Multi-Instance Learning with Key Instance Shift[*]

Ya-Lin Zhang and Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology, Nanjing University
Collaborative Innovation Center of Novel Software Technology and Industrialization
Nanjing 210023, China
{zhangyl, zhouzh}@lamda.nju.edu.cn

Abstract

Multi-instance learning (MIL) deals with the tasks where each example is represented by a bag of instances. A bag is positive if it contains at least one positive instance, and negative otherwise. The positive instances are also called key instances. Only bag labels are observed, whereas specific instance labels are not available in MIL. Previous studies typically assume that training and test data follow the same distribution, which may be violated in many real-world tasks. In this paper, we address the problem that the distribution of key instances varies between training and test phase. We refer to this problem as MIL with key instance shift and solve it by proposing an embedding based method MIKI. Specifically, to transform the bags into informative vectors, we propose a weighted multi-class model to select the instances with high positiveness as instance prototypes. Then we learn the importance weights for transformed bag vectors and incorporate original instance weights into them to narrow the gap between training/test distributions. Experimental results validate the effectiveness of our approach when key instance shift occurs.

1 Introduction

Multi-instance learning (MIL) [Herrera *et al.*, 2016] deals with the tasks where each example is represented by a bag of instances. A bag is positive if it contains at least one positive instance, and negative otherwise. The instances that trigger the positive labels are also called key instances. Only bag labels are observed, whereas instance labels are unknown in MIL. MIL has been applied in many domains, including image classification [Zhang *et al.*, 2002], text categorization [Andrews *et al.*, 2002], and web mining [Zhou *et al.*, 2005], etc.

Previous studies of MIL typically assume that the training and test data are drawn from the same distribution, which may be violated in many real-world tasks. The distribution discrepancy may arise when training and test data are collected in different time, location or with different labeling cost, etc. Take text categorization for example, supposing that we want to judge whether a text is about 'sport' and the training/test data are collected in 2012/2016, respectively. Then compared with training data, test data may have less texts about 'Kobe Bryant', but more about 'Stephen Curry'. Here the paragraphs containing 'Kobe Bryant' or 'Stephen Curry' can be regarded as key instances. In this case, the distribution of key instances changes between training and test phase due to different collection time. Also, if the training/test data are collected in America/China respectively, then compared with training data, test data may have less texts about 'baseball', but more about 'table tennis'. Here the paragraphs with 'baseball' or 'table tennis' can be regarded as key instances. The distribution of key instances varies due to different collection locations. In this paper, we formalize this problem as MIL with key instance shift and propose the MIKI (Multi-Instance with Key Instance shift) method to solve it.

MIKI follows the embedding based manner [Chen *et al.*, 2006], which transforms bags into single instance vectors via embedding. Instance prototypes are selected to encode both bag-level information and key instance shift information. First, a weighted multi-class model is trained to select the instances with high positiveness as instance prototypes, and the bags are transformed into informative vector representations with selected instance prototypes. Then, we learn the weights for the transformed bag vectors and incorporate the original instance weights into them to narrow the gap between training/test distributions. Finally, a model is learned by using the new representations and their weights. Experiments on different data sets demonstrate the effectiveness of our approach when key instance shift occurs. Even without key instance shift, MIKI still works comparable with the state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed method. Section 4 reports the experimental results. Section 5 applies the proposed method to moving object localization problem and Section 6 concludes.

2 Related Work

Multi-instance learning was first proposed in [Dietterich *et al.*, 1997]. Since then, many approaches have been proposed, which mainly fall into two categories: some try to directly solve the MIL problem in either instance level [Maron and Lozano-Pérez, 1998; Li *et al.*, 2009; Faria *et al.*, 2017] or bag

[*]This research was supported by the NSFC (61333014) and Huawei Fund (YBN2017030027).

level [Gärtner *et al.*, 2002; Zhou *et al.*, 2009], while the others transform MIL into single instance learning via embedding, among which MILES [Chen *et al.*, 2006] is a typical representative, and many methods [Amores, 2015; Yuan *et al.*, 2016; Wei *et al.*, 2017] have been proposed recently following this manner. It is noteworthy that previous MIL methods typically assume that the training and test data follow the same distribution. In contrast, we consider the key instance shift problem, which has not been thoroughly studied to our best knowledge.

In addition, due to the differences in gathering data, distribution change between training and test data may always arise, among which covariate shift [Shimodaira, 2000] has attracted much attention. In covariate shift, the input distribution $P(x)$ is different between training and test phase but the conditional distribution of output $P(y|x)$ remains unchanged. A common approach for covariate shift is importance reweighting, which assigns each training instance x with a weight $w(x) = p_{te}(x)/p_{tr}(x)$ to diminish the discrepancy of training and test marginals by some criteria [Sugiyama *et al.*, 2012]. Note that test data is needed to estimate the difference of distribution. Many methods have been proposed to estimate the weight, such as KMM [Huang *et al.*, 2006], KLIEP [Sugiyama *et al.*, 2008], LSIF and uLSIF [Kanamori *et al.*, 2009], etc. However, all of these studies focus on single-instance learning.

The most related literature of our work is [Zhang and Zhou, 2014], which considers the setting when MIL encounters distribution change and proposes the MICS method. In contrast to considering situations where the bag label is decided by multiple instances and their relations [Zhang and Zhou, 2014], in this paper, we follow the standard MIL setting that bag labels are determined by the key instances, and consider the distribution change problem of key instances.

