* contains a single or multiple instances. A bag is labeled positive if at least one of its instances is positive; otherwise, the bag is labeled negative. In the example of drug activity prediction, a bag corresponds to a molecule, and an instance corresponds to a configuration. A configuration is said to be positive if it can make the molecule bind to the receptor. Otherwise, it is called negative. In MIL, the class labels available in the training set are associated with bags rather than instances. For example, we only know whether or not a molecule is musky, but do not know in what configuration a molecule can bind itself to the receptor.

As more and more applications have been formulated as MIL problems, some of them are slightly different from the original formulation of MIL in [1]. Recent works [2], [3] have pointed out that there are actually two different settings for MIL. One setting is based on the *existential assumption*, which assumes that a bag can be determined to be positive as long as *one single positive instance* exists in it. This setting is the same as the original formulation in [1]. The other setting is based on the *collective assumption* [3], which assumes that a bag's label is collectively determined by *a set of instances* or even all the instances in the corresponding bag. One representative application conforming to the collective assumption is the region-based image classification task [4], [5], where the label of an image always refers to some object in that image. Because current automatic image segmentation methods may cut the object responsible for the label of the image into multiple parts, any single part cannot represent the object satisfactorily. It is the collective property of multiple parts that determines the label of the image.

From the formulation of MIL, we can easily see that a positive bag may contain some negative instances in addition to one or more positive instances. Hence, the true labels for the instances in a positive bag may or may not be the same as the corresponding bag label and, consequently, the instance labels are inherently ambiguous. In the MIL literature [2], *true positive instances* and *false positive instances* refer to the positive and negative instances, respectively, in the positive bags.

1.2 Motivation

Since labels are not available for the training instances, some methods [2], [6] simply assume that all instances in a bag have the same label as the bag label. However, this assumption can be very unreasonable for positive bags because a positive bag may contain as few as only one true positive instance. If the majority of the negative instances in a positive bag are mislabeled this way, the features learned for distinguishing positive instances from negative instances may end up being very misleading. Other methods [7], [8], [9] try to extend some standard single-instance learning (SIL) methods for multi-instance data by adding some constraints. Unfortunately, the resulting methods typically require solving non-convex optimization problems which suffer from the local minima problem and have high computation cost.

Considering the limitations of some previous methods, we advocate here the necessity of instance label disambiguation as a way to eventually improve the prediction accuracy of the bag labels. In the context of MIL, *disambiguation* essentially refers to identifying the true positive instances in the positive bags.

However, existing disambiguation methods cannot achieve promising performance. The APR (axis-parallel rectangle) method [1] tries to find an axis-parallel rectangle (or, more generally, hyper-rectangle) in the feature space to represent the area of the true positive instances. This rectangle should include at least one instance from each positive bag but exclude all instances from the negative bags. Although APR works quite well for the drug activity prediction problem, it is highly possible that no APR can be found for some other applications, such as image or text categorization, to satisfy the requirement that at least one instance from each positive bag is included while all instances from the negative bags are excluded. Moreover, APR is very sensitive to labeling noise. Suppose only one negative bag is mislabeled as a positive one. In order to include at least one instance from this mislabeled negative bag, the computed APR may

contain so many negative instances that it cannot represent the area of true positive instances at all (cf. Section 3.2.3). DD (diverse density) value [10] is proposed to measure how many different positive bags have instances near a point in the feature space and how far the negative instances are from that point. The DD method tries to find the point with the highest DD value as the target concept. The DD method is very sensitive to labeling noise too since the DD value at a point will be exponentially reduced if an instance from some negative bag is close to that point (cf. Section 3.2.3). This phenomenon has been validated empirically by [11]. Moreover, since the DD landscape contains local maxima, searching for the point with the highest DD value generally requires multiple restarts and hence incurs high computation cost. Other methods, such as mi-SVM [7], adopt some heuristic methods to solve the optimization problem, which may lead to local minima and incur high computation cost.

1.3 Main Contributions

In this paper, we propose a very efficient and robust MIL method, called MILD (Multiple-Instance Learning via Disambiguation), for general MIL problems. The main contributions of this paper can be summarized as follows:

- By investigating the properties of true positive instances in depth, we propose a novel disambiguation method for identifying the true positive instances in the positive bags. This method is not only very efficient but also very robust.
- Two feature representation schemes, one for instance-level classification and the other for bag-level classification, are proposed to convert the MIL problem into a standard single-instance learning problem that can be solved directly by well-known SIL algorithms, such as support vector machine (SVM).
- By combining the two feature representation schemes, a multi-view semi-supervised learning method based on co-training [12] is proposed for MIL. To the best of our knowledge, this is the first inductive semi-supervised learning method for MIL.
- To demonstrate the promising performance of our method, we extensively compare our method with many state-of-the-art MIL methods in diverse applications, including drug activity prediction, protein sequence classification and image classification.

One of the most attractive advantages of our methods is that, after instance label disambiguation, the MIL problem is converted into a standard single-instance learning problem. As a result,

many existing supervised and semi-supervised learning methods can be easily adapted for MIL applications.

2 Related Work

Over the past few years, many applications have been formulated as MIL problems. These include drug activity prediction [1], [13], [14], [15], image classification [4], [11], [16], [17], [18], text classification [2], [7], stock selection [7], [10], protein family modeling [19], computer aided diagnosis [20], [21], and security applications [22]. To solve these problems, many MIL methods have been proposed. The first MIL method is APR [1], which represents the target concept by an axis-parallel rectangle (or hyper-rectangle) in the feature space. The rectangle includes at least one instance from each positive bag but excludes all instances from the negative bags. [10] proposed a measure called DD (diverse density), which essentially measures how many different positive bags have instances near a point in the feature space and how far the negative instances are from that point. A DD-based method tries to find the point with the highest DD value as the target concept. EM-DD [23], [24] combines expectation-maximization (EM) [25] with the DD formulation by using EM to search for the most likely concept.

Several other methods try to modify standard single-instance learning methods for MIL by introducing constraints derived from the MIL formulation. [7] proposed two methods based on SVM, one (mi-SVM) for instance-level classification and the other (MI-SVM) for bag-level classification. Both methods are formulated as mixed integer quadratic programming problems. [26] applied deterministic annealing to the SVM formulation to find better local mimima compared to the heuristic methods. [6] proposed a kernel function directly for bags. With this multi-instance (MI) kernel, in principle any kernel-based classifier can be trained for classifying the bags. [27] extended this work by proposing a marginalized MI kernel to convert the MIL problem from an incomplete data problem to a complete data problem. [28] proposed an SVM-based method for sparse positive bags by directly enforcing the desired constraint that at least one of the instances in a positive bag is positive. [9] proposed to apply transductive SVM for solving MIL problems.

Recently, some methods try to convert MIL into a standard single-instance problem and then solve it with conventional methods by representing the bags as feature vectors. [4] proposed the DD-SVM method through a feature mapping defined on some instance prototypes learned by DD. [11] proposed another method, called MILES (Multiple Instance Learning via Embedded

instance Selection), based on a novel instance-based feature mapping and the 1-norm SVM model for simultaneous feature selection and classification.

However, most of the existing methods cannot answer the essential question giving rise to the gap between multiple-instance learning and single-instance learning: which are true positive instances and what property do the true positive instances have? This motivates the work in this paper.

3 MILD: Multiple-Instance Learning via Disambiguation

In this section, our disambiguation method is described in detail, and then two feature representation schemes are presented for instance-level classification and bag-level classification, respectively, based on which the MIL problem is converted into a standard single-instance leaning problem that can be directly solved by SIL algorithms, such as SVM. In addition, multi-view learning [12] can be easily adapted for MIL by combining the two views, the instance-level view and the bag-level view, of the bags.

3.1 Notations

B_i^+ denotes a positive bag and B_i^- denotes a negative bag. When the label of a bag does not matter, we simply denote the bag as B_i. B_{ij}^+ denotes an instance in a positive bag B_i^+ and B_{ij}^- is an instance in a negative bag B_i^-. Let $\mathbf{B} = \{B_1^+, B_2^+, \ldots, B_{n^+}^+, B_1^-, B_2^-, \ldots, B_{n^-}^-\}$ denote the set of n^+ positive and n^- negative training bags. $l(B_i) \in \{+1, -1\}$ is the bag label of B_i and $l(B_{ij}) \in \{+1, -1\}$ is the instance label of B_{ij}. In general, we always represent all instances as feature vectors of the same dimensionality. Hence, in this paper, an instance also refers to a feature vector.

3.2 Disambiguation

According to the MIL formulation, all instances in the negative bags are negative and hence there exists no ambiguity in the negative bags if there is no labeling noise. As for the positive bags, the only thing we know is that each positive bag must contain at least one true positive instance, but it may also contain many negative instances. Thus, ambiguity arises in the positive bags since we do not know the labels of the instances there. The goal of disambiguation is to identify the true positive instances in the positive bags.

3.2.1 Some Properties of True Positive Instances

***Assumption** 1: We assume that given a true positive instance t, the **probability** that an instance B_{ij} is positive is calculated as follows:*

$$\Pr\left(l(B_{ij}) = +1 \mid t\right) = \exp\left(-\frac{\|t - B_{ij}\|^2}{{\sigma_t}^2}\right), \tag{1}$$

where $\|x\| \triangleq \sqrt{\sum_k x_k^2}$ denotes the 2-norm of the vector x, and σ_t is a parameter [1] larger than 0.

