Pattern Recognition 47 (2014) 2517–2531

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Multiple-instance discriminant analysis

Jing Chai [a,*], Xinghao Ding [b], Hongtao Chen [a], Tingyu Li [a]

[a] College of Information Engineering, Taiyuan University of Technology, Taiyuan 030024, PR China
[b] School of Information Science and Engineering, Xiamen University, Xiamen 361005, PR China

ARTICLE INFO

Article history:
Received 28 June 2013
Received in revised form
12 December 2013
Accepted 4 February 2014
Available online 13 February 2014

Keywords:
Multiple-instance learning
Feature extraction
Dimensionality reduction
Block coordinate ascent

ABSTRACT

Multiple-instance discriminant analysis (MIDA) is proposed to cope with the feature extraction problem in multiple-instance learning. Similar to MidLABS, MIDA is also derived from linear discriminant analysis (LDA), and both algorithms can be treated as multiple-instance extensions of LDA. Different from MidLABS which learns from the bag level, MIDA is designed from the instance level. MIDA consists of two versions, i.e., binary-class MIDA (B-MIDA) and multi-class MIDA (M-MIDA), which are utilized to cope with binary-class (standard) and multi-class multiple-instance learning tasks, respectively. The block coordinate ascent approach, by which we seek positive prototypes (the most positive instance in a positive bag is termed as the positive prototype of this bag) and projection vectors alternatively and iteratively, is proposed to optimize B-MIDA and M-MIDA to obtain lower dimensional transformation subspaces. Extensive experiments empirically demonstrate the effectiveness of B-MIDA and M-MIDA in extracting discriminative components and weakening class-label ambiguities for instances in positive bags.

1. Introduction

Different from traditional supervised learning where class labels are attached to instances and the goal is to predict the class labels of unseen instances, in multiple-instance learning only class labels of bags (a set of instances is termed as a bag) are known and the goal is to predict the class labels of unseen bags. A bag is classified as positive iff it contains at least one positive instance, and otherwise it is classified as negative. Fig. 1 depicts the comparison between supervised (mono-instance) learning and multiple-instance learning, where blue circles and red stars denote positive and negative instances, respectively. The collection of several instances with a rectangular contour represents a positive bag, while that with an ellipsoidal contour represents a negative bag. In subfigure (b), the number around each bag denotes the index of the bag, the prefixes "+" and "−" denote the positive and the negative classes, respectively. E.g., "+1" denotes this is the first positive bag, "−2" denotes this is the second negative bag. It is obvious that each object in supervised learning is an instance and that in multiple-instance learning is a collection of instances, i.e., a bag. Moreover, through Fig. 1, it is easy to see that whether containing at least one positive instance or not determines the class label of a bag.

The terminology "multiple-instance learning" was originally proposed by Dietterich et al. [1] when they were investigating the drug activity prediction problem. In their seminal paper, Dietterich et al. considered the problem of predicting whether a candidate drug molecule binds to the target protein or not. In particular, a molecule may take on many different shapes, and if any of these shapes conforms closely to the structure of the binding site, the candidate molecule binds to the target protein. By treating each shape of a molecule as an instance and each molecule as a bag, it is easy to see that drug activity prediction is a typical multiple-instance learning problem.

Besides drug activity prediction, multiple-instance learning appears in many other areas, such as image categorization [2–5], image retrieval [6–9], text classification [10,11], stock selection [10,12], protein sequence classification [8,13], computer aided diagnosis [14,15], and security application [16]. Zhou [17] gave a survey on the topic of multiple-instance learning and reviewed some important issues of this topic, such as the learnability, application domains, typical algorithms, and potential research scopes in the future.

In the past 15 years, multiple-instance learning has become very popular in the machine learning community, and researchers have proposed many representative algorithms to cope with various multiple-instance learning tasks. Maron and Ratan [2] studied the natural scene classification problem under the multiple-instance learning framework. They utilized diverse density (DD) to measure the closeness of a point to at least one instance in each positive bag and the remoteness of this point from all instances in negative bags, and then utilized the point with maximum DD as the "target concept" to operate classifications. Zhang and Goldman [18] combined DD and expectation maximization (EM) into a unified framework, and

* Corresponding author. Tel.: +86 15035101386; fax: +86 351 6010029.
E-mail address: jingchai@aliyun.com (J. Chai).

http://dx.doi.org/10.1016/j.patcog.2014.02.002
0031-3203

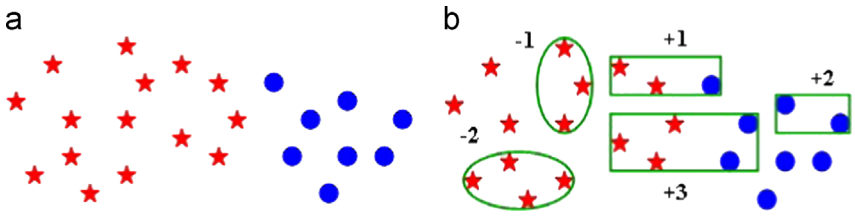

Fig. 1. Illustration of supervised learning and multiple-instance learning: (a) for supervised learning and (b) for multiple-instance learning. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

proposed the EM-DD algorithm to seek the point with maximum DD in an alternative and iterative way. Wang and Zucker [19] tried to utilize neighborhood information in multiple-instance learning and designed two *k*-nearest-neighbor (KNN) based multiple-instance classifiers: Bayesian-KNN and Citation-KNN. Andrews et al. [10] extended the support vector machines' (SVM) classifier to multiple-instance case and got MI-SVM and mi-SVM to cope with multiple-instance learning tasks. Gartner et al. [20] focused on the kernel design for multiple-instance data and proposed Multiple-Instance Kernel (MI-Kernel) to distinguish positive and negative bags. Zhou et al. [21] treated instances in each bag as independent samples and proposed two graph based algorithms (MI-Graph and mi-Graph) to mine the underlying structural information among within-bag instances. Zafra et al. [22] extended the traditional ReliefF algorithm [23] to multiple-instance case and proposed the ReliefF-MI algorithm to cope with multiple-instance feature selection tasks. Li et al. [24] studied the multiple-instance learning problem by assuming that instances are modeled as a mixture of concept and non-concept distributions, and thus classified a bag as positive if the fraction of concept instances in it was larger than a particular threshold. Zhang et al. [25] treated automatically grouping motion patterns in traffic scenes as a multiple-instance learning problem, and then proposed the Maximum Margin Multi-instance Multi-cluster Learning (M^4L) algorithm to cope with this problem.

Standard multiple-instance learning consists of two classes, i.e., a positive class and a negative class. However, with the rapid development of multiple-instance learning, its application domain has been extended from the binary-class case to the multi-class case [4,7,10]. In multi-class multiple-instance learning, for each given class, if any instance in a bag represents the class label of the given class, we say this instance is positive for the class, and hence, this bag is positive for the class as well; otherwise, this bag is negative for the class. Note that in multi-class multiple-instance learning, usually we do not define the specific negative bags for each class, because positive bags for some class can be simultaneously treated as negative bags for other classes, e.g., positive bags for class *c* are also negative bags for classes except *c*.

Similar to other machine learning branches such as supervised, unsupervised and semi-supervised learning, the feature extraction problem exists in multiple-instance learning as well, e.g., the multiple-instance data may also contain noisy and redundant components, the curse-of-dimensionality problem may also occur in high dimensional applications. Through feature extraction, we may reduce data's dimensionality and save memory space, remove useless and noisy components, reduce the time complexity in testing phase, weaken the disadvantage caused by the curse-of-dimensionality problem, and improve classification accuracies. In the past few years, several researchers have studied the multiple-instance feature extraction problem and proposed several dimensionality reduction algorithms. E.g., Sun et al. [26] designed a probabilistic multiple-instance dimensionality reduction algorithm, namely Multi-Instance Dimensionality Reduction (MIDR), and proposed to solve it by gradient descent along the tangent space of the orthonormal projection matrix; Ping et al. [27] utilized the structural information conveyed by instances in a bag to learn lower dimensional representations of original data, and designed an algorithm named as Multi-Instance Dimensionality reduction by Learning a mAximum Bag margin Subspace (MidLABS); Kim and Choi [28] proposed the Citation Local Fisher Discriminant Analysis (CLFDA) algorithm to utilize the citation and reference information in detecting false positive instances and extracting local discriminative information for multiple-instance learning.