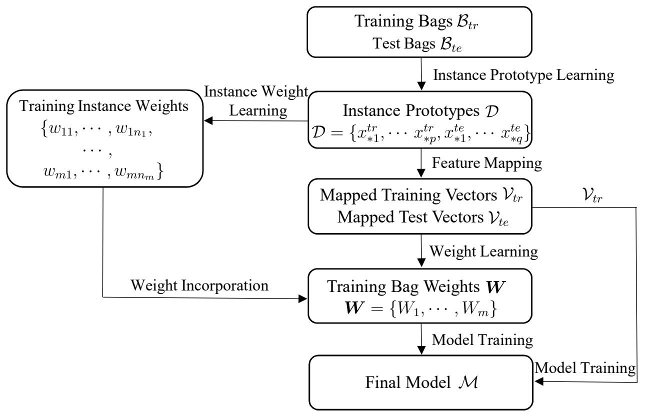
3 MIL with Key Instance Shift

3.1 Problem Statement and Notations

Let $\mathcal{X} = \mathbb{R}^d$ denote the instance space and $\mathcal{Y} = \{-1, +1\}$ denote label space. The learner is given a data set with m training bags $\mathcal{B}_{tr} = \{(X_1^{tr}, y_1), \ldots, (X_i^{tr}, y_i), \ldots, (X_m^{tr}, y_m)\}$, where $X_i = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{in_i}\}$ is a bag with $\boldsymbol{x}_{ij} \in \mathcal{X}$ representing an instance. If there exists any positive instance in X_i, then X_i is a positive bag with $y_i = +1$; otherwise X_i is negative with $y_i = -1$. Those positive instances are also called key instances, which trigger the positive labels. Note that only bag labels are available, yet specific instance labels are unknown. The learner is also given a set of n unlabeled test bags $\mathcal{B}_{te} = \{X_1^{te}, \ldots, X_n^{te}\}$. In MIL with key instance shift problem, the distributions of key instances in training and test phase are different. Let $P(\cdot)$ denote the distribution, $\boldsymbol{x}_*$ refer to the key instances, then we have $P(\boldsymbol{x}_*^{tr}) \neq P(\boldsymbol{x}_*^{te})$ in this setting. In this work, we focus on transductive setting: given $\mathcal{B}_{tr}$ and $\mathcal{B}_{te}$, where $P(\boldsymbol{x}_*^{tr}) \neq P(\boldsymbol{x}_*^{te})$, our goal is to learn a mapping $f : 2^{\mathcal{X}} \to \mathcal{Y}$ that works fine for test data.

3.2 Proposed Method

MIKI follows the embedding based manner [Chen *et al.*, 2006], which transforms bags into single instance vectors via embedding. To handle the key instance shift problem, we try to select key instance candidates from both training and test sets as instance prototypes so that both bag-level information and key instance shift information can be encoded. Specifically, assuming that the positive instances belong to different sub-concepts, then key instance shift may fall into any of these sub-concepts. Supposing that there are K sub-concepts, together with the negative concept, we build a $(K+1)$-class model to distinguish different sub-concepts and the negative concept. If an instance belongs to any sub-concept with a high score, it would be regarded as a key instance candidate with high probability. In this way, different shift information can be captured. Moreover, in order to alleviate the influence of key instance shift, we introduce the instance weights to balance different distributions (i.e., higher/lower weights for underrepresented/overrepresented instances), so that the selected instance prototypes may include more key instances and encode more key instance shift information. Similarly, after getting the transformed bag vectors, we learn the weights for them to narrow the gap of training/test distributions. Note that if an instance is important, the bag containing it should also be valued, so we further incorporate weights of the original instances into the weights of the transformed bag vectors. The final model is learned with the transformed bag vectors and their weights. Figure 1 summarizes our MIKI approach.



Training Bags $\mathcal{B}_{tr}$
Test Bags $\mathcal{B}_{te}$
Instance Prototype Learning
Instance Weight Learning
Training Instance Weights $\{w_{11}, \cdots, w_{1n_1}, \cdots, w_{m1}, \cdots, w_{mn_m}\}$
Instance Prototypes $\mathcal{D}$
$\mathcal{D} = \{x_{*1}^{tr}, \cdots x_{*p}^{tr}, x_{*1}^{te}, \cdots x_{*q}^{te}\}$
Feature Mapping
Mapped Training Vectors $\mathcal{V}_{tr}$
Mapped Test Vectors $\mathcal{V}_{te}$
$\mathcal{V}_{tr}$
Weight Learning
Weight Incorporation
Training Bag Weights $\boldsymbol{W}$
$\boldsymbol{W} = \{W_1, \cdots, W_m\}$
Model Training
Model Training
Final Model $\mathcal{M}$

Figure 1: Block diagram for the proposed method.

Instance Prototype Learning

In instance prototype learning phase, a weighted multi-class model is trained to select key instance candidates as instance prototypes. We use an iterative framework to alternately build the multi-class model and optimize the selected key instance candidates, and then update the instance weights[1]. Algorithm 1 summarizes the instance prototype learning framework.

Specifically, in each iteration, a weighted $(K+1)$-class model $\mathcal{M}_D$ is trained for the K sub-concepts of positive instances and the negative concept. If an instance belongs to any sub-concept with a high score, it would be regarded with high positiveness. To represent the positiveness, we denote the probability of instance $\boldsymbol{x}_{ij}$ belonging to class k as Pr_{ij}^k, and calculate the positiveness score S_{ij} for each instance $\boldsymbol{x}_{ij}$:

$$S_{ij} = \max_{k \in \{1,\ldots,K\}} (Pr_{ij}^k - Pr_{ij}^0) \,. \qquad (1)$$

[1]The importance weighting technique is same to the one introduced in the next section.