***Remark** 1:* The rationale of Assumption 1 is similar to that of kernel density estimation with the kernel being a Gaussian. Kernel density estimation can be thought of as placing a small "bump" at each observation, which will give higher probability density near the observations. Similarly, if we have already observed an instance t to be truly positive, then the chance that an instance near t is truly positive will be relatively high. However, one key point that should be noted is that unlike kernel density estimation which tries to estimate a *probability density* function, Assumption 1 is used to compute a *conditional probability*. From (1), we can easily see that $0 \leq \Pr(l(B_{ij}) = +1 \mid t) \leq 1$, $\Pr(l(B_{ij}) = +1 \mid t) = 0$ when $\|t - B_{ij}\| = +\infty$, and $\Pr(l(B_{ij}) = +1 \mid t) = 1$ when $\|t - B_{ij}\| = 0$. Obviously, this is a reasonable probability function, but not a reasonable probability density function. Furthermore, the assumption in (1) also coincides well with our intuition. If we have known that t is a true positive instance and $\|t - B_{ij}\| = 0$, B_{ij} will definitely be a true positive instance because B_{ij} is just equal to t, which means that $\Pr(l(B_{ij}) = +1 \mid t) = 1$ is reasonable. Similarly, if $\|t - B_{ij}\| = +\infty$, B_{ij} is infinitely far away from the true positive instance t, $\Pr(l(B_{ij}) = +1 \mid t) = 0$ is also reasonable. The farther B_{ij} is away from t, the lower is the probability that B_{ij} is positive given t, which is reasonable based on our intuition.

***Remark** 2:* We try to deal with the general case that all the true positive instances are not necessarily identical to a single point in the instance space, but are generated from some probability distribution over the whole instance space. Hence, each point in the instance space is associated with a probability value to indicate how confidently that point is truly positive.

***Definition** 1: The **most-likely-cause estimator** for estimating the probability that a bag B_i is*

1. This parameter will be learned from the training data by the algorithm introduced later.

a positive bag given a true positive instance t is defined as follows:

$$\begin{aligned}\Pr(l(B_i) = +1 \mid t) &= \max_{B_{ij} \in B_i} \Pr(l(B_{ij}) = +1 \mid t) \\ &= \max_{B_{ij} \in B_i} \exp\left(-\frac{\|t - B_{ij}\|^2}{\sigma_t^2}\right) \\ &= \exp\left(-\frac{d^2(t, B_i)}{\sigma_t^2}\right), \end{aligned} \tag{2}$$

where

$$d(t, B_i) = \min_{B_{ij} \in B_i} \|t - B_{ij}\|. \tag{3}$$

In other words, the distance $d(t, B_i)$ between an instance t and a bag B_i is simply equal to the distance between t and the nearest instance in B_i.

The definition of the most-likely-cause estimator completely follows the underlying principle of the MIL formulation. Let B_{ik} be the instance in bag B_i which is the most likely one to be positive, i.e., $\Pr(l(B_{ik}) = +1 \mid t)$ is the largest among all instances in B_i. If $\Pr(l(B_{ik}) = +1 \mid t)$ is small enough, certainly we should assign B_i to be negative. Otherwise, B_i should be positive, even if all instances in B_i other than B_{ik} have low probabilities to be positive. Moreover, the definition is also consistent with intuition. If $d(t, B_i) = 0$, which implies that t is an instance of B_i, $\Pr(l(B_i) = +1 \mid t)$ will be 1. This is reasonable, because B_i contains a true positive instance t. In general, the larger $d(t, B_i)$ is, the lower is the probability that B_i is positive given the true positive instance t.

***Theorem** 1: Given a true positive instance t, there exists a threshold θ_t which makes the decision function defined in (4) label the bags according to the Bayes decision rule.*

$$h_{\theta_t}^t(B_i) = \begin{cases} +1 & d(t, B_i) \le \theta_t \\ -1 & \textit{otherwise} \end{cases} \tag{4}$$

Proof: According to the Bayes decision rule [29, page 23], the label of B_i given a true positive instance t should be decided as follows:

$$l(B_i \mid t) = \begin{cases} +1 & \Pr(l(B_i) = +1 \mid t) \ge \Pr(l(B_i) = -1 \mid t) \\ -1 & \text{otherwise} \end{cases}$$

From Definition 1, we have

$$\begin{aligned}
&\Pr(l(B_i) = +1 \mid t) - \Pr(l(B_i) = -1 \mid t) \\
&= \exp\left(-\frac{d^2(t, B_i)}{\sigma_t^2}\right) - \left[1 - \exp\left(-\frac{d^2(t, B_i)}{\sigma_t^2}\right)\right] \\
&= 2\exp\left(-\frac{d^2(t, B_i)}{\sigma_t^2}\right) - 1.
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
&\Pr(l(B_i) = +1 \mid t) \geq \Pr(l(B_i) = -1 \mid t) \\
\Leftrightarrow\ &\Pr(l(B_i) = +1 \mid t) - \Pr(l(B_i) = -1 \mid t) \geq 0 \\
\Leftrightarrow\ &2\exp\left(-\frac{d^2(t, B_i)}{\sigma_t^2}\right) - 1 \geq 0 \\
\Leftrightarrow\ &d(t, B_i) \leq \sigma_t\sqrt{\ln 2}. \qquad (5)
\end{aligned}$$

Hence, if $\theta_t = \sigma_t\sqrt{\ln 2}$, the decision function defined in (4) will label the bags in accordance with the Bayes decision rule. □

Therefore, if t is a true positive instance, there must exist a decision function as defined in (4) to label the bags *well*, meaning that the distances from t to the positive bags are expected to be smaller than those to the negative bags.

For a negative instance (i.e., false positive instance), however, its distances to the positive and negative bags do not exhibit the same distribution as those from t. Since some positive bags may also contain negative instances just like the negative bags, the distances from the negative instance to the positive bags may be as random as those to the negative bags. This distributional difference provides an informative hint for identifying the true positive instances.

3.2.2 Disambiguation Method

Unlike in the previous subsection, t does not necessarily refer to a true positive instance in this subsection. However, we still define a decision function as that in (4) even when t is a negative instance. The difference is that if t is a negative instance, the corresponding decision function *cannot* label the bags *well*. It is this very phenomenon that forms the basis of our disambiguation method.

***Definition** 2: The **empirical precision** of the decision function in (4) is defined as follows:*

$$P_t(\theta_t) = \frac{1}{n^+ + n^-} \sum_{i=1}^{n^+ + n^-} \frac{1 + h_{\theta_t}^t(B_i)\, l(B_i)}{2}, \qquad (6)$$

where $B_i \in \mathbf{B}$ is a training bag.

The empirical precision essentially measures how well the decision function $h^t_{\theta_t}(\cdot)$ with threshold θ_t mimics $l(\cdot)$ in predicting the bag labels.

Now, the question is that we do not know the scaling factor σ_t^2 of the function in (2) for a specific instance t. As such, we cannot compute the threshold θ_t. It will be desirable to learn σ_t^2 from the training data automatically. To do so, we find the best threshold θ_t^* that maximizes the empirical precision to estimate σ_t^2: $\theta_t^* = \arg\max_{\theta_t} P_t(\theta_t)$. Thus the best (maximum) empirical precision $P^*(t)$ for t is given by:

$$P^*(t) = \max_{\theta_t} P_t(\theta_t). \tag{7}$$

***Remark** 3:* The basic idea of finding θ_t^* is analogous to maximum likelihood estimation (MLE) which finds the parameters that best explain the observed data. However, the criterion used here is the empirical precision as defined in (6) rather than the likelihood. Furthermore, our method is *totally different* from the DD method. The objective function of DD is *multiplicative*, making DD very sensitive to noise. Howerver, the objective function of our method is the empirical precision in (6) with an *additive* form, which guarantees the robustness of our method. The robustness of our method will be further discussed later.

In essence, $P^*(t)$ reflects the ability of instance t in discriminating the training bags. The larger $P^*(t)$ is, the more likely is t a true positive instance.

It is worth noting that although θ_t is a continuous-valued variable, we can find θ_t^* by checking a finite set of candidate values only. This is guaranteed by Theorem 2.

***Theorem** 2: The best empirical precision $P^*(t)$ for t is achieved when θ_t is an element in the set $\{d(t, B_i^+) \mid i = 1, \ldots, n^+\}$.*

Proof: Let θ_t^* be a value satisfying $P_t(\theta_t^*) = P^*(t)$ and θ_t' be the maximum value in $\{d(t, B_i^+) \mid i = 1, \ldots, n^+\}$ that is no larger than θ_t^*.

If B_i^+ is correctly labeled when $\theta_t = \theta_t^*$, then $d(t, B_i^+) \leq \theta_t^*$, and hence $d(t, B_i^+) \leq \theta_t'$. Therefore, any correctly labeled positive bag when $\theta_t = \theta_t^*$ is also correctly labeled when $\theta_t = \theta_t'$. Similarly, if B_i^- is correctly labeled when $\theta_t = \theta_t^*$, then $d(t, B_i^-) > \theta_t^*$, and hence $d(t, B_i^-) > \theta_t'$. Therefore, any correctly labeled negative bag when $\theta_t = \theta_t^*$ is also correctly labeled when $\theta_t = \theta_t'$.