Linear Discriminant Analysis (LDA) [29], which utilized the class-label information to maximize the ratio of between-class scattering to within-class scattering, was a classical supervised feature extraction algorithm and had been successfully applied in many supervised learning tasks [30–35]. In this paper, we propose Multiple-Instance Discriminant Analysis (MIDA), an extension of LDA, to cope with the multiple-instance feature extraction and dimensionality reduction problems. Since there are two kinds of multiple-instance learning problems, i.e., the binary-class one and the multi-class one, the proposed MIDA algorithm has two versions as well, which can be abbreviated as B-MIDA (Binary-class MIDA) and M-MIDA (Multi-class MIDA), respectively. Note that the above mentioned MidLABS algorithm can be treated as multiple-instance extension of LDA as well, and hence our MIDA work is very similar to MidLABS. Both MIDA and MidLABS try to maximize the trace of the between-class scattering matrix and minimizes that of the within-class one simultaneously, but their design principles are very different, because they construct the scattering matrices from different levels. MIDA constructs the scattering matrices from the instance level, i.e., it selects a prototype for each bag and utilizes this prototype as the representative of the bag to construct scattering matrices. In contrast, MidLABS constructs the scattering matrices from the bag level, i.e., it directly evaluates the similarity and scattering among bags. Moreover, since MIDA is derived from LDA, some limitations of LDA such as the unavailability for multi-modal data and the independently and identically distributed (i.i.d.) assumption for instances in the same class also exist in MIDA. Hence, although the experimental results shown in Section 5 demonstrate that MIDA performs well in extracting discriminative components, it may still be improved in the future.

Note that the main difference of multiple-instance learning from supervised learning is that there are class-label ambiguities for instances derived from positive bags. If we can find out the most positive instance in each positive bag, the disadvantage caused by the class-label ambiguities may be weakened and we may utilize supervised techniques to design multiple-instance feature extraction algorithms. Therefore, both B-MIDA and M-MIDA contain two types of unknown variables, of which the first type are positive prototypes (the most positive instance in each positive bag is termed as the positive prototype of this bag), the second type are projection vectors. It is difficult to optimize the two types of unknown variables simultaneously, because they are neither jointly convex w.r.t. (with respect to) the objective function nor can be optimized with analytical solutions. Instead, we utilize the block coordinate ascent approach [36] to update the above two types of unknown variables alternatively and iteratively. In each iteration, first we fix one type of unknown variables and update the other type of ones, then alternate the order of the above two types of unknown variables and update the fixed type of ones in last step. We repeat the above two steps iteratively, until

the relative change of the objective function in two neighboring iterations is less than a predefined threshold.

The rest of this paper is organized as follows. In Section 2, we give a brief review of several existing multiple-instance feature extraction algorithms and discuss their relationships to our work. In Section 3, we introduce B-MIDA and discuss how to optimize B-MIDA. In Section 4, we extend B-MIDA to the multi-class case and get M-MIDA, and then give the optimization of M-MIDA. In Section 5, we compare B-MIDA and M-MIDA with some competing algorithms via empirical experiments conducted on the synthetic and real-world datasets. Finally, we give concluding remarks and discuss the future work in Section 6.

2. Related algorithms

In this section, we give a brief review of three multiple-instance dimensionality reduction algorithms: MIDR [26], MidLABS [27], and CLFDA [28], discuss their relationships to our algorithms, and analyze their time complexities. Note that B-MIDA and M-MIDA have the same design principles, the slight difference between them is that B-MIDA is for binary-class learning whereas M-MIDA is for multi-class learning. In the following discussions, for simplicity, we utilize MIDA to represent both of them if this does not cause ambiguities.

2.1. MIDR

MIDR aims at making the posterior probability of a bag being positive close to one if the bag is truly positive and zero otherwise. The objectives of MIDR and MIDA are highly different from each other. MIDR minimizes the sum of squared losses between the above posteriors and the binary bag labels, whereas MIDA maximizes the difference between the between-class scatterings and the within-class ones. One point in common is that both MIDR and MIDA contain a process of seeking positive prototypes, despite that MIDA performs the seeking explicitly, while MIDR performs the seeking implicitly (involved in the calculation of the above posterior probabilities).

Next we analyze the time complexity of MIDR. Suppose the transformation matrix A to be calculated in MIDR is of size $D \times d$. MIDR uses gradient descent to update A, and the update consists of two kinds of iterations: the outer iteration and the inner one. In the outer iteration, the main work is to calculate the gradient of the objective function w.r.t. A, during which we need to calculate the gradient of the posterior each instance being positive w.r.t. A, summarize these gradients to get the total gradient, and project the total gradient onto the tangent space. The time complexity of each outer iteration (without considering the inner iteration contained in it) is $O(n_{\text{sum}}Dd^2)+O(D^2d^4)$, where n_{sum} denote the number of all instances in all bags, $O(n_{\text{sum}}Dd^2)$ is the time complexity of calculating the gradients, $O(D^2d^4)$ is the time complexity of projecting the total gradient onto the tangent space. In each outer iteration, there is also an inner iteration which is adopted to tune the step size of the gradient update, and the time complexity of each inner iteration is approximately the same to that of each outer one (without considering the inner iteration). Let t_{in} and t_{out} respectively denote the average number of inner iterations and the number of outer iterations, then the overall time complexity of solving MIDR is $O(t_{\text{in}}t_{\text{out}}n_{\text{sum}}Dd^2 + t_{\text{in}}t_{\text{out}}D^2d^4)$.

2.2. MidLABS

Both MIDA and MidLABS simultaneously maximize the between-class scatterings and minimize the within-class ones, hence both of them can be treated as multiple-instance extensions of LDA. One obvious difference between them is that MIDA utilizes the trace-difference formulation while MidLABS utilizes the trace-ratio one. However, since Guo et al. [31] have shown that the trace-difference formulation is very close to the corresponding trace-ratio one (one important conclusion of [31] shows that the transformation matrix of the trace-difference problem is the same to that of the corresponding trace-ratio problem, as long as the trade-off parameter (in our case, α) of the trace-difference problem equals the optimal ratio of the corresponding trace-ratio problem; the detailed proof of this conclusion can be found in Theorem 2 of [31]), the difference in formulations is not the major one between MIDA and MidLABS. One major difference is that they construct scattering matrices from different levels. MIDA constructs scattering matrices from the instance level, i.e., it selects a prototype for each bag and utilizes the prototype as the representative of this bag to construct scattering matrices. In contrast, MidLABS constructs scattering matrices from the bag level by directly evaluating the scatterings among bags. The other major difference is that MidLABS takes the structural information of data into account, whereas MIDA does not. MidLABS treats instances in each bag as non-i.i.d. ones, i.e., it considers the relationship among instances in each bag by measuring their distances and building an edge between two instances if their distance is smaller than a threshold, and then utilizes edges to describe the structural information among within-bag instances. In contrast, MIDA treats instances in the same class as i.i.d. ones, i.e., it utilizes the selected positive instances (positive prototypes) and mean vectors of negative instances (negative prototypes) to construct scattering matrices. In short, similar to LDA, MIDA does not consider the structural information of data as well, because it pays no attention to the relationship among within-bag instances.

The design of MidLABS consists of two steps: constructing scattering matrices and operating eigenvalue decomposition. There are two kinds of scattering matrices in MidLABS, i.e., the node matrices and the edge matrices. Let l denote the number of all bags, n_{ave} denote the average number of instances in each bag, then the time complexity of constructing the node matrices is $O(l^2n_{\text{ave}}^2D^2)$. Before the construction of the edge matrices, the Euclidean distance between each pair of within-bag instances should be calculated, and the time complexity of calculating these Euclidean distances is $O(ln_{\text{ave}}^2D)$. After that, we may construct the edge matrices, and the time complexity of this process is $O(l^2n_{\text{ave}}^4D^2)$, which is approximately n_{ave}^2 times of that of constructing the node matrices, due to that the number of edges in a bag is usually the square of the number of nodes in this bag. Therefore, the overall time complexity of constructing scattering matrices is $O(l^2n_{\text{ave}}^2D^2 + ln_{\text{ave}}^2D + l^2n_{\text{ave}}^4D^2)$, which can be approximate as $O(l^2n_{\text{ave}}^4D^2)$. The time complexity of operating eigenvalue decomposition is $O(D^3)$. Hence, the overall time complexity of solving MidLABS can be approximated as $O(l^2n_{\text{ave}}^4D^2+D^3)$.