Algorithm 1 Instance Prototype Learning for MIKI

```
Input: Training bags $\mathcal{B}_{tr} = \{(X_i^{tr}, y_i)\}_{i=1}^m$, Testing bags $\mathcal{B}_{te} = \{X_j^{te}\}_{j=1}^n$, Instance selection parameters $n_\theta^{tr}$ and $n_\theta^{te}$.
Output: Instance Prototype Set $\mathcal{D}$.
1: Initialize all training instance weights $w$.
2: repeat
3:     Train multi-class model $\mathcal{M}_D$ by Algorithm 2;
4:     Calculate positiveness score for all instances by Eq. 1;
5:     Separate instances into $C_p^{tr}$, $C_n^{tr}$, $C_p^{te}$ and $C_n^{te}$ by Eq. 4;
6:     Reestimate the weights $\boldsymbol{w}^p$ ($\boldsymbol{w}^n$) for instances in $C_p^{tr}$ ($C_n^{tr}$)
       to approximate instances in $C_p^{te}$ ($C_n^{te}$);
7: until max iteration is reached.
8: Output $\mathcal{D} = C_{tr}^p \cup C_{te}^p$.
```

Algorithm 2 Learning algorithm for multi-class model $\mathcal{M}_D$

```
Input: Training bags $\mathcal{B}_{tr} = \{(X_i^{tr}, y_i)\}_{i=1}^m$, Training instance weights $w$, Positive cluster number $K$.
Output: Multi-class model $\mathcal{M}_D$
1: Initialize the labels for all training instances;
2: repeat
3:     Train multi-class model with the weighted training instances;
4:     Update the labels of all instances from positive bags by Eq. 2;
5:     For each positive bag, if all instances are labeled negative,
       select the most positive one and adjust its label by Eq. 3;
6: until no label changes or max iteration is reached.
7: Output the final model as $\mathcal{M}_D$.
```

Algorithm 2 shows the details of learning $\mathcal{M}_D$. We first run k-means on all instances from positive bags and initialize their labels as the associated cluster numbers, and assign all instances from negative bags with label 0. A multi-class SVM model $\mathcal{M}$ is trained with the weighted instances and their assigned labels. Given an instance $\boldsymbol{x}_{ij}$, the model $\mathcal{M}$ will output the probability Pr_{ij}^k of $\boldsymbol{x}_{ij}$ belonging to class k. Then for each instance $\boldsymbol{x}_{ij}$ from positive bags, its label is updated to be

$$y_{ij} = \underset{k\in\{0,1,\ldots,K\}}{\arg\max}\; Pr_{ij}^k\,. \tag{2}$$

The labels of instances in negative bags stay unchanged. For each positive bag X_i, if all instances are labeled negative, the label of the instance (denote as $\boldsymbol{x}_{ip}$) with the highest positiveness score is adjusted to be

$$y_{ip} = \underset{k\in\{1,\ldots,K\}}{\arg\max}\; Pr_{ip}^k\,, \tag{3}$$

so that each positive bag contains at least one positive instance and the initialized false positives are calibrated through iteration. The step sequence interleaves until no label changes or max iteration is reached, then the model $\mathcal{M}_D$ is obtained.

With the obtained model $\mathcal{M}_D$, the positiveness score of each instance is estimated and key instance candidates are selected. Specifically, let S_θ^{tr} (S_θ^{te}) denote the n_θ^{tr}-th (n_θ^{te}-th) highest score of all training (test) instances, we then separate the instances into key instance candidates and negative instance candidates according to their positiveness scores:

$$\begin{aligned}
C_p^{tr} &= \{\boldsymbol{x}_{ij} | S_{ij} \geq S_\theta^{tr}, \boldsymbol{x}_{ij} \in \mathcal{B}_{tr}\}\,,\\
C_n^{tr} &= \{\boldsymbol{x}_{ij} | S_{ij} < S_\theta^{tr}, \boldsymbol{x}_{ij} \in \mathcal{B}_{tr}\}\,,\\
C_p^{te} &= \{\boldsymbol{x}_{ij} | S_{ij} \geq S_\theta^{te}, \boldsymbol{x}_{ij} \in \mathcal{B}_{te}\}\,,\\
C_n^{te} &= \{\boldsymbol{x}_{ij} | S_{ij} < S_\theta^{te}, \boldsymbol{x}_{ij} \in \mathcal{B}_{te}\}\,.
\end{aligned} \tag{4}$$

Here, C_p^{tr} and C_p^{te} are those instances with higher positiveness scores, so they are more likely to be potential key instances and carry the key instance shift information.

After the separation, we reestimate the weights for training instances in key (negative) instance candidates correspondingly, i.e., reestimate the weights for C_p^{tr} (C_n^{tr}) by using C_p^{te} (C_n^{te}). Thus, all training instance weights are updated. Then the model $\mathcal{M}_D$ will be trained again with updated weights and the key instance candidates will be updated. This step sequence interleaves until the max iteration is reached, and the selected instance prototypes (i.e., $C_p^{tr} \cup C_p^{te}$) are outputted as instance prototype set $\mathcal{D}$.

With $\mathcal{D}$, we map both training and test bags into new vector representations following the typical procedure [Chen *et al.*, 2006]. Concretely, each bag X_i is mapped into a $(n_\theta^{tr} + n_\theta^{te})$-dimensional vector $\boldsymbol{v}_i$, and the j-th attribute v_{ij} is

$$v_{ij} = \max_{\boldsymbol{x}_{ik}\in X_i} \exp(-\gamma \|\boldsymbol{x}_{ik} - D_j\|^2)\,, \tag{5}$$

where D_j is the j-th term of $\mathcal{D}$. The new representations keep both bag-level information and key instance shift information.