Thus we have, $P_t(\theta_t') = P_t(\theta_t^*) = P^*(t)$. □

Therefore, to estimate the best threshold θ_t^* (and hence σ_t^2), we only need to compute the distance from t to each positive training bag. This procedure can be done very efficiently.

Ideally, disambiguation should identify *all* the true positive instances in the positive bags. However, the only thing we know is that each positive bag contains at least one true positive instance, but we do not know the exact number of true positive instances for one specific positive bag. Different positive bags may have different number of positive instances, and furthermore, the discriminative ability of the true positive instances from different positive bags might also be different. Hence, although we know in general the true positive instances will have higher ability to discriminate the training bags than negative instances, it is not easy to find a *simple* rule to identify *all* the true positive instances in the positive bags. Actually, from our experiments (cf. Section 4.3), we find that identifying *all* the true positive instances might exceed the requirement necessary for us to complete common MIL tasks, and consequently incur some unnecessary cost. Here, we design a simple disambiguation method to identify just *one* positive instance from each positive bag, which is enough for most MIL tasks based on our experiments [2].

Since the MIL formulation requires that each positive bag contains at least one true positive instance, it is always possible to find a true positive instance in a positive bag. In our algorithm, we select from each positive training bag the instance with the largest P^* value as a candidate true positive instance. After true positive instance selection, the disambiguation process is completed.

Algorithm 1 summarizes the disambiguation method presented above.

3.2.3 Comparison with Other Disambiguation Methods

APR (axis-parallel rectangle) [1] and DD (diverse density) [10] are two well-known disambiguation methods. Compared with them, our disambiguation method is more efficient and robust. Quantitative empirical comparison will be performed in the next section. We first give a qualitative comparison here with respect to the robustness property.

APR represents the target concept by an axis-parallel rectangle which includes at least one instance from each positive bag but excludes all instances from the negative bags. Figure 1 gives a toy example to show how APR works. There are altogether nine bags ($B_1^+, B_2^+, B_3^+, B_4^+, B_5^-, B_6^-, B_7^-, B_8^-, B_9^-$) in the two-dimensional feature space. If there is no noise, APR will choose the area in the red

2. The extension of our simple version of disambiguation method for some complex tasks will be discussed in Section 3.5.1.

Algorithm 1 Disambiguation Method

Input: All training bags $B_1^+, \ldots, B_{n^+}^+, B_1^-, \ldots, B_{n^-}^-$

Initialize: $X = \{x_1, x_2, \ldots, x_p\}$ is the set of re-indexed instances from all positive training bags; $T^* = \phi$ (empty set), where T^* is used to keep the set of selected true positive instances.

for $k = 1$ **to** p **do**

Compute $P^*(x_k)$ according to (7)

end for

for $i = 1$ **to** n^+ **do**

$t_i^* = \arg\max_{B_{ij}^+ \in B_i^+} P^*(B_{ij}^+)$

Add t_i^* to T^*

end for

Output: T^*

(smaller) rectangle as the true positive area, which is a very reasonable choice for this noise-free case. However, if we mislabel even just one negative bag, say B_5^-, to be positive, APR can no longer find a rectangle to include at least one instance from each positive bag but exclude all instances from the negative bags. If B_6^- is not in the training set, then the algorithm will choose the blue (bigger) rectangle, which is obviously not the real true positive area. Hence, APR is very sensitive to labeling noise. Furthermore, in many applications such as image classification, it is very difficult or even impossible for APR to find a rectangle that satisfies all the constraints.

As for DD [10], it finds a single point in the feature space as well as the best feature weights corresponding to that point to represent the concept and then decides the label of each bag based on the distance from the bag to the concept point. If the weighted distance from the concept point to any instance of a bag is below a threshold, the bag will be labeled as positive. Hence, the true positive area of DD is an ellipse (or, more generally, hyperellipsoid). In the toy example in Figure 1, we can assume that both features have equal weights and hence the computed true positive area is a circle. When there is no noise, DD finds the red point as the computed concept point and the red (smaller) circle as the true positive area. This result is quite reasonable. However, DD is also very sensitive to labeling noise, which has been pointed out in [11]. From the toy example, we can also observe this phenomenon easily. If we mislabel only one positive

bag, say B_1^+, to be negative, then the concept point will move to the blue point and the true positive area will be the blue (bigger) circle, which is also not the real true positive area.

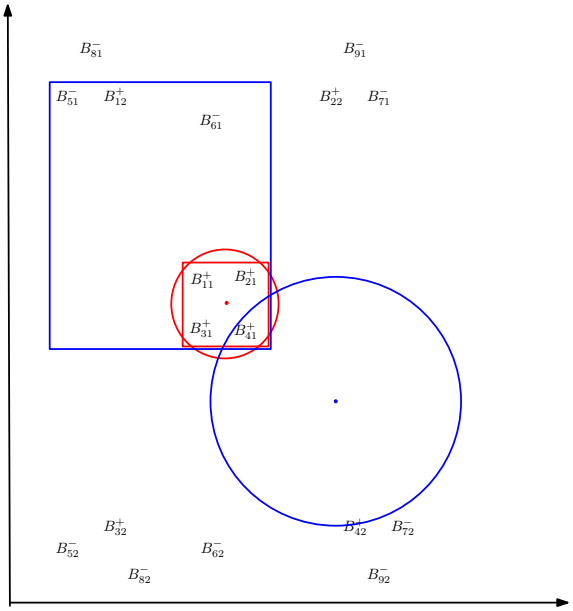


Fig. 1. A toy example to show that APR and DD are sensitive to labeling noise. B_{ij}^+ and B_{ik}^+ are two instances from the same positive bag B_i^+, while B_{ij}^- and B_{ik}^- are two instances from the same negative bag B_i^-.

On the contrary, our disambiguation method is much more robust. In the empirical precision measure defined in (6), negating a small number of labels will not change the overall precision value significantly. This makes our method more robust towards labeling noise. To illustrate this, let us assume that x^+ is a true positive instance from a positive training bag and x^- is a negative instance (false positive instance) from the same bag. The precisions $P^*(x^+)$ and $P^*(x^-)$ are computed from (7). Since negative instances also exist in positive bags, the distances from x^- to the positive bags are as random as those to the negative bags. Hence, $P^*(x^-)$ may approach 50% while $P^*(x^+)$ is relatively high because $d(x^+, B_i^+)$ is expected to be smaller than $d(x^+, B_j^-)$. If we add noise by changing the labels of $d\%$ of positive bags and $d\%$ of negative bags, then $P^*(x^-)$ may still be about 50% but $P^*(x^+)$ may decrease by up to $d\%$. However, as long as $P^*(x^+) - d\% > P^*(x^-)$, the selected true positive instance is still x^+. In real applications, $P^*(x^+)$ can be very high, even 100%, if all the true positive instances form a

cluster and most negative instances are relatively far away from the cluster, implying that some of the true positive instances can still be correctly selected even if the noise level is very high. This will be illustrated in our experiments to be presented later in the experiment section.

In the toy example of Figure 1, if we mislabel a positive bag, say B_1^+, to be negative, our method can still find that B_{21}^+, B_{31}^+ and B_{41}^+ are true positive instances. On the other hand, if we mislabel one negative bag, say B_5^-, to be positive, B_{11}^+, B_{21}^+, B_{31}^+ and B_{41}^+ can still be correctly computed as positive instances. However, from the above analysis, DD and APR will fail with these kinds of mislabeling information.

Moreover, a promising property of our method is that it has no parameters to tune. The only parameter θ_t can be determined entirely from the training data. From Theorem 2, we only need to set θ_t to each of the distances from t to the positive bags and then find the value that gives the highest empirical precision. This is very promising as no effort is needed for parameter tuning using time-consuming methods such as cross validation. In fact, it is almost real-time in our experimental evaluation. For example, it only takes 0.06 second on a 2GHz PC in our experiment to complete the disambiguation process for one fold of the 10-fold cross validation on the MUSK1 data set (cf. Section 4.1.1). As a result, the overall method based on our disambiguation method is one order of magnitude faster than DD-based methods based on experimental evaluation on the drug activity prediction task (cf. Table 7).

3.3 Maximum Margin Formulation of MIL

Maximum margin classifiers such as SVM deliver promising generalization performance because model complexity or capacity is controlled well by maximizing the margin [30]. However, due to ambiguities in the label information, such classifiers designed for standard single-instance learning cannot be applied to MIL directly. In this section, we propose two feature representation schemes based on the disambiguation method presented above to directly adapt SVM to the MIL setting. One scheme is for instance-level classification (called MILD_I) while the other one is for bag-level classification (called MILD_B). The best choice between them will depend on the application domain, which will be described at the end of this subsection.

3.3.1 Instance-Level Classification

After disambiguation as described above, the set of true positive instances, T^*, is computed. Moreover, all instances in the negative bags are assumed to be negative. We can make use of these labeled instances to train a classifier and then use it to predict the labels of the instances in any new bag. If at least one instance of a bag is predicted to be positive, then the bag is labeled positive. Otherwise, the bag is labeled negative.