2.3. CLFDA

CLFDA [28] performs multiple-instance dimensionality reduction by incorporating the citation and reference information [19] into local Fisher discriminant analysis [32], thus it can be treated as the multiple-instance extension of LDA as well. The central idea of CLFDA and that of MIDA are kind of complementary to each other, because MIDA tries to seek correctly labeled instances in positive bags (i.e., positive prototypes), whereas CLFDA tries to detect incorrectly labeled instances in positive bags (i.e., false positive instances). In order to detect false positive instances, CLFDA first pre-labels all instances with their bag labels, and then adopts the neighborhood information among them to detect the false positive ones. However, the rationality of the pre-labeling process is questionable, because simply treating all instances in positive bags as positive instances is usually not a reasonable

choice, especially for cases which involve a large number of negative instances in positive bags.

The design of CLFDA consists of three steps: detecting false positive instances, constructing scattering matrices and operating eigenvalue decomposition. Before the detection of false positive instances, the Euclidean distance between each pair of instances, based on which we may generate the max(R,C)-*NN* graph [28], should be calculated in advance. The time complexity of calculating these Euclidean distances is $O(n_{\text{sum}}^2 D)$, where n_{sum} denotes the total number of instances in all bags. The time complexity of constructing scattering matrices is $O(n_{\text{sum}}^2 D^2)$, and that of operating eigenvalue decomposition is $O(D^3)$. Therefore, the overall time complexity of solving CLFDA is $O(n_{\text{sum}}^2 D + n_{\text{sum}}^2 D^2 + D^3)$, which can be approximated as $O(n_{\text{sum}}^2 D^2 + D^3)$.

3. B-MIDA

In this section, we discuss binary-class multiple-instance feature extractions and demonstrate how to extend LDA to get B-MIDA. For convenience of representation, first we introduce some useful notations. Suppose there are $l = l^+ + l^-$ bags in total, of which l^+ bags are positive and l^- bags are negative. B_i^+ ($i \in \{1, \ldots, l^+\}$) and i denote the ith positive bag and the positive bag index, respectively. Let $x_{i,j}^+ \in R^D$ denote the jth instance in B_i^+, then B_i^+ can be expressed as $[x_{i,1}^+, \ldots x_{i,n_i^+}^+]$, where n_i^+ denotes the number of instances in B_i^+. Similarly, B_i^- ($i \in \{1, \ldots, l^-\}$), i, $x_{i,j}^- \in R^D$, and n_i^-, denote the ith negative bag, the negative bag index, the jth instance in B_i^-, and the number of instances in B_i^-, respectively.

3.1. Scattering matrices

Now we discuss how to construct scattering matrices for binary-class multiple-instance feature extraction. As mentioned above, the main difficulty of multiple-instance learning lies in that there are class-label ambiguities for instances derived from positive bags. Therefore, if we may correctly find out the positive prototype for each class, we can transform multiple-instance data to single-instance data by utilizing these prototypes as representatives of bags to construct scattering matrices. Suppose all positive prototypes are known in advance (the way of seeking these prototypes is discussed in following optimization subsection), we can construct the positive within-class scattering matrix as

$$S_{B-\text{MIDA}}^{w\,+} = \sum_{i=1}^{l^+} (pt_i^+ - pt_{\text{mean}}^+)(pt_i^+ - pt_{\text{mean}}^+)^{\text{T}} \tag{1}$$

where pt_i^+ denotes the positive prototype for the ith positive bag B_i^+, pt_{mean}^+ denotes the mean positive prototype

$$pt_{\text{mean}}^+ = \frac{1}{l^+} \sum_{i=1}^{l^+} pt_i^+ \tag{2}$$

The positive within-class scattering matrix in (1) is the sum of the scatterings between each positive prototype and the mean positive prototype. It is easy to verify that the upper bound of the rank of $S_{B-\text{MIDA}}^{w\,+}$ is $\min(D, l^+ - 1)$.

By treating the mean of all instances in each negative bag as the corresponding negative prototype, we may construct the negative within-class scattering matrix as

$$S_{B-\text{MIDA}}^{w\,-} = \sum_{i=1}^{l^-} (pt_i^- - pt_{\text{mean}}^-)(pt_i^- - pt_{\text{mean}}^-)^{\text{T}} \tag{3}$$

where

$$pt_i^- = \frac{1}{n_i^-} \sum_{j=1}^{n_i^-} x_{i,j}^- \tag{4}$$

denotes the ith negative prototype, and

$$pt_{\text{mean}}^- = \sum_{i=1}^{l^-} n_i^- pt_i^- \Big/ \left(\sum_{i=1}^{l^-} n_i^- \right) \tag{5}$$

is the mean of all instances in all negative bags and utilized to denote the mean negative prototype. The negative within-class scattering matrix in (3) is the sum of the scatterings between each negative prototype and the mean negative prototype. The upper bound of the rank of $S_{B-\text{MIDA}}^{w\,-}$ is $\min(D, l^-)$.

The total within-class scattering matrix is the sum of the positive and negative within-class scattering matrices

$$S_{B-\text{MIDA}}^{w} = S_{B-\text{MIDA}}^{w\,+} + S_{B-\text{MIDA}}^{w\,-} \tag{6}$$

By utilizing both positive and negative prototypes, the between-class scattering matrix can be expressed as

$$S_{B-\text{MIDA}}^{b} = \sum_{p=1}^{l^+} \sum_{q=1}^{l^-} (pt_p^+ - pt_q^-)(pt_p^+ - pt_q^-)^{\text{T}} \tag{7}$$

The between-class scattering matrix in (7) is the sum of the scatterings between each positive prototype and each negative prototype. The upper bound of the rank of $S_{B-\text{MIDA}}^{b}$ is $\min(D, l^+ \times l^-)$.

3.2. Optimization process

On the basis of within-class and between-class scattering matrices, we may formulate B-MIDA in a similar way to that of LDA. However, here we make a slight difference to LDA by replacing the trace-ratio formulation with the trace-difference one [30] (compared with trace-ratio, trace-difference makes the following updating of positive prototypes more easy to handle) and express B-MIDA as

$$\arg\max_{G} \ \text{trace}[G^{\text{T}}(S_{B-\text{MIDA}}^{b} - \alpha S_{B-\text{MIDA}}^{w})G] \ \text{ s.t.} G^{\text{T}}G = I_d \tag{8}$$

where $G \in R^{D \times d}$ is the transformation matrix with d denoting the dimensionality of the transformed subspace, $I_d \in R^{d \times d}$ is the identity matrix, α is the trade-off parameter utilized to control the relative importance of the between-class scattering matrix to the within-class one.

If all variables in $S_{B-\text{MIDA}}^{b}$ and $S_{B-\text{MIDA}}^{w}$ are known in advance, we may solve (8) easily by performing eigenvalue decomposition and formulate the transformation matrix G as the combination of top d eigenvectors. Unfortunately, the positive prototypes in $S_{B-\text{MIDA}}^{b}$ and $S_{B-\text{MIDA}}^{w}$ are unknown variables. Hence, there are two types of unknown variables in B-MIDA in total, of which the first type are the positive prototypes, the second type are the transformation vectors. Moreover, the two types of unknown variables are coupled with each other and not jointly convex w.r.t. the objective function of (8), thus it is difficult to get analytical solutions or optimize the two types of unknown variables simultaneously. We adopt the block coordinate ascent approach to cope with the above coupled optimization problem: first we fix the transformation matrix and seek the positive prototypes, then fix the positive prototypes and update the transformation matrix; repeat the above two steps iteratively until the relative change of the objective function $\text{trace}[G^{\text{T}}(S_{B-\text{MIDA}}^{b} - \alpha S_{B-\text{MIDA}}^{w})G]$ in two neighboring iterative rounds is less than a predefined threshold. Let $obj(r)$ and $obj(r+1)$ denote the objective function in round r and round $r+1$, respectively, then the relative change of the objective

function in round r and round $r+1$ can be expressed as $|(obj(r+1)-obj(r))/obj(r)|$, where $||$ denotes the absolute value. If the above relative change is less than the predefined threshold ξ, the iterative process could be terminated after round $r+1$. Some researchers [36–38] found that when ξ was small enough, optimization results were insensitive to the specific values of ξ. In our experiments, empirical results show that when ξ is less than 10^{-6}, there is almost no change in optimization results, thus for simplicity we set ξ as 10^{-6} in all following experiments.