Weight Learning

In this section, we present the details of learning the weights for transformed training vectors. Given transformed training vectors $\{\boldsymbol{v}_i^{tr}\}_{i=1}^m$ with density $p_{tr}(\boldsymbol{v})$ and transformed test vectors $\{\boldsymbol{v}_i^{te}\}_{i=1}^n$ with density $p_{te}(\boldsymbol{v})$, we want to estimate the importance $w(\boldsymbol{v}) = p_{te}(\boldsymbol{v})/p_{tr}(\boldsymbol{v})$ for all transformed training vectors. Following [Kanamori *et al.*, 2009], we model the weight $w(\boldsymbol{v})$ by the following linear model:

$$\hat{w}(\boldsymbol{v}) = \sum_{l=1}^{b} \alpha_l \varphi_l(\boldsymbol{v})\,, \tag{6}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_b)^T$ are parameters to be learned and $\{\varphi_l(\boldsymbol{v})\}_{l=1}^b$ are basis functions satisfying $\varphi_l(\boldsymbol{v}) \geq 0$ for all $\boldsymbol{v}$ and l. Here we choose a fixed number of Gaussian kernels centered at test points as the basis functions. We need to estimate the parameters $\{\alpha_l\}_{l=1}^b$ to make $\hat{w}(\boldsymbol{v})$ approximate to $w(\boldsymbol{v})$. To achieve this, we minimize the following squared error $L_0(\boldsymbol{v})$:

$$\begin{aligned}
L_0(\boldsymbol{v}) =& \frac{1}{2}\int (\hat{w}(\boldsymbol{v}) - w(\boldsymbol{v}))^2 p_{tr}(\boldsymbol{v}) d\boldsymbol{v}\\
=& \frac{1}{2}\int (\hat{w}(\boldsymbol{v})^2 p_{tr}(\boldsymbol{v}) - 2\hat{w}(\boldsymbol{v}) w(\boldsymbol{v}) p_{tr}(\boldsymbol{v})\\
&+ w(\boldsymbol{v})^2 p_{tr}(\boldsymbol{v})) d\boldsymbol{v}\,,
\end{aligned} \tag{7}$$

since the last term in Eq. 7 is a constant given the training set, it can be safely ignored. Using $p_{te}(\boldsymbol{v}) = w(\boldsymbol{v}) p_{tr}(\boldsymbol{v})$, we denote the first two terms by $L(\boldsymbol{v})$:

$$\begin{aligned}
L(\boldsymbol{v}) =& \frac{1}{2}\int \hat{w}(\boldsymbol{v})^2 p_{tr}(\boldsymbol{v}) d\boldsymbol{v} - \int \hat{w}(\boldsymbol{v}) p_{te}(\boldsymbol{v}) d\boldsymbol{v}\\
=& \frac{1}{2}\sum_{l,l'=1}^{b} \alpha_l \alpha_{l'} (\int \varphi_l(\boldsymbol{v}) \varphi_{l'}(\boldsymbol{v}) p_{tr}(\boldsymbol{v}) d\boldsymbol{v})\\
&- \sum_{l=1}^{b} \alpha_l (\int \varphi_l(\boldsymbol{v}) p_{te}(\boldsymbol{v}) d\boldsymbol{v})\,.
\end{aligned} \tag{8}$$

Approximating to the expectations in $L(\boldsymbol{v})$ by empirical averages, we obtain $\hat{L}(\boldsymbol{v})$:

$$\begin{aligned}\hat{L}(\boldsymbol{v}) =& \frac{1}{2}\sum_{l,l'=1}^{b}\alpha_l\alpha_{l'}\Big(\frac{1}{m}\sum_{i=1}^{m}\varphi_l(\boldsymbol{v}_i^{tr})\varphi_{l'}(\boldsymbol{v}_i^{tr})\Big) \\ &-\sum_{l=1}^{b}\alpha_l\Big(\frac{1}{n}\sum_{j=1}^{n}\varphi_l(\boldsymbol{v}_j^{te})\Big) \\ =& \frac{1}{2}\boldsymbol{\alpha}^T\hat{H}\boldsymbol{\alpha} - \hat{\boldsymbol{h}}^T\boldsymbol{\alpha}\,,\end{aligned} \tag{9}$$

where $\hat{H}$ is the $b\times b$ matrix with the (l,l')-th element $\hat{H}_{l,l'} = \sum_{i=1}^{m}\varphi_l(\boldsymbol{v}_i^{tr})\varphi_{l'}(\boldsymbol{v}_i^{tr})/m$ and $\hat{\boldsymbol{h}}$ is the b-dimensional vector with the l-th element $\hat{h}_l = \sum_{j=1}^{n}\varphi_l(\boldsymbol{v}_j^{te})/n$. Based on Eq. 9, we get the following optimization problem:

$$\min_{\alpha\in\mathbb{R}^b}\Big[\frac{1}{2}\boldsymbol{\alpha}^T\hat{H}\boldsymbol{\alpha} - \hat{\boldsymbol{h}}^T\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}\Big]\,. \tag{10}$$

Eq. 10 is an unconstrained convex quadratic program, so the solution can be analytically computed as $\boldsymbol{\alpha} = (\hat{H}+\lambda I_b)^{-1}\hat{\boldsymbol{h}}$, where I_b is the b-dimensional identity matrix, thus the parameters can be efficiently estimated and the weight of each training sample can be obtained. This importance weighting technique is exactly the one used in Algorithm 1 with $\boldsymbol{v}$ replaced by $\boldsymbol{x}$.