In some MIL problems, however, some bags may contain a large number of instances. If we use all the labeled instances described above to train a classifier, its computational and storage requirements may become too large. One possible remedy is to select a representative set of instances for training. Selecting one instance from each bag can enhance diversity and hence representation power. Since T^* in Algorithm 1 contains one instance from each positive bag, it is reasonable to choose T^* as the set of positive training instances. For the negative training instances, we select one representative instance N_k^- from each negative bag B_k^- based on the positive training set T^*, as follows:

$$N_k^- = \arg \min_{B_{kj}^- \in B_k^-} \left\{ \min_{t^* \in T^*} ||B_{kj}^- - t^*||^2 \right\}, \tag{8}$$

which is the instance in B_k^- nearest to T^*. The rationale of this selection scheme is that the negative instances selected will likely define the negative side of the margin, playing a role similar to the support vectors in SVM. Moreover, selecting one instance from each bag enhances diversity in the training set.

After selecting the representative instances, we propose a novel instance-level feature representation scheme. For any instance x_k in the training set, we embed it into a feature space with each dimension corresponding to one of the positive and negative training bags. The feature vector of the embedded instance is given by:

$$\begin{aligned} \psi(x_k) = [&d(x_k, B_1^+), \ldots, d(x_k, B_{n^+}^+), \\ &d(x_k, B_1^-), \ldots, d(x_k, B_{n^-}^-)]^T, \end{aligned} \tag{9}$$

where $d(x_k, B_i)$ denotes the distance from x_k to a bag B_i computed according to (3).

Intuitively, if an instance is a true positive instance, then the feature values in (9) corresponding to the positive bags will be smaller while those corresponding to the negative bags will be larger. For a negative instance, on the other hand, its feature values do not give the same distribution.

Since some positive bags may also contain negative instances just like the negative bags, the distances from the negative instance to the positive bags are as random as those to the negative bags. Experimental results in Section 4.4.1 show that the proposed feature representation scheme is much more discriminative than using the original instance features.

Afterwards, an SVM classifier is trained based on this feature representation. Suppose a bag to predict during the prediction stage is B_i and the trained SVM is $f(\cdot)$. Here, $f(\cdot)$ refers to the (continuous) decision value before thresholding but not the (discrete) prediction label. Since having a single positively classified member instance is necessary and sufficient to classify a bag as positive, the following rule is adopted to predict the label of a new bag based on the instance-level classifier $f(\cdot)$:

$$l(B_i) = \text{sign}\left(\max_{B_{ij} \in B_i} f(B_{ij})\right). \tag{10}$$

3.3.2 Bag-Level Classification

Since the labels of the training bags are already known, we can train an SVM to classify the bags directly. Like in DD-SVM [4], we propose a similar feature mapping to map every bag B_i to a point $\mu(B_i)$ as follows:

$$\mu(B_i) = (d(t_1^*, B_i), d(t_2^*, B_i), \ldots, d(t_{n^+}^*, B_i))^T, \tag{11}$$

where $t_k^* \in T^*$, and T^* is the set of selected true positive instances in Algorithm 1.

3.3.3 Choice between MILD_I and MILD_B

From (10), it is not difficult to see that MILD_I is only suitable for application domains satisfying the existential assumption. On the contrary, MILD_B is suitable for application domains satisfying either the existential or the collective assumption. In MILD_B, for a specific bag B_i, different instances B_{ij} in it might be selected as the nearest instances to compute the distance $d(t_k^*, B_i)$ for different t_k^*. So the feature vector $\mu(B_i)$ actually implicitly encodes the collective contribution of those instances nearest to the selected positive instances in T^*. Moreover, to compute $\mu(B_i^+)$ for a positive bag B_i^+, MILD_B automatically excludes the contribution of those instances far away from the selected true positive instances in T^*. This is promising because those excluded instances are most likely to be negative. Hence, most information encoded in the feature vector $\mu(B_i^+)$ for a positive bag is from the positive instances in the bag, which makes our method much more attractive than other methods that unreasonably assume all the instances from a

positive bag as positive. As for applications satisfying the existential assumption, as long as at least one positive instance exists in a bag B_i, to compute $\mu(B_i)$, the distance between t_k^* and B_i is expected to be smaller than the distance from t_k^* to a negative bag. Hence, containing one positive instance is enough to discriminate a positive bag from the negative bags.

Therefore, choosing between MILD_I and MILD_B is by no means an *ad hoc* decision. If the existential assumption is satisfied, both of them are good candidates. Otherwise, we should choose MILD_B. In practice, it is not difficult to check which assumption is satisfied for a specific application.

3.4 Multi-View Learning for MIL

The instance-level classifier and the bag-level classifier provide us with multiple views, the instance-level view and the bag-level view, to look at a bag. Hence, the multi-view learning approach [31] can be easily adapted for MIL. Here, we adopt *co-training* [12] to perform semi-supervised MIL. The basic idea of co-training is to train one learner based on each view of the labeled examples and to iteratively use each learner to label the unlabeled examples with the highest prediction confidence. More specifically, if we want to label some unlabeled examples as positive, we will choose those with the largest prediction values [3] as candidates. Otherwise, if we want to label some unlabeled examples as negative, we will choose those with the smallest prediction values as candidates.

Algorithm 2 summarizes the co-training method for MIL, in which F_i and F_b denote the instance-level classifier and bag-level classifier, respectively. We can see that in each iteration, $2n_1 + 2n_2$ unlabeled bags will be assigned labels by the classifiers trained based on the existing labeled data, and then these newly labeled data are added into the existing labeled data to get an expanded set of labeled data. This process iterates for several rounds till some termination condition is satisfied. After the whole algorithm is terminated, the final F_i and F_b will be used for prediction.

3. Here, the prediction value refer to the (continuous) decision value before thresholding but not the (discrete) prediction label.

Algorithm 2 Co-training Method for Multiple-Instance Learning

Input: A set of labeled training bags L; a set of unlabeled training bags U; the number of co-training iterations Q.

Initialize: Construct a pool U' of bags by randomly selecting u bags from U.

for $q = 1$ **to** Q **do**

Train a classifier F_i based on the labeled training bags L;

Train a classifier F_b based on the labeled training bags L;

Use F_i to label n_1 positive and n_2 negative bags with the highest prediction confidence from U';

Use F_b to label n_1 positive and n_2 negative bags with the highest prediction confidence from U';

Move these self-labeled bags from U' to L;

if the number of bags in U is less that $2n_1 + 2n_2$ **then**

break;

end if

Randomly select $2n_1 + 2n_2$ bags from U and move them from U to replenish U';

end for

3.5 Discussion

3.5.1 Extension of Current Disambiguation Method

For the applications satisfying the existential assumption, if we want to identify all the true positive instances in the positive bags, one direct way is to use the selected true positive instances in Algorithm 1 to train an instance-level classifier based on the method introduced in Section 3.3.1. Then we can apply this trained classifier to classify all the instances from the positive training bags to identify all the true positive instances.

In some cases that the collective assumption is satisfied, if we know that at least K positive instances exist in each positive bag, we can extend Algorithm 1 to select from each positive training bag K instances with the largest P^* values as candidate true positive instances.

3.5.2 Disambiguation Versus Instance Selection

Unlike MILD which tries to select positive instances only from positive training bags, some other methods, such as MILES [11] and the AdaBoost based method [32] [4], try to select some instances from the training bags, including both positive and negative bags, and then learn a bag-level classifier based on the selected instances. Hence, both MILES and the AdaBoost based method [32] have to compute a much larger feature matrix in (12) and, consequently, incur higher computation cost than MILD, which will be discussed in detail in Section 4.7.

Furthermore, the goal of MILD is different from that of MILES and the AdaBoost based method [32]. MILD tries to identify the *true positive* instances while the other two methods cannot guarantee the selected instances to be truly positive. Actually, from the experiments in [11] and [32], we can see that the other two methods will select many *negative* instances for classifier training. Hence, based on the selected instances by the other two methods, it is not easy to train an instance-level classifier.

3.5.3 Computation Cost

In Algorithm 1, for each one of the p re-indexed instances from all positive training bags, we have to compute the distances from it to the instances from all the training bags. Let v denote the total number of instances in the whole training bags. We can see that the computation cost is $O(pv)$ with $v > p$, which will be prohibitive when the number of positive training bags is large. In this case, we can use the idea of the vocabulary generating process for generic object recognition [34] to reduce the computation cost. More specifically, we can randomly select part of the training bags to complete the disambiguation procedure. This is reasonable because from our experiment, in general, we observe that identifying only part of the true positive instances might not necessarily decrease the classification accuracy (cf. Section 4.3).

4 Experimental Evaluation

In all the experiments, the Gaussian kernel is used for the SVM training for our method. We use LIBSVM [35] to train all the SVM classifiers with the output being probability estimates.

4. The AdaBoost based method in [32] is for generic object recognition, which can also be seen as an MIL problem when each image is represented as a set of local regions. This is similar to the formulation of localized content-based image retrieval [33].