Next we discuss the iterative optimization process in detail. We start our discussion with the first step, i.e., seeking positive prototypes. Note that (8) can be reformulated as

$$\begin{aligned}
&\underset{G}{\operatorname{argmax}}\ \operatorname{trace}[G^{\mathrm{T}}(S^{b}_{B-\mathrm{MIDA}}-\alpha S^{w}_{B-\mathrm{MIDA}})G] \\
&= \sum_{p=1}^{l^+}\left\{\sum_{q=1}^{l^-}\|G^{\mathrm{T}}(pt_p^+-pt_q^-)\|_2^2-\alpha\|G^{\mathrm{T}}(pt_p^+-pt^+_{\mathrm{mean}})\|_2^2\right\} \\
&\quad -\alpha\sum_{q=1}^{l^-}\|G^{\mathrm{T}}(pt_q^--pt^-_{\mathrm{mean}})\|_2^2 \\
&= \sum_{p=1}^{l^+}T_p-\alpha\sum_{q=1}^{l^-}\|G^{\mathrm{T}}(pt_q^--pt^-_{\mathrm{mean}})\|_2^2 \\
&\text{s.t.}\quad G^{\mathrm{T}}G=I_d
\end{aligned} \tag{9}$$

where $\|\|_2^2$ denotes the squared l_2-norm and T_p denotes $\sum_{q=1}^{l^-}\|G^{\mathrm{T}}(pt_p^+-pt_q^-)\|_2^2-\alpha\|G^{\mathrm{T}}(pt_p^+-pt^+_{\mathrm{mean}})\|_2^2$.

Note that positive prototypes are only contained in the first objective term of (9), i.e., the $\sum_{p=1}^{l^+}T_p$ term. Therefore, the second objective term can be omitted during the seeking of positive prototypes. Since pt^+_{mean} is the mean of all positive prototypes, the seeking of pt_p^+ is coupled with the seeking of other positive prototypes. We adopt a heuristic approach similar to that utilized in k-means clustering [29] to update pt_p^+ and pt^+_{mean}: first assume pt^+_{mean} is known and fix it, try all instances in B_p^+ and find the instance that maximizes T_p, reset this instance as pt_p^+, i.e., the positive prototype for the pth positive bag; then seek other positive prototypes in the same way, and after all positive prototypes have been sought, recalculate the new pt^+_{mean} according to (2).

Now we revisit (8) and explain why we utilize the trace-difference formulation instead of the trace-ratio one to design B-MIDA. The reason is that the trace-difference formulation simplifies the seeking of positive prototypes. If we utilize the trace-ratio formulation to design B-MIDA, then it is easy to see that each positive prototype will appear both in the summarized numerator terms and the summarized denominator terms, thus the seeking of positive prototypes will become very difficult (due to the summarized formulation of both numerator and denominator). In contrast, adopting the trace-difference formulation will make the above seeking much easier to cope with.

When all positive prototypes have been found, the transformation matrix can be calculated easily by performing eigenvalue decomposition. Suppose that $\lambda_1>\cdots>\lambda_d$ are the top d largest eigenvalues of $S^{b}_{B-\mathrm{MIDA}}-\alpha S^{w}_{B-\mathrm{MIDA}}$ and $u_1,\ldots,u_d$ are the corresponding eigenvectors, then $G_{B-\mathrm{MIDA}}$ can be expressed as

$$G_{B-\mathrm{MIDA}}=[u_1,\ldots,u_d] \tag{10}$$

Before the iterative optimization process starts, we need to initialize the positive prototypes and transformation matrix. We utilize kernel density estimation [39,40] to initialize positive prototypes. Given any positive bag, we adopt all instances in negative bags as training samples to estimate the negative densities for all instances in this positive bag, and then select the instance with the lowest negative density as the corresponding positive prototype.

In particular, we utilize the Gaussian kernel in our estimation, and the negative density of a given instance x can be expressed as

$$p(x)=\frac{1}{V}\sum_{i=1}^{l^-}\sum_{j=1}^{n_i^-}\exp\left(-\frac{\|x-x_{i,j}^-\|_2^2}{\sigma}\right) \tag{11}$$

where σ is the bandwidth of the Gaussian kernel, V is the normalizing factor which makes $p(x)$ an appropriate probability density function (PDF). The bandwidth σ is a free parameter which exhibits a strong influence on the resulting estimation, and we utilize the following approach to select the appropriate value of σ in our initialization. First we set the candidate set of σ as {0.001, 0.01, 0.1, 1, 10, 100, 1000}, then utilize each candidate σ to select positive prototypes and get the mean positive prototype pt^+_{mean} according to (2), calculate the Euclidean distance between pt^+_{mean} and pt^-_{mean}, i.e., $||pt^+_{\mathrm{mean}}-pt^-_{\mathrm{mean}}||_2$, and finally set σ as the value maximizing the above distance and keep the prototypes w.r.t. this value as the initialized ones. After all positive prototypes have been found, we may initialize the transformation matrix according to (10), and then the iterative optimization process starts.

The pseudo code of the initialization of B-MIDA is shown in Algorithm 1, and that of the entire optimization of B-MIDA is shown in Algorithm 2.

Algorithm 1. Initialization of B-MIDA.

Input: B_p^+ $(p\in\{1,\ldots,l^+\})$, B_q^- $(q\in\{1,\ldots,l^-\})$, the candidate set of σ {0.001, 0.01, 0.1, 1, 10, 100, 1000}
Output: Initialized positive prototypes pt_p^+ $(p\in\{1,\ldots,l^+\})$ and transformation matrix $G_{B-\mathrm{MIDA}}$.
Initialization:
Use each σ in the candidate set to select positive prototypes:
For each $p\in\{1,\ldots,l^+\}$, estimate the negative density of each instance in B_p^+ according to (11), and select the instance with the lowest negative density as pt_p^+.
Calculate pt^+_{mean} according to (2), and get the Euclidean distance $\|pt^+_{\mathrm{mean}}-pt^-_{\mathrm{mean}}\|_2$.
Find the σ corresponding to the maximal value of $\|pt^+_{\mathrm{mean}}-pt^-_{\mathrm{mean}}\|_2$, and set the positive prototypes corresponding to this σ as the initialized ones.
Initialize $G_{B-\mathrm{MIDA}}$ according to (10).
End

Algorithm 2. Entire optimization process of B-MIDA.

Input: B_p^+ $(p\in\{1,\ldots,l^+\})$, B_q^- $(q\in\{1,\ldots,l^-\})$, α, d, convergence tolerance ξ, maximum iteration number t, maximum iteration number for the first step t_1.
Output: Projection matrix $G_{B-\mathrm{MIDA}}$.
Initialize the positive prototypes pt_p^+ $(p\in\{1,\ldots,l^+\})$ and the transformation matrix $G_{B-\mathrm{MIDA}}$ according to Algorithm 1.
Iterative optimization:
Repeat:
First step: Fix $G_{B-\mathrm{MIDA}}$, repeat the following iterations until there is no change in pt_p^+ $(p\in\{1,\ldots,l^+\})$ or the iteration number reaches t_1:
For each $p\in\{1,\ldots,l^+\}$, try all instances in B_p^+, find the one maximizing T_p in (9) and select it as pt_p^+. After all pt_p^+ $(p\in\{1,\ldots,l^+\})$ have been selected, recalculate pt^+_{mean} according to (2).

Second step: Fix pt_p^+ $(p \in \{1,\ldots,l^+\})$ and pt_{mean}^+, operate eigenvalue decomposition on $S_{B-\text{MIDA}}^b - \alpha S_{B-\text{MIDA}}^w$, find the eigenvectors corresponding to top d largest eigenvalues and formulate $G_{B-\text{MIDA}}$ according to (10).