With the aforementioned method, we can get the weights for all transformed training vectors, and we denote the weights as $\boldsymbol{W}^0$. Moreover, we address that if a bag contains an instance that should be valued, the bag should be valued too, thus we further incorporate instance weights into the transformed bag weights. Note that when we learn the instance prototypes, the instances are separated into C_p^{tr} (C_p^{te}) and C_n^{tr} (C_n^{te}) according to their positiveness. We follow the same procedure to learn the weights for instances in C_p^{tr} (C_n^{tr}) to approximate the instances in C_p^{te} (C_n^{te}), separately. Thus, all training instance weights are estimated and they are further integrated into the transformed bag weights. Concretely, for training bag X_i^{tr} and its transformed vector $\boldsymbol{v}_i^{tr}$, denote the weight learned from the transformed bag vectors as W_i^0 and the weights for instances in this bag as $\{w_{ij}\}_{j=1}^{n_i}$, the final weight of $\boldsymbol{v}_i^{tr}$ is set to be:

$$W_i = \max_{j\in\{1,\dots,n_i\}} w_{ij}\cdot W_i^0\,. \tag{11}$$

At the end of MIKI, we use the transformed training feature vectors and their corresponding weights to learn the classification model, which is employed it to classify the transformed test feature vectors.

Note that the number of selected instance prototypes in MIKI is much smaller than that used in MILES and many other embedding based methods, and thus, MIKI is more efficient in this aspect. The iterative instance prototype learning process of MIKI is similar to miSVM [Andrews *et al.*, 2002], and thus, mechanisms for time cost control and acceleration for miSVM can be applied.

4 Experiments

The experiments are performed on both synthetic data and real world datasets. We compare the proposed method with many state-of-the-art algorithms, including instance-level methods miSVM and MISVM [Andrews *et al.*, 2002], and bag-level methods miGraph [Zhou *et al.*, 2009], MILES [Chen *et al.*, 2006] and MILDE [Amores, 2015]. We also derive three variants for comparison. The first one miSVM+, works by estimating the weights for all instances, and incorporating them directly to miSVM. The second one MILES+, is a variant of MILES and the weights are obtained based on the transformed feature vectors. In addition, since we have used instances from test bags in MIKI, we derive the third variant MILES++, using all instances from both training and test bags for feature mapping with importance weighting technique incorporated. Furthermore, we also compare with MICS [Zhang and Zhou, 2014], which focuses on the setting of MIL with distribution change. RBF kernel is used for all SVM-based methods. For MIKI, we simply set K to 5 for synthetic dataset and 10 for the other datasets without any tuning, and set max iteration to 5 to accelerate the method. Other parameters are selected via 5-fold cross validation.

4.1 Experiments on Synthetic Data

We first perform experiments on synthetic dataset. We generate two sub-concepts of positive instances which are represented by Gaussian distribution $P_1 = \mathcal{N}(\mu_1,\sigma^2)$ and $P_2 = \mathcal{N}(\mu_2,\sigma^2)$, and one negative concept represented by $P_3 = \mathcal{N}(\mu_3,\sigma^2)$. Here we set $\mu_1 = (1,1)$, $\mu_2 = (2,2)$, $\mu_3 = (3.5,2.5)$, $\sigma^2 = (0.5,0.5)$. Each bag contains 10 instances, and each positive bag contains 1 positive instance, which is from one of the sub-concepts (i.e., P_1 or P_2). We generate 200 negative and 200 positive bags. Half of the positive bags contain positive instances from one sub-concept, and the other half from another.

We follow the typically deliberately biased sampling procedure [Zadrozny, 2004] to separate the bags into disjoint training and test sets. Specifically, we define a random variable s_i for each bag, where $s_i = 1$ means that the i-th bag is selected into training set, and $s_i = 0$ otherwise. For positive bag X_i, denote the key instance as $\boldsymbol{x}_{i*}$, then the positive bags are sampled following the rules below:

$$\begin{aligned}&Pr(s_i = 1|\boldsymbol{x}_{i*}\in SC_1) = a\,,\\ &Pr(s_i = 1|\boldsymbol{x}_{i*}\in SC_2) = b\,.\end{aligned} \tag{12}$$

Here, SC_1 and SC_2 denote different sub-concepts (i.e.,P_1 and P_2) and $a = 0.8$, $b = 0.2$. In other words, compared with test data, there are more positive bags in training data containing positive instances from sub-concept P_1, but less from P_2.

We repeat experiments for 30 times by using the sampling procedure to generate training and test sets. As shown in Table 1, our method outperforms all other ones. Though distribution change is considered, miSVM+, MILES+ and MILES++ do not show significant improvement, verifying that simply applying single-instance distribution shift approaches to MIL does not work. MICS does not get favorable result too.

Furthermore, we examine the influence of parameter K. In MIKI, we simply set K to 5, but not the ground-truth number of sub-concepts. The results in Figure 2 show that the behavior tends to get better as the value of K getting closed to the ground-truth. However, our method is not that sensitive to K. Moreover, we observe that almost 88% of the selected instance

Table 1: Testing accuracy (%, mean ± std.) on synthetic dataset. The highest average accuracy is marked in bold.

	miSVM	MISVM	miGraph	MILES	MILDE	miSVM+	MILES+	MILES++	MICS	MIKI
Accuracy	77.2±1.7	82.6±1.3	81.4±1.9	83.4±1.9	83.7±1.4	80.3±1.4	83.0±2.0	83.0±2.1	78.8±4.5	**88.3±1.1**

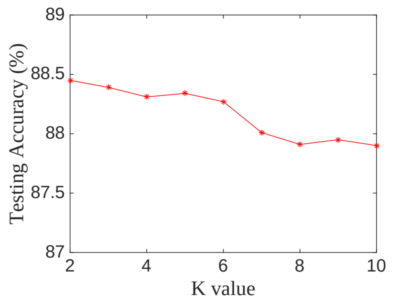


Figure 2: Testing accuracy with different K values.

prototypes of MIKI are positive, verifying the effectiveness of selecting 'positive' instances in MIKI.