4.1 Data Sets for Evaluation

We evaluate our proposed method based on some publicly available benchmarks: the MUSK data sets [1] for drug activity prediction, the TrX data set for protein sequence classification [19], and the SIVAL set [5] for image retrieval.

4.1.1 MUSK Data Sets

The MUSK data sets [1], MUSK1 and MUSK2, consist of descriptions of molecules. A molecule is considered as a bag and each of its steric configurations (i.e., molecular shapes) is an instance. Each configuration is represented by a 166-dimensional feature vector derived from the surface properties. MUSK1 contains 47 musk molecules (positive bags) and 45 similar non-musk molecules (negative bags). MUSK2 contains 39 musk molecules and 63 non-musk molecules. A total of 72 molecules are shared between MUSK1 and MUSK2. MUSK1 contains approximately 5.17 instances per bag and MUSK2 contains 64.69 instances per bag.

4.1.2 TrX Data Set

The goal of the identification of Thioredoxin (TrX) proteins is to classify the given protein sequences according to whether they are from the family of TrX proteins [19]. Due to low conservation of the *primary sequences* [36] in the protein superfamily Thioredoxin-fold (Trx-fold), conventional classification methods such as hidden Markov model (HMM) cannot model it well. In fact, it is natural to formulate this problem as an MIL problem. As pointed out by [36], "All Trx-fold proteins satisfy one of the following conditions and few non-Trx-fold proteins satisfy any of them: (a) a Kyte-Doolittle value of -4.5 around position 85 and a Kyte-Doolittle value of -0.75 near position 10; (b) a Kyte-Doolittle value of 3.5 near position 55 and Kyte-Doolittle value of 4.25 near position 92 and Kyte-Doolittle value of 1.5 near position 25; etc." Hence, the class labels of the proteins (Trx-fold or non-Trx-fold) are decided by one or several positions in the primary sequences of the proteins. However, the conditions responsible for the labels of the protein sequences, such as those in (a) and (b) above, are not *a priori* known. Instead, we must learn them from the training data. This is exactly what our disambiguation method tries to do if the problem is formulated as follows: a bag corresponds to an aligned primary sequence around the motif and an instance in the bag corresponds to a position in the sequence. In the TrX data set [19], for each sequence, the primary sequence motif (typically

CxxC), which is known to be conserved in all TrX-fold proteins, is first found. Then, a window of size 214 around the motif (30 residues upstream, 180 downstream) is extracted and aligned around the motif. Each aligned protein is represented by a bag. An instance corresponds to a position, which is represented by an 8-dimensional profile of the amino acid at that position. Some information about the TrX data set is summarized in Table 1.

TABLE 1
Information about the TrX data set.

#Positive Bags	#Negative Bags	#Instances	#Dimensions
20	168	25855	8

4.1.3 SIVAL Image Set

The SIVAL image set [33] contains 1,500 images of 25 categories, with 60 images for each category. These categories are complex objects photographed against 10 different highly diverse backgrounds. Six different images are taken for each object-background pair. For each object category, the same physical object is used in all scenes but the scenes are highly complex and diverse. The objects are photographed at different angles and they can appear at any location in the images. The target object occupies only about 10–15% of the whole image area in most images. Category 1 to category 25 are: "AjaxOrange", "Apple", "Banana", "BlueScrunge", "CandleWithHolder", "CardboardBox", "CheckeredScarf", "CokeCan", "DataMiningBook", "DirtyRunningShoe", "DirtyWorkGloves", "FabricSoftenerBox", "FeltFlowerRug", "GlazedWoodPot", "GoldMedal", "GreenTeaBox", "JuliesPot", "LargeSpoon", "RapBook", "SmileyFaceDoll", "SpriteCan", "StripedNoteBook", "TranslucentBowl", "WD40Can", and "WoodRollingPin". Figure 2 shows some sample images from the SIVAL image set.

For image retrieval, each image corresponds to a bag, and an image will be segmented into several regions with each region corresponding to an instance. We use the same preprocessing method as that in [5], [33] to generate the bags. Hence, each image is represented as a bag of 32 30-dimensional instances.

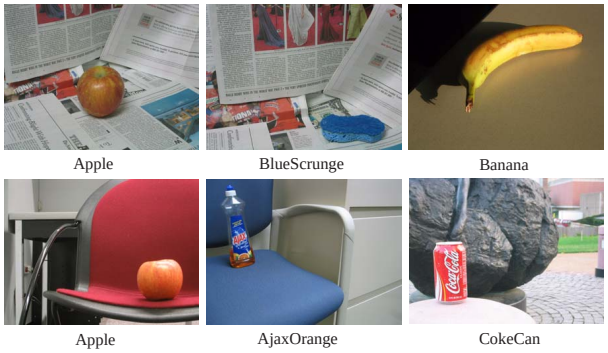


Fig. 2. Sample images from the SIVAL image set.

4.2 Evaluation of Disambiguation Method

Although many data sets from diverse application areas have been used as benchmarks for MIL, few of them, with the exception of the augmented SIVAL image set [5], [37] and the semi-synthetic 20 Newsgroups set [37], provide ground truth labels for the instances in the bags. The SIVAL image set is adopted to evaluate the performance of our disambiguation method. We use the 60 images (bags) containing the object "FabricSoftenerBox" as positive training bags and the 60 images containing the object "CheckeredScarf" as negative training bags. One instance from each positive training bag is selected based on Algorithm 1. Experimental results show that *all* the selected instances (i.e., elements of T^* in Algorithm 1) have +1 as their ground truth label, which demonstrates the effectiveness of our disambiguation method.

4.3 Varying the Number of Selected Instances

First, we perform 10-fold cross-validation on the MUSK1 data set using a simple instance selection strategy which extends Algorithm 1 to select from each positive training bag K instances with the largest P^* values. The average accuracy of 10-fold cross-validation is shown in Figure 3. We can see that when $K = 1$, which corresponds to the disambiguation strategy in Algorithm 1, both MILD_B and MILD_I achieve very promising performance. However, adding extra instances ($K = 2, 3, 4$) will deteriorate the performance, especially for MILD_I. Hence, we can assert that there must be some negative instances selected when $K = 2, 3, 4$. Furthermore, compared with the case of $K = 2$, the much better performance of MILD_I when $K = 1$ confirms

that the identified instances in Algorithm 1 are most likely to be true positive instances. This also conforms to our prior knowledge that the MUSK data sets satisfy the existential assumption.

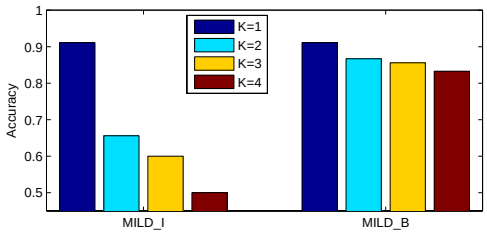


Fig. 3. Accuracy on MUSK1 data set against the number (K) of instances selected from each positive bag.

Second, we use the selected positive instances when $K = 1$ to train an instance-level classifier based on the method introduced in Section 3.3.1, and subsequently use this trained classifier to classify all the instances in the positive training bags. In one of the 10-fold cross-validation, we have 42 positive training bags, 192 instances from which are classified to be positive by the trained instance-level classifier. Then, based on these 192 selected instances, we train MILD_I and MILD_B, the performance of which is *almost the same* as those using 42 initially selected positive instances. Compared with the poor results of the simple strategy above by setting $K = 2, 3, 4$, the good results here mean that the extension strategy in Section 3.5.1 is effective to identify all the true positive instances. Another finding is that it is unnecessary to identify *all* the true positive instances and Algorithm 1 is enough for many MIL problems satisfying the existential assumption.

Third, we study the extension strategy in Section 3.5.1 when the existential assumption is not satisfied. From prior domain knowledge about localized content-based image retrieval [33], it is not difficult to realize that this application does not satisfy the existential assumption but the collective assumption. Hence, only MILD_B is suitable for this application. We use a simple instance selection strategy which extends Algorithm 1 to select from each positive training bag K instances with the largest P^* values. We study the effect of K on the SIVAL image set. We treat the 60 images of Category 1 ("AjaxOrange") as positive and the other 1,440 images as negative. We randomly select eight positive and eight negative images to form the training set and the remaining 1,484 images to form the test set. The results are reported based on 30 rounds

of independent test. Average AUC (area under the ROC curve) values with 95% confidence interval of MILD_B over 30 rounds of test against the number of instances selected from each positive bag are shown in Figure 4. We can see that there is no significant difference among the performances for different values of K when $K < 10$. When $K < 10$, all the selected instances are not necessarily to be truely positive, but the performance of MILD_B will not be significantly affected if only a small part of negative instances exist in the selected instances, which also conforms to the findings in Figure 3. Hence, for applications satisfying the collective assumption, the simple extension strategy in Section 3.5.1 is also enough. How to choose K can be directly observed from the training data. For example, in the SIVAL image set, each image (i.e., bag) contains 32 instances, and we observe that the target object occupies about 10–15% of the whole image area in most images. Hence, any $K \leq 5$ $(32 \times 15\%)$ will be suitable, which also conforms to our experimental results.