Until the relative change of the objective function of (9) is less than ξ or the iteration number reaches t.

3.3. Time complexity

Now we analyze the time complexity of B-MIDA. In the initialization step, we need to calculate the Euclidean distance between each instance from any positive bag and each instance from any negative bag. Let n_{sum}^+ and n_{sum}^- denote the number of instances in all positive bags and that in all negative bags, respectively. Then the time complexity of the initialization step is $O(n_{\text{sum}}^+ n_{\text{sum}}^- D)$. In the optimization of B-MIDA, there are two kinds of iterations, i.e., the outer iteration and the inner one. In each inner iteration, we need to try all instances in positive bags and select some as the positive prototypes. For each candidate instance, T_p in (9) suggests that the required time complexity is $O(l^- Dd^2)$, and hence the overall time complexity for each inner iteration is $O(n_{\text{sum}}^+ l^- Dd^2)$. In each outer iteration, besides the seeking of positive prototypes, we still need to operate eigenvalue decomposition once, and the corresponding complexity is $O(D^3)$. Let the average number of inner iterations and the number of outer iterations respectively be t_{in} and t_{out}, then the overall time complexity of solving B-MIDA is $O(n_{\text{sum}}^+ n_{\text{sum}}^- D + t_{\text{in}} t_{\text{out}} n_{\text{sum}}^+ l^- Dd^2 + t_{\text{out}} D^3)$.

4. M-MIDA

In this section, we discuss how to extend B-MIDA to the multi-class case and get M-MIDA. Suppose there are C classes in total, where $c \in \{1,\ldots,C\}$ denotes the corresponding class label. For class c, suppose there are l_c^+ positive bags, of which $B_{c,i}^+$ $(i \in \{1,\ldots,l_c^+\})$ and i denote the ith positive bag and the positive bag index for class c, respectively. Let $x_{c,i,j}^+ \in R^D$ denote the jth instance in $B_{c,i}^+$, then $B_{c,i}^+$ can be expressed as $[x_{c,i,1}^+,\ldots,x_{c,i,n_{c,i}^+}^+]$, where $n_{c,i}^+$ denotes the number of instances in $B_{c,i}^+$. We do not define the specific negative bags for each class, because in multi-class multiple-instance learning positive bags for a given class can be simultaneously treated as negative bags for other classes, e.g., $B_{c,i}^+$ is a negative bag for classes except c.

4.1. Scattering matrices

For multi-class multiple-instance learning, the within-class scattering matrix can be constructed as

$$S_{M-\text{MIDA}}^w = \sum_{c=1}^{C} \sum_{i=1}^{l_c^+} (pt_{c,i}^+ - pt_{c,\text{mean}}^+)(pt_{c,i}^+ - pt_{c,\text{mean}}^+)^{\text{T}} \tag{12}$$

where $pt_{c,i}^+$ denotes the positive prototype for the ith positive bag of class c, i.e., $B_{c,i}^+$, and $pt_{c,\text{mean}}^+$ denotes the mean positive prototype for class c

$$pt_{c,\text{mean}}^+ = \frac{1}{l_c^+} \sum_{i=1}^{l_c^+} pt_{c,i}^+ \tag{13}$$

The within-class scattering matrix in (12) is the sum of the scatterings between each positive prototype for some class and the mean positive prototype for this class. The upper bound of the rank of $S_{M-\text{MIDA}}^w$ is $\min(D, \sum_{c=1}^{C} l_c^+ - C)$.

The between-class scattering matrix can be constructed as

$$S_{M-\text{MIDA}}^b = \sum_{c=1}^{C} \sum_{i=1}^{l_c^+} \sum_{e \neq c, e=1}^{C} (pt_{c,i}^+ - pt_{e,\text{mean}}^+)(pt_{c,i}^+ - pt_{e,\text{mean}}^+)^{\text{T}} \tag{14}$$

The between-class scattering matrix in (14) is the sum of the scatterings between each positive prototype for some class and the mean positive prototypes for other classes. The upper bound of the rank of $S_{M-\text{MIDA}}^b$ is $\min(D, (C-1) \times \sum_{c=1}^{C} l_c^+)$.

4.2. Optimization process

M-MIDA can also be formulated as

$$\arg\max_G \text{trace}[G^{\text{T}}(S_{M-\text{MIDA}}^b - \alpha S_{M-\text{MIDA}}^w)G] \text{ s.t.} G^{\text{T}}G = I_d \tag{15}$$

where $G \in R^{D \times d}$ is the transformation matrix, $I_d \in R^{d \times d}$ is the identity matrix, α is the trade-off parameter.

By substituting (12) and (14) into (15), we may rewrite M-MIDA as

$$\begin{aligned} &\arg\max_G \text{trace}[G^{\text{T}}(S_{M-\text{MIDA}}^b - \alpha S_{M-\text{MIDA}}^w)G] \\ &= \sum_{c=1}^{C} \sum_{i=1}^{l_c^+} \left\{ \sum_{e \neq c, e=1}^{C} \|G^{\text{T}}(pt_{c,i}^+ - pt_{e,\text{mean}}^+)\|_2^2 - \alpha \|G^{\text{T}}(pt_{c,i}^+ - pt_{c,\text{mean}}^+)\|_2^2 \right\} \\ &= \sum_{c=1}^{C} \sum_{i=1}^{l_c^+} T_{c,i} \\ &\text{s.t.} \quad G^{\text{T}}G = I_d \end{aligned} \tag{16}$$

where $T_{c,i}$ denotes $\sum_{e \neq c, e=1}^{C} \|G^{\text{T}}(pt_{c,i}^+ - pt_{e,\text{mean}}^+)\|_2^2 - \alpha \|G^{\text{T}}(pt_{c,i}^+ - pt_{c,\text{mean}}^+)\|_2^2$.

If all positive prototypes are known, $G_{M-\text{MIDA}}$ can be expressed as the combination of eigenvectors corresponding to the top d largest eigenvalues of $S_{M-\text{MIDA}}^b - \alpha S_{M-\text{MIDA}}^w$

$$G_{M-\text{MIDA}} = [u_1,\ldots,u_d] \tag{17}$$

Similar to B-MIDA, we do not know positive prototypes in advance but have to adopt the heuristic approach to seek them: for each class, without loss of generality let it be class c, first assume $pt_{c,\text{mean}}^+$ is known and fix it, try all instances in $B_{c,i}^+$ and find the one maximizing $T_{c,i}$, reset this instance as $pt_{c,i}^+$, i.e., the positive prototype for the ith positive bag of class c; then seek the positive prototypes for other bags of class c in the same way, and after all positive prototypes of class c have been sought, recalculate $pt_{c,\text{mean}}^+$ according to (13).

Similar to B-MIDA, we adopt the block coordinate ascent approach to alternatively update the two kinds of unknown variables (positive prototypes and transformation matrix) of M-MIDA in an iterative way, and set the threshold ξ which is utilized to terminate the optimization as 10^{-6} in all following experiments.

First we should initialize the positive prototypes and transformation matrix. The positive prototypes for each class are initialized by Gaussian kernel density estimation, which is operated by utilizing all instances in negative bags for this class as training samples and selecting the instance with the lowest negative density in a given positive bag for this class as the corresponding positive prototype. E.g., the negative density of a given instance x for class c can be expressed as

$$p(x) = \frac{1}{V} \sum_{e \neq c, e=1}^{C} \sum_{i=1}^{l_e^+} \sum_{j=1}^{n_{e,i}^+} \exp\left(-\frac{\|x - x_{e,i,j}^+\|_2^2}{\sigma} \right) \tag{18}$$

where V is the normalizing factor, σ is the kernel width and of which the candidate set is the same to that of B-MIDA. The transformation matrix is initialized according to (17). When

the initialization is finished, the iterative optimization process starts.

The pseudo code of the initialization of M-MIDA is shown in Algorithm 3, and that of the entire optimization of M-MIDA is shown in Algorithm 4.

Algorithm 3. Initialization of M-MIDA.