4.2 Experiments on Text Categorization

We then perform experiments on text categorization tasks. We derive 25 data sets from the popularly used *20 Newsgroups* corpora. The documents belong to 6 main classes and 20 sub-classes. For each dataset, we set one main class (e.g., 'comp') as positive, and the others as negative. For positive class, two sub-classes (e.g., 'comp.sys.ibm.pc' and 'comp.windows.x') are used as different sub-concepts. 200 positive and 200 negative bags are generated. Each positive bag contains averagely 1.5 instances from positive class, and the negative instances are randomly and uniformly drawn from the negative classes. Half of the positive bags contain positive instances from one sub-concept, and the other half from another sub-concept. An instance is a post represented by 100-dimensional LDA feature [Blei *et al.*, 2003].

Similarly, the biased sampling procedure is used to separate bags. Eq. 12 is used to sample the bags, with SC_1 and SC_2 being different sub-classes, and $a = 0.8$, $b = 0.2$. Thus, compared with test data, there are more positive instances in training bags from SC_1, but less from SC_2.

We repeat experiments for 30 times by using the sampling procedure to generate training and test sets. The results are reported in Table 2, with the win/tie/loss counts in the last row. As we can see, MIKI achieves 22 times the best performance, which is significantly better than other algorithms. MICS does not work well because as we have mentioned in the end of related work, it is designed to handle other types of distribution shift rather than key instance shift. It is interesting to see that miSVM+ and MILES+/MILES++ are not necessarily better

Table 2: Testing accuracy (%, mean ± std.) on text categorization tasks. The highest average accuracy is marked in bold. The last row shows the win/tie/loss counts of MIKI versus other methods (paired *t*-tests at 95% significance level).

	Instance-level method		Bag-level method			Variant of state-of-the-art method and distribution change method				Proposed method
	miSVM	MISVM	miGraph	MILES	MILDE	miSVM+	MILES+	MILES++	MICS	MIKI
comp_gra_ibm	64.0±5.1	59.9±7.6	69.9±4.7	67.7±3.8	68.9±3.7	63.7±4.9	67.4±4.1	67.6±4.1	69.7±4.8	**74.1±5.2**
comp_gra_mac	62.8±4.6	61.1±8.0	65.7±3.4	69.1±4.5	67.7±3.8	62.9±4.4	68.2±5.0	69.6±4.1	65.1±3.0	**71.1±4.9**
comp_gra_os	79.0±3.0	63.5±10.9	75.0±3.3	78.7±2.9	80.1±2.8	79.1±3.0	78.7±3.0	78.6±3.1	74.4±3.3	**82.5±3.3**
comp_gra_win	76.0±4.7	62.2±7.1	70.9±3.1	75.2±3.0	76.7±3.2	76.0±4.3	75.3±3.3	75.8±3.5	69.7±5.3	**79.0±3.5**
comp_ibm_mac	73.6±5.1	62.4±10.7	72.2±3.6	77.0±3.0	78.1±2.8	73.9±5.0	76.9±2.7	77.2±3.2	72.2±3.9	**79.2±2.7**
comp_ibm_win	69.6±4.5	56.0±5.2	68.7±4.0	69.3±3.3	65.9±4.0	70.2±4.2	66.4±4.5	68.4±4.3	68.9±4.0	**77.9±4.7**
comp_mac_win	67.9±6.7	57.9±5.3	65.0±3.9	63.7±4.4	63.1±3.3	66.5±7.9	63.9±5.1	67.4±4.2	65.0±3.8	**76.5±4.8**
comp_os_ibm	70.1±5.2	62.1±9.0	73.6±3.7	71.6±2.8	71.7±4.5	69.8±5.3	71.6±3.0	72.0±3.7	74.0±3.6	**79.3±3.5**
comp_os_mac	68.3±6.7	62.2±9.6	70.4±3.4	73.0±3.5	74.0±3.6	67.5±7.1	72.9±3.7	73.1±3.8	69.9±3.8	**76.9±3.0**
comp_os_win	77.3±3.7	62.8±9.8	71.8±3.7	75.1±3.1	76.5±2.7	77.4±3.4	75.8±3.2	75.1±2.8	71.9±3.7	**79.8±4.0**
rec_auto_baseball	68.8±9.8	58.0±5.4	63.0±5.1	63.2±6.3	58.7±3.8	69.1±9.6	60.1±5.9	69.8±5.4	62.2±4.8	**81.1±4.4**
rec_auto_hockey	78.4±9.9	61.6±7.3	64.0±3.2	70.3±5.7	59.6±3.4	78.2±9.7	68.7±6.1	71.5±7.0	63.0±4.2	**85.6±3.9**
rec_auto_moto	59.0±5.8	57.3±4.0	59.8±3.6	63.0±2.6	64.6±3.7	57.6±4.8	63.1±2.6	63.0±2.5	59.7±4.0	**68.9±4.9**
rec_baseball_hockey	92.0±2.4	74.9±4.9	73.8±4.0	87.4±2.7	89.1±2.2	92.2±2.3	88.4±2.3	88.6±2.9	73.6±4.0	**93.8±2.0**
rec_moto_baseball	62.5±5.9	59.5±5.0	59.6±3.2	66.2±3.8	71.5±6.6	62.7±6.1	66.0±3.8	65.8±4.5	59.4±3.7	**74.6±6.8**
rec_moto_hockey	71.1±7.5	61.6±7.4	62.5±3.9	67.6±5.2	77.7±5.3	70.9±7.1	69.4±4.2	71.9±3.9	63.5±3.1	**83.0±3.3**
sci_crypt_elec	57.3±3.4	54.2±3.7	56.2±3.6	58.5±2.5	**60.1±2.2**	57.4±3.0	59.2±2.1	59.4±2.5	55.4±3.9	59.3±3.4
sci_crypt_med	58.1±4.0	55.7±3.7	58.4±4.4	59.8±3.8	58.1±2.3	57.9±3.4	60.1±4.3	63.0±4.2	58.0±4.2	**66.3±6.6**
sci_crypt_space	56.1±2.0	55.3±4.0	55.1±4.8	58.6±2.1	58.7±2.3	56.9±1.4	59.1±2.6	59.3±2.9	56.9±3.4	**61.5±3.7**
sci_elec_med	53.8±4.2	53.4±4.3	55.5±3.4	59.2±5.8	**61.4±5.8**	52.6±3.1	60.6±4.5	60.0±5.6	55.6±3.3	60.1±5.6
sci_elec_space	51.3±3.6	52.5±3.9	54.7±3.8	59.6±3.5	**60.8±3.5**	51.8±3.8	58.3±4.5	59.4±3.9	54.4±3.7	58.6±4.7
sci_med_space	53.7±2.3	54.5±3.4	53.2±3.3	57.2±3.4	56.8±3.4	53.7±2.7	57.6±3.9	56.9±3.3	54.0±2.6	**57.7±5.5**
talk_guns_mideast	63.5±8.8	59.4±4.0	64.3±4.0	73.5±3.4	76.6±2.8	64.2±7.5	73.5±3.4	74.1±3.1	64.3±4.0	**77.1±3.8**
talk_guns_misc	54.3±5.3	57.2±4.0	59.7±3.8	68.3±3.5	69.9±3.3	54.2±5.5	68.4±3.4	68.8±3.4	60.5±3.1	**70.1±3.9**
talk_mideast_misc	60.1±4.5	56.3±3.8	61.0±3.6	67.0±2.5	67.6±2.5	59.9±4.3	67.3±3.4	67.0±3.0	61.2±3.6	**69.5±5.3**
MIKI:W/T/L	25/0/0	25/0/0	25/0/0	21/4/0	18/6/1	25/0/0	21/4/0	20/5/0	25/0/0	-