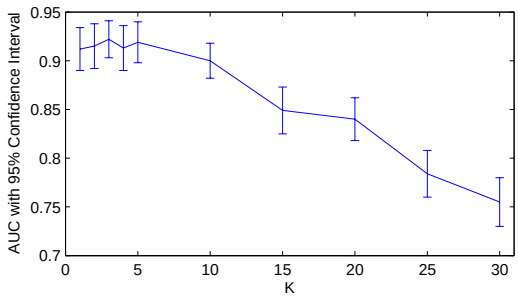


Fig. 4. Performance on SIVAL data set against the number (K) of instances selected from each positive bag.

Another finding from Figure 4 is that when $K > 10$ the performance will deteriorate dramatically. This is because the number of selected negative instances will exceed that of the selected positive instances when $K > 10$, which will make the feature representation in (11) contain too much noisy information. The SIVAL data set differs from other image sets, such as those for generic object recognition [34], in many aspects. In many other image sets, the target object occupies a large portion of the whole image, while in SIVAL the main part of an image is the background (cf. Figure 2). Furthermore, the background of some category in other image sets is always specific to that category of images. For example, in general, the background in the

images of "Computers" is very different from the background in the images of "Horses". But for SIVAL, the background for one category can appear for another category. The poor performance of MILD_B when $K > 10$ on the SIVAL set also implies that the universal bag-of-visual-words (BOV) representation [38] might not be suitable for the SIVAL data set, because BOV based methods randomly select some local regions (either positive or negative) from some images (either positive or negative) to build a vocabulary. For the SIVAL-like image sets, in which the main part of an image is the background, most words in the built vocabulary will be from the background. Because the background can appear in either positive or negative images, the information in the histogram for an image will be dominated by the background information. For example, in Figure 2, the first two images in the upper row will give higher similarity than the two images in the leftmost column if universal BOV representation is adopted, which is not expected by us. One possible strategy to tackle this problem is to adopt the adapted vocabularies in [34] or to integrate the label information into the vocabulary generating process, which is beyond the scope of this paper. Anyhow, our method can achieve very promising performance by identifying the true positive instances.

4.4 Performance of Supervised Methods

4.4.1 Drug Activity Prediction

In LIBSVM [35], there are two parameters, the cost parameter c and the Gaussian kernel parameter g, for SVM training. In our experiment, both of them are chosen from $\{2^{-10}, 2^{-9}, \ldots, 2^{0}, \ldots, 2^{9}, 2^{10}\}$, and those values giving the minimum 2-fold cross-validation error on the training set are selected to set the two parameters. We found that $\{c = 2^4, g = 2^{-5}\}$ for MILD_I and $\{c = 2^2, g = 2^{-1}\}$ for MILD_B gave the minimum 2-fold cross-validation error on MUSK1, and $\{c = 2^7, g = 2^{-6}\}$ for MILD_I and $\{c = 2^2, g = 2^{-1}\}$ for MILD_B gave the minimum 2-fold cross-validation error on MUSK2. We fix these values for all the subsequent experiments. We apply 10-fold cross-validation with 20 random splits of the data to estimate the prediction accuracy on the test data. For MUSK1, the prediction accuracy ranges of MILD_I and MILD_B over 20 runs are [87.8%, 94.4%] and [87.8%,93.3%], respectively. As for MUSK2, the accuracy ranges are [86.0%, 93.0%] and [83.0%,89.0%], respectively.

The average accuracy and 95% confidence interval of the results over 20 runs of 10-fold cross-validation are reported in Table 2. We also list some other results on the same data sets

for comparison. Furthermore, we report the results of another instance-level classifier, called MILD_I0, based on the original instance features rather than our proposed feature mapping in (9). Compared with MILD_I0, the better performance of MILD_I confirms the effectiveness of our instance-level feature representation scheme. From the table, we can see that both the instance-level and the bag-level classifiers achieve performance comparable with the state-of-the-art results, which once again confirms that our disambiguation method is very effective. The performance of our methods on MUSK1 is better than those of all the other methods, with the exception of APR. Note that the APR result is based on choosing the parameters to maximize the *test set* (not training set) performance and hence it is not a fair comparison. Furthermore, APR has been designed specially for drug activity prediction but not for general MIL tasks. The good performance of MILD_I implies that the drug activity prediction task satisfies the existential assumption, which conforms to our prior knowledge about this domain.

TABLE 2

Performance on MUSK data sets (in %, the 95% confidence intervals of the results of MILES and our methods are shown in the square brackets).

Method	MUSK1	MUSK2
MI-SVM [7]	77.9	84.3
mi-SVM [7]	87.4	83.6
DD [10]	88.9	82.5
EM-DD [23]	84.8	84.9
APR [1]	**92.4**	89.2
MissSVM [9]	87.6	80.0
DD-SVM [4]	85.8	**91.3**
MILES [11]	86.3:[84.9,87.7]	87.7:[86.3,89.1]
MILD_I0	87.9:[87.3,88.5]	84.4:[83.5,85.2]
MILD_I	91.1:[90.5,91.7]	89.6:[88.7,90.4]
MILD_B	90.2:[89.5,90.9]	86.5:[85.8,87.1]

4.4.2 Identification of TrX Proteins

Ray et al. [2] empirically compared several well-known MIL methods and their single-instance counterparts on the TrX data set. All these methods are briefly described in Table 3.

TABLE 3
MIL methods and their single-instance counterparts evaluated by Ray et al [2]. The single-instance counterparts are in italics.

Method	Description
Gauss	The single-instance counterpart of DD.
DD(1)	DD(k) refers to diverse density with k disjuncts (i.e., different concept points) $\{c_1, \ldots, c_k\}$, where the probability that an instance belongs to a concept is computed as follows: $\Pr(B_{ij} \in \{c_1, \ldots, c_k\}) \propto \mathrm{softmax}(\Pr(B_{ij} \in c_1), \ldots, \Pr(B_{ij} \in c_k))$. DD(1) is the same as the original DD method.
DD(3)	Diverse density with 3 disjuncts.
DD(5)	Diverse density with 5 disjuncts.
FOIL	A relational learner [39] defining a target relation over instances by assuming that all instances from a positive bag in the training set satisfy this relation while none from the negative bags does.
MI/FOIL	The MI counterpart of FOIL that defines the target relation over bags.
LR	The ordinary logistic regression by assuming that all instances in the positive bags are positive.
MI/LR	MI counterpart of LR, which adopts LR to estimate the probability of the instances being positive, and then computes the probability of a bag to be positive by applying softmax to combine the probabilities of the instances.
SVM(Stat)	SVM with the minimax kernel [6].
SVM(NSK)	SVM with the normalized set kernel [6].
SVM(Quad)	The single-instance counterpart of SVM(Stat) and SVM(NSK), which assumes that every instance in a bag has the same label as that of the bag, and then trains an SVM with a quadratic kernel to classify the instances.

We use the same experimental settings as those in [2] to evaluate MILD on the TrX data set. We adopt the AUC value, which is also used in [2], as the performance measure. The result is reported in Table 4, in which the AUCs of the methods in Table 3 are also included for comparison. Our method, MILD_B, achieves the best result on this data set. The performance of MILD_I is comparable to the best result. In particular, compared with the similar methods using SVM as classifier, both MILD_I and MILD_B achieve far better results, verifying once

again that our disambiguation method is very effective. Furthermore, it is meaningful to compare our instance-level classifier with *SVM(Quad)* because the difference between them is that our method tries to disambiguate the labels of the instances in the positive bags while *SVM(Quad)* simply assumes all instances in the positive bags to be positive. The performance of MILD_I is far better than that of *SVM(Quad)*, showing that disambiguation is very necessary and significant for MIL.

TABLE 4
Protein classification results.

Method	AUC
Gauss	0.340
DD(1)	0.805
DD(3)	0.797
DD(5)	0.828
FOIL	0.721
MI/FOIL	0.543
LR	0.720
MI/LR	0.752
SVM(Quad)	0.584
SVM(Stat)	0.550
SVM(NSK)	0.716
MILD_I	0.797
MILD_B	**0.850**

4.4.3 Localized Content-Based Image Retrieval

We evaluate our methods on the SIVAL image set for localized content-based image retrieval [33]. For each category, we use the *one-versus-the-rest* strategy to evaluate the performance. We randomly select eight positive and eight negative images to form the training set and the remaining 1,484 images to form the test set. The results are reported based on 30 rounds of independent test. Because the existential assumption is not satisfied here, we only report the result of MILD_B. Based on the results in Section 4.3, we simply set $K = 3$. The average AUC values with 95% confidence interval for the 25 categories are reported in Table 5, in which

ACCIO! [33] uses an ensemble of hypotheses learned by EM-DD [23] for classification. We can find that our method achieves the best performance, which verifies our claim that the feature representation scheme for MILD_B can encode the collective contribution of a set of instances in a bag.

TABLE 5

Average AUC values (in percent) with 95% confidence interval over 30 rounds of test on the SIVAL image set (The best performance is shown in bold face).