Input: $B_{c,i}^{+}$ ($c \in \{1,\ldots,C\}$, $i \in \{1,\ldots,l_c^{+}\}$), the candidate set of σ $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.
Output: Initialized positive prototypes $pt_{c,i}^{+}$ ($c \in \{1,\ldots,C\}$, $i \in \{1,\ldots,l_c^{+}\}$) and transformation matrix $G_{M-\mathrm{MIDA}}$.
Initialization:
Use each σ in the candidate set to select positive prototypes:
For each $c \in \{1,\ldots,C\}$ and each $i \in \{1,\ldots,l_c^{+}\}$, estimate the negative density of each instance in $B_{c,i}^{+}$ according to (18), and select the instance with the lowest negative density as $pt_{c,i}^{+}$.
For each $c \in \{1,\ldots,C\}$, calculate $pt_{c,\mathrm{mean}}^{+}$ according to (13); get the summarized Euclidean distance $\sum_{c=1}^{C}\sum_{e\neq c,e=1}^{C}\|pt_{c,\mathrm{mean}}^{+}-pt_{e,\mathrm{mean}}^{+}\|_2$.
Find the σ corresponding to the maximal value of $\sum_{c=1}^{C}\sum_{e\neq c,e=1}^{C}\|pt_{c,\mathrm{mean}}^{+}-pt_{e,\mathrm{mean}}^{+}\|_2$, and set the positive prototypes corresponding to this σ as the initialized ones.
Initialize $G_{M-\mathrm{MIDA}}$ according to (17).
End

Algorithm 4. Entire optimization process of M-MIDA.

Input: $B_{c,i}^{+}$ ($c \in \{1,\ldots,C\}$, $i \in \{1,\ldots,l_c^{+}\}$), α, d, convergence tolerance ξ, maximum iteration number t, maximum iteration number for the first step t_1.
Output: Transformation matrix $G_{M-\mathrm{MIDA}}$.
Initialize the positive prototypes $pt_{c,i}^{+}$ ($c \in \{1,\ldots,C\}$, $i \in \{1,\ldots,l_c^{+}\}$) and the transformation matrix $G_{M-\mathrm{MIDA}}$ according to Algorithm 3.
Iterative optimization:
Repeat:
First step: Fix $G_{M-\mathrm{MIDA}}$, for each class $c \in \{1,\ldots,C\}$ repeat the following iterations until there is no change in $pt_{c,i}^{+}$ ($i \in \{1,\ldots,l_c^{+}\}$) or the iteration number reaches t_1:
For each $i \in \{1,\ldots,l_c^{+}\}$, try all instances in $B_{c,i}^{+}$, find the one maximizes $T_{c,i}$ in (16) and select it as $pt_{c,i}^{+}$. After all $pt_{c,i}^{+}$ ($i \in \{1,\ldots,l_c^{+}\}$) have been selected, recalculate $pt_{c,\mathrm{mean}}^{+}$ according to (13).
Second step: Fix $pt_{c,i}^{+}$ ($c \in \{1,\ldots,C\}$, $i \in \{1,\ldots,l_c^{+}\}$) and $pt_{c,\mathrm{mean}}^{+}$ ($c \in \{1,\ldots,C\}$), operate eigenvalue decomposition on $S_{M-\mathrm{MIDA}}^{b}-\alpha S_{M-\mathrm{MIDA}}^{w}$, find the eigenvectors corresponding to top d largest eigenvalues and formulate $G_{M-\mathrm{MIDA}}$ according to (17).
Until the relative change of the objective function of (16) is less than ξ or the iteration number reaches t.

4.3. Time complexity

The analysis of the time complexity of M-MIDA is similar to that of B-MIDA. In the initialization step, the time complexity of kernel density estimation is $O(\sum_{c=1}^{C} n_{c,\mathrm{sum}}^{+}(\sum_{e\neq c,e=1}^{C} n_{e,\mathrm{sum}}^{+})D)$, which can be approximated as $O({n_{\mathrm{sum}}^{+}}^{2}D)$, where $n_{c,\mathrm{sum}}^{+}$ denotes the number of instances in all positive bags for class c, $n_{\mathrm{sum}}^{+}=\sum_{c=1}^{C} n_{c,\mathrm{sum}}^{+}$ denotes the number of instances in all bags. In each inner iteration, the time complexity of seeking positive prototypes is $O(n_{\mathrm{sum}}^{+}CDd^2)$. In each outer iteration, the time complexity of operating eigenvalue decomposition is $O(D^3)$. Let t_{in} and t_{out} denote the average number of inner iterations and the number of outer iterations, respectively, then the overall time complexity of solving M-MIDA is $O(n_{\mathrm{sum}}^{+}2D+t_{\mathrm{in}}t_{\mathrm{out}}n_{\mathrm{sum}}^{+}CDd^2+t_{\mathrm{out}}D^3)$.

5. Experiments

We adopt both synthetic and real-world datasets to evaluate the classification performance of MIDA. First, we respectively adopt two synthetic datasets (generated by ourselves), four LJ-*r.f.s* datasets and nine MILWEB datasets to compare MIDA with several competing feature extraction algorithms: PCA (principal component analysis) [29], LDA, MIDR, MidLABS and CLFDA. Among these competing feature extraction algorithms, PCA and LDA are classical ones and not for multiple-instance learning, whereas MIDR, MidLABS and CLFDA are for multiple-instance learning. Next, we compare B-MIDA with several representative multiple-instance classification algorithms and the above three multiple-instance feature extraction algorithms via experiments conducted on five benchmark datasets. Finally, we evaluate the classification performance of M-MIDA by making a comparison between M-MIDA and several competing algorithms through an image categorization task.

5.1. On synthetic datasets

In this subsection we adopt two Gaussian distributed synthetic datasets to evaluate the classification performance of B-MIDA and that of M-MIDA. The reason of adopting Gaussian distributed data in our evaluation is that both B-MIDA and M-MIDA are multiple-instance extensions of LDA, and that the projection of LDA is Bayesian optimal (it is the same to that obtained by adopting the Bayesian decision rule) when data are Gaussian distributed (the detailed proof can be found in Section 4.2.3 of [41]). For convenience, we term the synthetic dataset for the evaluation of B-MIDA as dataset one and that for M-MIDA as dataset two. For both datasets, each instance consists of 32 dimensions, of which the first two dimensions are relevant ones and the other 30 dimensions are noisy ones.

For dataset one, the two relevant dimensions are generalized as follows: both positive and negative instances are Gaussian distributed, where $\mu^{+}=[-4;4]$, $\mu^{-}=[4;-4]$, $\sum^{+}=\sum^{-}=[2\ 0;\ 0\ 2]$ denote the means for positive and negative instances, and the covariance matrices for positive and negative instances, respectively. The 30 noisy dimensions are independent with each other, with each drawn from a Gaussian with mean 0 and variance 9. Each bag (no matter it is positive or negative) contains eight instances. The definition of binary-class multiple-instance learning shows that a positive bag contains at least one positive instance, and for simplicity, we assume that there is one and only one positive instance in each positive bag. We generate one positive instance and seven negative instances, and then utilize their combination as a positive bag. We directly generate eight negative instances and utilize their combination as a negative bag. We generate 40 positive bags and 40 negative bags, of which half are for training and the other half are for testing.

According to the definition of multi-class multiple-instance learning, a bag is termed as positive for a given class if it contains at least one positive instance for this class, and there are no specific negative bags for each class because a positive bag for some class is simultaneously a negative bag for any other class.

For dataset two, suppose there are three classes in total, and the two relevant dimensions of positive instances for class 1, class 2, and class 3 are all Gaussian distributed, where $\mu_1^+ = [-4; 4]$, $\mu_2^+ = [4; \ -4]$, $\mu_3^+ = [8; 8]$, $\sum_1^+ = \sum_2^+ = \sum_3^+ = [2\ 0;\ 0\ 2]$ denote the relevant means of positive instances for class 1, class 2, class3, and the relevant covariance matrices of positive instances for class 1, class 2, class 3, respectively. For simplicity, we assume the two relevant dimensions of negative instances for three classes follow the same Gaussian distribution with $\mu^- = [-2; 15]$ and $\sum^- = [4\ 0;\ 0\ 4]$. For both positive and negative instances, the 30 noisy dimensions are independent with each other, with each drawn from the Gaussian with mean 0 and variance 9. Each bag (no matter it is positive or negative) contains eight instances. Given any class, we generate one positive instance and seven negative instances for this class, respectively, and then utilize their combination as a positive bag for this class. We generate 40 positive bags for each class, and utilize half of them for training and the other half for testing.