Table 3: Testing accuracy (%, mean ± std.) on Moving Object Localization experiments. The highest average accuracy is bold.

	miSVM	MISVM	miGraph	MILES	MILDE	miSVM+	MILES+	MILES++	MICS	MIKI
Accuracy	70.5±11.8	60.5±6.7	71.1±5.4	75.4±4.7	75.7±4.4	67.7±10.2	73.7±5.2	74.8±3.6	72.8±5.0	**79.9±5.1**

than miSVM and MILES, respectively. This suggests that a simple extension by applying single-instance distribution shift approach does not work. One important reason is that in multi-instance learning there is no accurate information of instance label, and thus, instance weighting approach which is effective for single-instance setting may become misleading in multi-instance setting.

4.3 Experiments on non-Shift Data

We also run experiments on the popularly used benchmark datasets to validate the robustness of MIKI in no key instance shift setting. We conduct ten times 10-fold cross validations as previous studies done. As shown in Table 4, our method behaves comparable with state-of-the-art methods. The results indicate that although it is designed for key instance shift setting, our method still works fine in non-shift setting, demonstrating the robustness of MIKI. This is reasonable, since if there is no change of distribution, the estimated weights should be approximately equal to 1, thus the results of our method should be similar to other embedding based methods.

Table 4: Testing accuracy (%) on benchmark datasets. The compared results are from related literatures.

Dataset	miSVM	MISVM	miGraph	MILES	MILDE	MIKI
Musk1	87.4	77.9	**88.9**	84.2	87.1	88.2
Musk2	83.6	84.3	90.3	83.8	**91.0**	84.3
Elephant	82.0	81.4	86.8	**89.1**	85.0	86.5
Fox	58.2	59.4	61.1	**76.0**	66.5	66.5
Tiger	78.9	84.0	**86.0**	**86.0**	83.0	82.5

5 Application to Moving Object Localization

With the help of RFID or wireless LAN, moving object localization is widely applied in many areas such as locating cleaning robot, tracing livestock in the farm, tracking inmate in prison [Ferrer *et al.*, 2010], and it has been studied by using machine learning techniques [Senta *et al.*, 2007]. However, sometimes, we only care about if an entity has been to a target region, but not the exact location at each moment. Thus, investing too much efforts in gathering numerous sensing data and their exact location is not necessary. Instead, we only need to know whether the trajectory is intersected with the target region. However, even with the data collected in this way, there may still exist inconsistencies between training and test data, leading to key instance shift. For example, as shown in Figure 3, it's the top view of a room. Some RFID readers (the rectangles) are placed in the room to locate an moving entity with a RFID tag (the triangle) attached. A particular region (the circle) is marked as the target region. The entity is moving in the room and the sensing data of its location is collected by the readers. If it goes through the target region, then the trajectory is marked positive. However, if the environment changes between training and test phase, say, training data is collected from the entities with faster speed, whereas test data is collected from slower entities, then distribution discrepancy may arise between training and test phase, leading to the performance degradation of the obtained model.