Category ID	MILD_B	MILES	MISSL [5]	ACCIO! [33]
1	**92.3 ± 2.3**	90.2 ± 2.3	90.0 ± 2.1	77.0 ± 3.4
2	**66.3 ± 2.9**	64.5 ± 2.5	51.1 ± 4.4	63.4 ± 3.4
3	66.7 ± 2.8	**68.1 ± 3.1**	62.4 ± 4.3	65.9 ± 3.3
4	70.5 ± 2.8	72.6 ± 2.5	**76.8 ± 5.2**	69.5 ± 3.4
5	82.7 ± 2.5	84.0 ± 2.3	**84.5 ± 0.8**	68.8 ± 2.3
6	80.9 ± 2.5	**81.2 ± 2.7**	69.6 ± 2.5	67.9 ± 2.2
7	**95.2 ± 0.8**	93.7 ± 1.2	88.9 ± 0.7	90.8 ± 1.6
8	91.3 ± 2.0	92.4 ± 0.8	**93.3 ± 0.9**	81.5 ± 3.5
9	74.0 ± 3.3	71.1 ± 3.2	**77.3 ± 4.3**	74.7 ± 3.4
10	**85.9 ± 2.0**	85.3 ± 1.7	78.2 ± 1.6	83.7 ± 1.9
11	75.3 ± 3.0	**77.1 ± 3.1**	73.8 ± 3.4	65.3 ± 1.5
12	**97.9 ± 0.5**	97.1 ± 0.7	97.7 ± 0.3	86.6 ± 3.0
13	**95.1 ± 0.8**	93.9 ± 0.7	90.5 ± 1.1	86.9 ± 1.7
14	**73.6 ± 2.6**	68.2 ± 3.1	51.5 ± 3.3	72.7 ± 2.3
15	80.1 ± 3.2	80.7 ± 2.9	**83.4 ± 2.7**	77.7 ± 2.6
16	**94.0 ± 1.3**	91.2 ± 1.7	80.4 ± 3.5	87.3 ± 3.0
17	**80.9 ± 2.4**	78.7 ± 2.9	68.0 ± 5.2	79.2 ± 2.6
18	**60.4 ± 1.8**	58.2 ± 1.6	50.2 ± 2.1	57.6 ± 2.3
19	**68.3 ± 2.1**	61.7 ± 2.4	61.3 ± 2.8	62.8 ± 1.7
20	76.1 ± 3.0	77.5 ± 2.6	**80.7 ± 2.0**	77.4 ± 3.3
21	80.6 ± 3.3	80.4 ± 2.0	**81.2 ± 1.5**	71.9 ± 2.5
22	**73.4 ± 2.5**	68.7 ± 2.4	70.2 ± 2.9	70.2 ± 3.2
23	74.8 ± 2.2	73.2 ± 3.1	63.2 ± 5.2	**77.5 ± 2.3**
24	92.3 ± 0.9	88.1 ± 2.2	**93.9 ± 0.9**	82.0 ± 2.4
25	66.3 ± 2.2	62.1 ± 2.5	51.6 ± 2.6	**66.7 ± 1.7**
Average	**79.8**	78.4	74.8	74.6

4.5 Performance of Semi-Supervised Methods

We evaluate the co-training method presented in Algorithm 2 on the MUSK1 data sets. Since semi-supervised learning is usually most useful when the number of labeled examples available is small, we use the following settings to evaluate our method. We apply 10-fold cross validation for evaluation, with nine folds for training and one fold for testing. Then, from the training set, we randomly choose 10 positive and 10 negative bags to form the labeled set L and the remaining training bags to form the unlabeled set U. The original co-training method [12] also creates another smaller pool U' for example selection during the training process. But in our implementation, we directly perform example selection from the remaining unlabeled examples rather than U' because the training set is relatively small for the MUSK1 data sets. In Algorithm 2, both n_1 and n_2 are set to 2. The iterative process will stop when there are not enough unlabeled training examples for selection. This 10-fold cross validation process is repeated for 10 rounds of independent test.

The average accuracy of the results over 10 runs of 10-fold cross-validation is shown in Figure 5, in which MILES, MILD_I, MILD_B are supervised methods trained on the initial labeled data only, while Semi_I, Semi_B are the corresponding semi-supervised versions of MILD_I and MILD_B that also make use of the unlabeled data via co-training. It shows that using unlabeled data for training can dramatically improve the classification accuracy.

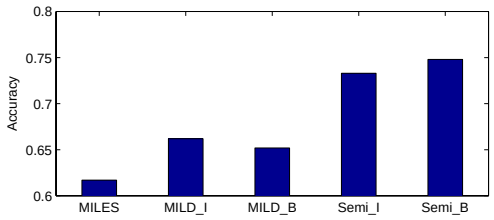


Fig. 5. Performance of semi-supervised MIL.

4.6 Sensitivity to Labeling Noise

We use the same setting as that in MILES [11] to evaluate the noise sensitivity based on the COREL image set [11]. We add $d\%$ of noise by changing the labels of $d\%$ of positive bags and

$d\%$ of negative bags. We compare our methods with DD-SVM and MILES under different noise levels based on 200 images from Category 2 (Historical buildings) and Category 7 (Horses). The training and test sets are of the same size. The average classification accuracy over 20 randomly generated test sets is shown in Figure 6. MILES and all our methods are much more robust than DD-SVM. MILD_I is more sensitive to noise than MILES when the noise level is low ($<25\%$), but its robustness is comparable to MILES when the noise level is high. Actually, it is more meaningful to compare MILD_B with DD-SVM, because the only difference between them is that MILD_B uses our disambiguation method to find some true positive instances while DD-SVM uses DD to find some instance prototypes. The better performance of MILD_B implies that our disambiguation method is more robust than the DD method.

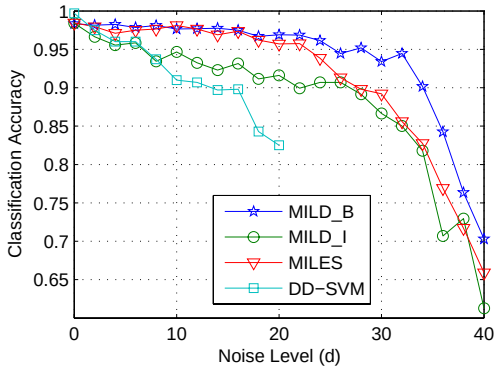


Fig. 6. Sensitivity to labeling noise.

4.7 Efficiency

Compared with other MIL methods, our methods are significantly more efficient in terms of the computation time.

Before applying 1-norm SVM to perform feature selection, MILES needs to compute the

following feature matrix:

$$
\begin{aligned}
&[m_1^+, \ldots, m_{n^+}^+, m_1^-, \ldots, m_{n^-}^-] \\
&= [m(B_1^+), \ldots, m(B_{n^+}^+), m(B_1^-), \ldots, m(B_{n^-}^-)] \\
&= \begin{bmatrix}
s(x_1, B_1^+) & s(x_1, B_2^+) & \cdots & s(x_1, B_{n^-}^-) \\
s(x_2, B_1^+) & s(x_2, B_2^+) & \cdots & s(x_2, B_{n^-}^-) \\
\vdots & \vdots & \ddots & \vdots \\
s(x_n, B_1^+) & s(x_n, B_2^+) & \cdots & s(x_n, B_{n^-}^-)
\end{bmatrix},
\end{aligned} \tag{12}
$$

where $m_i^+ = m(B_i^+)$ ($m_i^- = m(B_i^-)$) is used to denote the feature mapping of B_i^+ (B_i^-), $s(x_k, B_i) = \exp\left(-\frac{d^2(x_k, B_i)}{\sigma^2}\right)$, σ is a scaling coefficient, and $\{x_1, x_2, \ldots, x_n\}$ is the set of re-indexed instances from all the positive and negative training bags.

As for MILD, for each instance from the positive training bags, we compute the distances from the instance to all training bags to compute P^* in (7). This corresponds to one row in the feature matrix of MILES. However, an important difference is that MILD need not compute those rows in (12) corresponding to instances from the negative training bags. From our experiments, we find that the main computation cost of MILD is for the computation of these distances. Hence, MILD is more efficient than MILES even if other costs are not considered. Table 6 shows the computation time required to compute the feature matrix for MILES and that to compute all the distances for MILD on the MUSK data sets. The time for each MUSK data set is the total time for 10-fold cross validation.

Table 7[5] compares the overall computation time required by DD-SVM, MILES and our methods on MUSK data sets. The training time after SVM model selection and the time spent on model selection are reported separately. The result of DD-SVM is from [11], and the training

5. For MILES, the results shown in this table are different from those reported by [11]. Obviously, in their reported results, the "training time after SVM model selection" does not include the time for computing the feature matrix. For fair comparison, we add the time listed in Table 6 to their original results because the reported results for MILD also include all the computation costs.

TABLE 6
Time for computing all the distances (in minutes).

Method	MUSK1	MUSK2
MILES	0.4	160
MILD	**0.2**	**24**

of MILES and our methods were performed on a 2GHz PC. The training time on each MUSK data set is the total training time for 10-fold cross validation. We can find that all our methods are much more efficient than DD-SVM and MILES.

TABLE 7
Computation time comparison (in minutes): training time after SVM model selection + time spent on model selection.