Since positive prototypes play an important role in MIDA, before operating performance evaluation on MIDA, first we check out the ability of kernel density estimation in identifying positive prototypes. In Table 1, for both heavily corrupted datasets ("heavily corrupted" means that the number of noisy dimensions is much larger than that of relevant ones), we give the true number of positive prototypes in the training set and that of correctly identified ones via kernel density estimation. Through Table 1, we find that kernel density estimation may correctly identify more than 65% of positive prototypes for these two heavily corrupted datasets, and hence adopting kernel density estimation to initialize positive prototypes for MIDA is an acceptable choice.

To effectively evaluate the feature extraction performance of B-MIDA, we compare it with two classical supervised feature extraction algorithms: PCA and LDA, as well as three multiple-instance feature extraction algorithms: MIDR, MidLABS, and CLFDA. For LDA, we simply utilize bag labels as instance labels. Considering there are only two relevant dimensions for dataset one, for simplicity, we fix the subspace dimensionality of all feature extraction algorithms but LDA as 2. For LDA, since the upper bound of its subspace dimensionality equals the number of classes minus 1, which is always 1 for binary-class multiple-instance learning, we fix the LDA subspace as one-dimensional. The above competing algorithms are only for feature extraction, thus we need to adopt additional classifiers to operate classifications. Here the adopted classifier is Citation-KNN [19], and the number of citers and that of references are set in the same way to that of [19]. In Table 2, we give the classification accuracies of Citation-KNN on the original data, the PCA, LDA, MIDR, MidLABS, CLFDA, and B-MIDA transformed data. The best result is represented in bold face.

Since MIDR and MidLABS were just for binary-class multiple-instance learning, on dataset two we only compare M-MIDA with PCA, LDA, and CLFDA. We fix the subspace dimensionality of all competing algorithms as 2, and adopt the DD-SVM classifier as the classification tool for them (the original Citation-KNN classifier [19] was only designed for binary-class classifications). The classification results of DD-SVM on the original data, the PCA, LDA, CLFDA, and M-MIDA transformed data are shown in Table 3. The best result is represented in bold face.

Table 1
Ability of kernel density estimation in identifying positive prototypes.

Dataset	One	Two
# True positive prototypes	20	60
# Correctly identified Positive prototypes	13	41
Identifying accuracy (%)	65.0	68.3

Table 2
Classification accuracies (%) on synthetic dataset one.

Algorithm	Result
Citation-KNN [19]	52.5
PCA [29]+Citation-KNN	67.5
LDA [29]+Citation-KNN	57.5
MIDR [26]+Citation-KNN	72.5
MidLABS [27]+Citation-KNN	70.0
CLFDA [28]+Citation-KNN	67.5
B-MIDA+Citation-KNN	**75.0**

Table 3
Classification accuracies (%) on synthetic dataset two.

Algorithm	Result
DD-SVM [7]	71.7
PCA [29]+DD-SVM	73.3
LDA [29]+DD-SVM	66.7
CLFDA [28]+DD-SVM	75.0
M-MIDA+DD-SVM	**83.3**

The results shown in Tables 2 and 3 demonstrate that we may obtain some performance improvement through the preprocessing of any multiple-instance feature extraction algorithm (MIDR, MidLABS, CLFDA, B-MIDA, and M-MIDA). In particular, through the preprocessing of B-MIDA and M-MIDA, we obtain the best results. PCA also improve the classification accuracies by reducing noises. LDA performs the worst among all competing feature extraction algorithms, which shows the significant difference between multiple-instance learning and supervised learning and demonstrates that simply using bag labels to represent instance labels is not a reasonable choice.

5.2. On LJ-r.f.s and MILWEB datasets

In this subsection, we operate performance evaluation on B-MIDA via two binary-class multiple-instance learning datasets: LJ-*r.f.s* and MILWEB. We first give brief descriptions of these two datasets, and then give the classification results of B-MIDA and several competing algorithms on the two datasets.

5.2.1. The LJ-r.f.s dataset

The LJ-*r.f.s* dataset was constructed by Amar et al. [42] when they were studying to generate artificial datasets to evaluate the classification performance of multiple-instance learners. They gave a way of generating chemically realistic artificial datasets, of which some actors that may affect the datasets (e.g., the number of conformers, the number of relevant features and their degrees of relevance) could be controlled by the generators. Within the name "LJ-*r.f.s*", LJ is short for the Lennard–Jones potential and adopted as the basis for mimic intermolecular interactions in the generation of artificial datasets; *r*, *f*, and *s* are three variables and denote the number of relevant features, the number of all features (i.e., the dimensionality), and the number (or levels) of different scale factors used for relevant features, respectively.

LJ-*r.f.s* is the name of a set of datasets, and a specific dataset can be obtained by fixing the values of *r*, *f*, and *s*. We utilize four LJ-*r.f.s* datasets: LJ-150.283.2, LJ-150.283.4, LJ-150.283.10 and LJ-150.283.15 to evaluate and compare the abilities of different feature extraction algorithms in improving classification accuracies. There are two reasons for selecting these four datasets, of which the first one is that

Table 4
Brief description of nine MILWEB datasets.

Datasets	v1	v2	v3	v4	v5	v6	v7	v8	v9
# Training positive bags	17	18	14	56	62	60	39	35	37
# Instances in training positive bags	488	499	377	1710	2105	1916	1258	1063	1229
# Testing positive bags	4	3	7	33	27	29	16	20	18
# Instances in testing positive bags	68	57	179	956	561	750	452	647	481
# Training negative bags	58	57	61	19	13	15	36	40	38
# Instances in training negative bags	1724	1720	2137	581	441	546	1142	1120	1092
# Testing negative bags	34	35	31	5	11	9	22	18	20
# Instances in testing negative bags	1143	1147	730	176	316	211	571	593	621
# Features	20	20	20	20	20	20	20	20	20

Table 5
Classification accuracies (%) on four LJ-*r.f.s* datasets.

Algorithm	LJ-150.283.2	LJ-150.283.4	LJ-150.283.10	LJ-150.283.15
Citation-KNN [19]	71.5	68.5	70.0	72.0
PCA [29]+Citation-KNN	73.5	73.5	76.0	72.0
LDA [29]+Citation-KNN	68.0	67.5	70.5	71.5
MIDR [26]+Citation-KNN	79.5	79.0	80.5	80.5
MidLABS [27]+Citation-KNN	78.0	79.0	75.5	78.0
CLFDA [28]+Citation-KNN	80.0	79.0	81.0	78.5
B-MIDA+Citation-KNN	**80.5**	**82.0**	**82.5**	**82.0**

the dimensions of these four datasets are high (i.e., 283 dimensions) and about half of them are relevant ones (i.e., 150 relevant dimensions), which makes these four datasets very suitable to testify the feature extraction performance of candidate dimensionality reduction algorithms. The second reason is that without any preprocessing, the classification accuracies on these four datasets are very low, and hence, the preprocessing of feature extraction is very necessary. There are 60 positive bags and 140 negative bags for each of the above our datasets, and the number of instances in positive bags and that in negative bags are 250 and 546, respectively. In average, there are approximately 3.98 instances in each bag. For LJ-150.283.2, LJ-150.283.4, LJ-150.283.10 and LJ-150.283.15, the levels of different scale factors used for relevant features are 2, 4, 10 and 15, respectively.

5.2.2. The MILWEB dataset

The MILWEB dataset[1] was constructed by Zhou et al. [43] when they were investigating the web index recommendation problem. They treated web index recommendation as a multiple-instance learning problem and found that it was hopeful to develop user-adaptive intelligent web browsers by the study of this problem. A web index page is a web page which contains plentiful information but only provides titles or brief summaries while leaving the detailed presentation to its linked pages. A positive web index page is such a page that the user is interested in at least one of its linked pages. In contrast, a negative web index page is the one that none of its linked pages interests the user. Therefore, each index page can be regarded as a bag and its linked pages can be regarded as the instances in the bag. The goal is to label unseen web index pages as positive or negative.