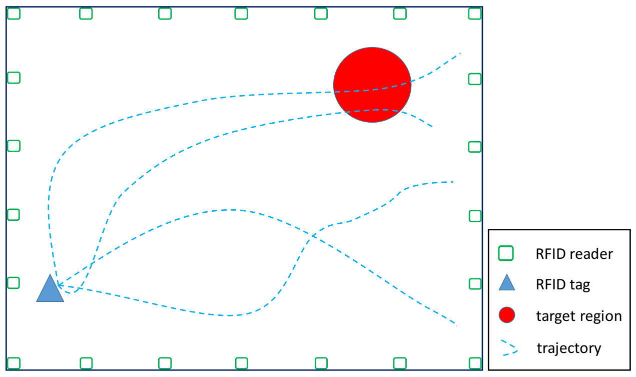


Figure 3: Diagram of moving object localization experiment

In this section, we formalize the aforementioned setting as an MIL problem. It's natural to regard each trajectory as a bag and the data from different location as instances, and the data collected from the target region is regarded as key instance. In addition, the inconsistencies of data collection result in the problem of key instance shift, which may seriously debase the performance of traditional MIL algorithms and make it necessary to consider the distribution change between training and test phase. In this paper, we apply MIKI to this problem and run experiments to verify its effectiveness.

We sample 80 bags from training data and test data each time, and we collect 30 sets of data in this way. As the results in Table 3 show, MIKI outperforms all other algorithms. The performances of the variants are even worse than miSVM and MILES, verifying again that direct adaptation of importance weighting technique can not work. Moreover, MICS behaves unfavorable, which inspires us that the distribution change of key instances should be addressed.

6 Conclusion

Previous studies of multi-instance learning (MIL) typically assume I.I.D. distribution, which may be violated in many applications. In this paper, we address the problem of MIL with key instance shift and propose an embedding based method MIKI to solve it. Experiments demonstrate the effectiveness of MIKI when key instance shift happens.

We mainly address the key instance shift problem of MIL in this work. We can also consider the scenario that distribution of negative instances varies. More sophisticatedly, distribution of both positive and negative instances may change in some tasks. The success of our proposed methods suggest that it is helpful to make good use of the instances in test bags.

References

[Amores, 2015] J. Amores. MILDE: multiple instance learning by discriminative embedding. *Knowledge and Information Systems*, 42(2):381–407, 2015.

[Andrews *et al.*, 2002] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568, 2002.

[Blei *et al.*, 2003] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Chen *et al.*, 2006] 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.

[Dietterich *et al.*, 1997] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

[Faria *et al.*, 2017] A. Faria, F. Coelho, A. Silva, H. Rocha, G. Almeida, A. Lemos, and A. Braga. MILKDE: A new approach for multiple instance learning based on positive instance selection and kernel density estimation. *Engineering Applications of Artificial Intelligence*, 59:196–204, 2017.

[Ferrer *et al.*, 2010] G. Ferrer, N. Dew, and U. Apte. When is RFID right for your service ? *International Journal of Production Economics*, 124(2):414–425, 2010.

[Gärtner *et al.*, 2002] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *Proceeding of the 19th International Conference on Machine Learning*, volume 2, pages 179–186, 2002.

[Herrera *et al.*, 2016] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez-Tarragó, and S. Vluymans. *Multiple Instance Learning: Foundations and Algorithms*. Springer, 2016.

[Huang *et al.*, 2006] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19*, pages 601–608, 2006.

[Kanamori *et al.*, 2009] T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10:1391–1445, 2009.

[Li *et al.*, 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 *Proceeding ot 20th European Conference on Machine Learning*, pages 17–32, 2009.

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

[Senta *et al.*, 2007] Y. Senta, Y. Kimuro, S. Takarabe, and T. Hasegawa. Machine learning approach to self-localization of mobile robots using RFID tag. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6, 2007.

[Shimodaira, 2000] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[Sugiyama *et al.*, 2008] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems 21*, pages 1433–1440, 2008.

[Sugiyama *et al.*, 2012] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

[Wei *et al.*, 2017] X.-S. Wei, J. Wu, and Z.-H. Zhou. Scalable algorithms for multi-instance learning. *IEEE Transactions on Neural Networks and Learning Systems*, 28(4):975–987, 2017.

[Yuan *et al.*, 2016] J. Yuan, X. Huang, H. Liu, B. Li, and W. Xiong. Submil: Discriminative subspaces for multi-instance learning. *Neurocomputing*, 173:1768–1774, 2016.

[Zadrozny, 2004] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceeding of the 21st International Conference on Machine Learning*, pages 114–121, 2004.

[Zhang and Zhou, 2014] W.-J. Zhang and Z.-H. Zhou. Multi-instance learning with distribution change. In *Proceeding of the 28th AAAI Conference on Artificial Intelligence*, pages 2184–2190, 2014.

[Zhang *et al.*, 2002] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts. Content-based image retrieval using multiple-instance learning. In *Proceeding of the 19th International Conference on Machine Learning*, pages 682–689, 2002.

[Zhou *et al.*, 2005] Z.-H. Zhou, K. Jiang, and M. Li. Multi-instance learning based web mining. *Applied Intelligence*, 22(2):135–147, 2005.

[Zhou *et al.*, 2009] Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li. Multi-instance learning by treating instances as non-iid samples. In *Proceeding of the 26th International Conference on Machine Learning*, pages 1249–1256, 2009.