Method	MUSK1	MUSK2
DD-SVM [11]	500 + 112	1500 + 240
MILES	0.5 + 29	161.2 + 129
MILD_B	**0.2 + 0.5**	**24 + 0.7**
MILD_I	**0.4 + 0.8**	**25.6 + 1.1**

A major advantage of our method is that no parameter tuning is needed during the disambiguation process. The only time spent on parameter tuning is for SVM model selection, which is needed for many other methods too. Hence, for our methods, most of the time is spent on the computation of the distances. With our feature representation schemes, the dimensionality of the examples is very low, which directly results in the very low computation cost for SVM training. For example, it only takes about 3 seconds to perform 441 (21 values each for c and g) rounds of 2-fold cross validation on the training set for parameter selection on MUSK1, which implies that the training process for SVM after model selection is almost real-time. Compared with MILD_B, MILD_I has to spend some extra time for the representative negative instance selection defined in (8) and for the instance-level feature mapping defined in (9). Another finding

is that the disambiguation process is very fast after the distance matrix has been computed. It only takes 0.01 minute and 0.05 minute to complete disambiguation for MUSK1 and MUSK2, respectively. The time spent on each MUSK data set is the total time for 10-fold cross validation. This promising efficiency of our disambiguation method is directly resulted from Theorem 2.

5 Conclusion and Future Work

The inherent ambiguity of instance labels in MIL makes MIL problems difficult to solve. In this paper, we propose a very efficient and robust method for MIL via disambiguation. Our method is based on an instance label disambiguation method which exploits the distinguishing properties of true positive instances to identify the true positive instances in the positive bags. Based on the disambiguation method, two feature representation schemes, one for instance classification and the other for bag classification, are proposed to represent the bags from two different views. Moreover, the two views can be integrated into a multi-view learning framework for semi-supervised MIL. We have conducted extensive experimental evaluation on a number of applications to show that our method delivers very promising performance and compares favorably with state-of-the-art MIL methods.

The underlying principle of our method is to try to bridge the gap between multiple-instance learning and single-instance learning. Moreover, our method can be used to perform either instance-level or bag-level classification. In addition, unlike MISSL [5] which is only transductive in nature, our semi-supervised MIL method is inductive, which, to the best of our knowledge, is the first inductive semi-supervised MIL method proposed thus far.

The most attractive properties of our method are highlighted as follows:

- Unlike most existing MIL methods, our method can answer the essential question giving rise to the gap between MIL and standard single-instance learning.
- The classification accuracy of our method is at least comparable with the state-of-the-art MIL methods in diverse applications.
- Our method is much more efficient than existing methods.
- Our method is very robust.

Once we have converted an MIL problem into a single-instance problem, many standard single-instance methods can be applied for MIL based on our feature representation schemes. Besides

the co-training method, many sophisticated semi-supervised learning methods can effectively utilize both labeled and unlabeled data to build better models. Currently, very few MIL methods, except [5], exploit the unlabeled data during model training. With our feature representation schemes, other semi-supervised learning methods such as graph-based methods based on graph Laplacians [40] can also be extended for MIL. This will be our future pursuit.

ACKNOWLEDGMENTS

This research has been supported by General Research Fund 621407 from the Research Grants Council of the Hong Kong Special Administrative Region, China. We thank Dr. Yixin Chen for sharing the code and data for MILES.

REFERENCES

[1] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles." *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[2] S. Ray and M. Craven, "Supervised versus multiple instance learning: an empirical comparison." in *International Conference on Machine Learning*, 2005.

[3] X. Xu, "Statistical learning in multiple instance problem," Master's thesis, University of Waikato, Hamilton, NZ, 2003, 0657.594.

[4] Y. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions." *Journal of Machine Learning Research*, vol. 5, pp. 913–939, 2004.

[5] R. Rahmani and S. A. Goldman, "MISSL: Multiple-instance semi-supervised learning." in *International Conference on Machine Learning*, 2006.

[6] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels." in *International Conference on Machine Learning*, 2002.

[7] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning." in *Advances in Neural Information Processing Systems*, 2002, pp. 561–568.

[8] P.-M. Cheung and J. T. Kwok, "A regularization framework for multiple-instance learning." in *International Conference on Machine Learning*, 2006.

[9] Z.-H. Zhou and J.-M. Xu, "On the relation between multi-instance learning and semi-supervised learning." in *International Conference on Machine Learning*, 2007.

[10] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning." in *Advances in Neural Information Processing Systems*, 1997.

[11] Y. Chen, J. Bi, and J. Z. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1931–1947, 2006.

[12] A. Blum and T. M. Mitchell, "Combining labeled and unlabeled sata with co-training." in *Annual Conference on Learning Theory*, 1998, pp. 92–100.

[13] J. Davis, V. S. Costa, S. Ray, and D. Page, "An integrated approach to feature invention and model construction for drug activity prediction." in *Proceedings of the International Conference on Machine Learning*, 2007.

[14] P. Auer, "On learning from multi-instance examples: Empirical evaluation of a theoretical approach." in *Proceedings of the International Conference on Machine Learning*, 1997, pp. 21–29.

[15] J. Ramon and L. D. Raedt, "Multi-instance neural networks," in *ICML-2000 Workshop on Attribute-Value and Relational Learning*, 2000.

[16] Z.-H. Zhou and M.-L. Zhang, "Multi-instance multi-label learning with application to scene classification," in *Advances in Neural Information Processing Systems*, 2006, pp. 1609–1616.

[17] J. Bi, Y. Chen, and J. Z. Wang, "A sparse support vector machine approach to region-based image categorization." in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1)*, 2005, pp. 1121–1128.

[18] O. Maron and A. L. Ratan, "Multiple-instance learning for natural scene classification." in *Proceedings of the International Conference on Machine Learning*, 1998, pp. 341–349.

[19] Q. Tao, S. Scott, N. V. Vinodchandran, and T. T. Osugi, "SVM-based generalized multiple-instance learning via approximate box counting." in *International Conference on Machine Learning*, 2004.

[20] J. Bi and J. Liang, "Multiple instance learning of pulmonary embolism detection with geodesic distance along vascular structure." in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[21] G. Fung, M. Dundar, B. Krishnapuram, and R. B. Rao, "Multiple instance learning for computer aided diagnosis." in *Advances in Neural Information Processing Systems*, 2006.

[22] G. Ruffo, "Learning single and multiple decision trees for security applications," Ph.D. dissertation, Univ. of Turin, Italy, 2000.

[23] Q. Zhang and S. A. Goldman, "EM-DD: An improved multiple-instance learning technique." in *Advances in Neural Information Processing Systems*, 2001.

[24] Q. Zhang, S. A. Goldman, W. Yu, and J. Fritts, "Content-based image retrieval using multiple-instance learning." in *Proceedings of the International Conference on Machine Learning*, 2002, pp. 682–689.

[25] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[26] P. V. Gehler and O. Chapelle, "Deterministic annealing for multiple-instance learning." in *International Conference on Artificial Intelligence and Statistics*, 2007.

[27] J. T. Kwok and P.-M. Cheung, "Marginalized multi-instance kernels." in *International Joint Conferences on Artificial Intelligence*, 2007.

[28] R. C. Bunescu and R. J. Mooney, "Multiple instance learning for sparse positive bags." in *International Conference on Machine Learning*, 2007.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[30] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[31] S. Rüping and T. Scheffer, "Learning with multiple views." in *Proceedings of ICML Workshop on Learning with Multiple Views*, 2005.

[32] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, "Generic object recognition with boosting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 416–431, 2006.

[33] R. Rahmani, S. A. Goldman, H. Zhang, J. Krettek, and J. E. Fritts, "Localized content based image retrieval," in *Multimedia Information Retrieval*, 2005, pp. 227–236.

[34] F. Perronnin, "Universal and adapted vocabularies for generic visual categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1243–1256, 2008.

[35] C.-C. Chang and C.-J. Lin, *LIBSVM: a Library for Support Vector Machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

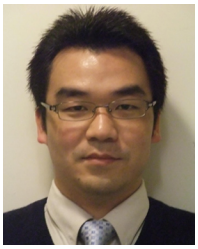
[36] C. Wang, S. D. Scott, J. Zhang, Q. Tao, D. E. Fomenko, and V. N. Gladyshev, "A study in modeling low-conservation protein superfamilies." University of Nebraska, Tech. Rep. UNL-CSE-2004-0003, 2004.

[37] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning." in *Advances in Neural Information Processing Systems*, 2007.

[38] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, pp. 1470–1477.

[39] J. R. Quinlan, "Learning logical definitions from relations." *Machine Learning*, vol. 5, pp. 239–266, 1990.

[40] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph Laplacians for semi-supervised learning." in *Advances in Neural Information Processing Systems*, 2005.



Wu-Jun Li received his B.Sc. and M.Eng. degrees in Computer Science from Nanjing University, China, in 2003 and 2006, respectively.

He is currently a Ph.D. student at the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. His research interests are in statistical machine learning, pattern recognition and data mining.

Dit-Yan Yeung received his B.Eng. degree in Electrical Engineering and M.Phil. degree in Computer Science from the University of Hong Kong, and Ph.D. degree in Computer Science from the University of Southern California. He started his academic career as an Assistant Professor at the Illinois Institute of Technology. He then joined the Hong Kong University of Science and Technology where he is now a Professor of Computer Science and Engineering. His research interests are in computational and statistical approaches to machine learning, pattern recognition and data mining.