The web index pages in MILWEB were collected and labeled by nine volunteers according to their interests, and hence there are nine binary-class datasets in total. The biggest bag contains 200 instances, while the smallest one contains only four instances. In average, each bag contains 30.29 (3423/113) instances. Each instance is described by the 1st to 15th most frequent terms appearing in the corresponding linked page. TFIDF are used to represent the frequent terms, and normalization is performed instance by instance. For each of the nine datasets, 75 bags were randomly selected for training while the remaining 38 bags were used for testing. We use the processed data such that all bags and instances are as same as those used in [43]. A brief description of the nine MILWEB datasets is given in Table 4.

5.2.3. Classification results

Similar to the experiments in Section 5.1, on the four LJ-*r.f.s* datasets and nine MILWEB datasets we also compare B-MIDA with five competing feature extraction algorithms: PCA, LDA, MIDR, MidLABS and CLFDA, of which PCA and LDA are supervised ones, MIDR, MidLABS and CLFDA are multiple-instance ones. Citation-KNN is adopted as the subsequent classifier for all algorithms. The nine MILWEB datasets had been separated into the training set and testing set [43], and we simply utilize the same separated data in our experiments. The four LJ-*r.f.s* datasets were not separated into the training set and testing set in advance, and we operate performance evaluations on them via 10-fold cross validations.

For the four LJ-*r.f.s* datasets, the candidate set of the subspace dimensionality d is set as {5, 10, 20, 30, …, 140, 150}, while for the nine MILWEB datasets that of d are set as {1, 3, 5, …, 13, 15}. For the four LJ-*r.f.s* and nine MILWEB datasets, we set the candidate set of α as {0.001, 0.01, 0.1, 1, 10, 100}, and then select α and d jointly by performing five-fold cross validations on the training set. The classification results for the four LJ-*r.f.s* datasets and nine MILWEB datasets are shown in Tables 5 and 6, respectively, of which the best results are represented in bold faces.

Besides classification accuracies, in Tables 7 and 8, we also give the dimensionality reduction results of different feature extraction algorithms on four LJ-*r.f.s* and nine MILWEB datasets. We show the original dimensions and the average reduced dimensions of different feature extraction algorithms when they obtain their best classification results. We do not give the dimensionality reduction results of LDA, because the upper bound of the subspace dimensionality of LDA equals the number of classes minus 1, which is 1 and thus trivial for binary-class multiple-instance learning.

The classification results in Table 5 demonstrate that on most datasets, B-MIDA significantly outperforms the two supervised feature

[1] http://lamda.nju.edu.cn/data_MILWEB.ashx

Table 6
Classification accuracies (%) on nine MILWEB datasets.

Algorithm	v1	v2	v3	v4	v5	v6	v7	v8	v9
Citation-KNN [19]	92.1	86.8	86.8	92.1	76.3	81.6	55.3	65.8	68.4
PCA [29]+Citation-KNN	94.7	92.1	86.8	89.5	78.9	86.8	65.8	68.4	**78.9**
LDA [29]+Citation-KNN	81.6	86.8	84.2	84.2	76.3	78.9	60.5	63.2	76.3
MIDR [26]+Citation-KNN	94.7	**94.7**	86.8	**94.7**	78.9	86.8	68.4	**73.7**	71.1
MidLABS [27]+Citation-KNN	**97.4**	**94.7**	**89.5**	**94.7**	84.2	86.8	**73.7**	65.8	71.1
CLFDA [28]+Citation-KNN	**97.4**	**94.7**	**89.5**	92.1	**86.9**	84.2	65.8	71.1	71.1
B-MIDA+Citation-KNN	**97.4**	92.1	**89.5**	**94.7**	81.6	**89.5**	71.1	**73.7**	73.7

Table 7
Dimensionality reduction results on four LJ-*r.f.s* datasets.

Dimensions	LJ-150.283.2	LJ-150.283.4	LJ-150.283.10	LJ-150.283.15	Average
Original	283.0	283.0	283.0	283.0	283.0
PCA [29]	12.5	9.5	9.0	16.5	11.9
MIDR [26]	23.0	26.5	21.0	27.0	24.4
MidLABS [27]	21.0	19.5	17.5	16.0	18.5
CLFDA [28]	55.5	63.0	45.5	32.0	49.0
B-MIDA	31.0	36.0	38.0	33.0	34.5

Table 8
Dimensionality reduction results on nine MILWEB datasets.

Dimensions	v1	v2	v3	v4	v5	v6	v7	v8	v9	Average
Original	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
PCA [29]	15.0	11.0	3.0	13.0	15.0	9.0	1.0	5.0	9.0	9.0
MIDR [26]	9.0	7.0	9.0	11.0	1.0	3.0	1.0	13.0	9.0	7.0
MidLABS [27]	1.0	15.0	3.0	7.0	5.0	9.0	1.0	1.0	3.0	5.0
CLFDA [28]	3.0	13.0	11.0	11.0	1.0	9.0	15.0	5.0	11.0	8.8
B-MIDA	13.0	1.0	9.0	13.0	7.0	7.0	5.0	1.0	7.0	7.0

Table 9
Brief description of five benchmark datasets.

Datasets	Musk1	Musk2	Elephant	Fox	Tiger
# Positive bags	47	39	100	100	100
# Instances in positive bags	207	1017	762	647	544
# Negative bags	45	63	100	100	100
# Instances in negative bags	269	5581	629	673	676
# Average instances in each bag	5.17	64.69	6.96	6.60	6.10
# Features	166	166	230	230	230

extraction algorithms: PCA and PDA, and slightly outperforms the three multiple-instance ones: MIDR, MidLABS and CLFDA. Table 7 shows that the subspace dimensionality of B-MIDA is higher than that of most competing algorithms (it is only smaller than that of CLFDA on three datasets), which demonstrates that the superiority of B-MIDA in classification accuracies on the four LJ-*r.f.s* datasets is achieved at the cost of more subspace dimensionality.

On the nine MILWEB datasets, B-MIDA obtains very promising learning performances, i.e., Table 6 shows that B-MIDA achieves the highest, the second highest, and the third highest classification accuracies on five, two, and two datasets, respectively. The dimensionality reduction ability of B-MIDA is also very competitive. Table 8 shows that the average reduced dimensionality of B-MIDA is the second lowest, i.e., it is just higher than that of MidLABS.

5.3. On benchmark datasets

In this subsection we evaluate the classification performance of B-MIDA on five benchmark datasets: Musk1, Musk2, Elephant, Fox and Tiger. The Musk1 and Musk2 datasets are utilized to describe molecules. A given molecule is considered as a bag and its each steric configuration (i.e., molecular shape) is considered as an instance. Elephant, Fox and Tiger are three binary-class image annotation datasets. Binary-class image annotation refers to deciding whether a given image belongs to the predefined category or not. For the three image annotation datasets, each image is segmented into several regions, and we treat each image as a bag and each region of the image as an instance. A brief description of the above five benchmark datasets is given in Table 9. For other details of Musk1 and Musk2, refer to [1]; for other details of Elephant, Fox and Tiger, refer to [10].

Table 10
Classification accuracies (%) on five benchmark datasets.

Algorithm	Musk1	Musk2	Elephant	Fox	Tiger
ID-APR [1]	92.4	89.2	N/A	N/A	N/A
DD [2]	88.0	84.0	N/A	N/A	N/A
EM-DD [18]	84.8	84.9	78.3	56.1	72.1
Citation-KNN [19]	90.0	89.1	87.8	62.0	82.5
MI-SVM [10]	77.9	84.3	81.4	59.4	84.0
mi-SVM [10]	87.4	83.6	82.0	58.2	78.9
MI-Kernel [20]	88.0	89.3	84.3	60.3	84.2
MI-Graph [21]	90.0	90.0	85.1	61.2	81.9
mi-Graph [21]	88.9	90.3	86.8	61.6	86.0
MIForests [3]	85.0	82.0	84.0	64.0	82.0
MIDR [26]+Citation-KNN	95.8	93.6	91.2	78.5	87.5
MidLABS [27]+Citation-KNN	97.6	93.5	88.2	**81.3**	83.0
CLFDA [28]+Citation-KNN	92.1	90.3	89.4	71.6	84.4
B-MIDA+Citation-KNN	**98.8**	**96.9**	**94.8**	81.1	**90.5**
B-MIDA+KNN	90.9	85.5	94.1	76.8	90.1

To make a sufficient comparison, in Table 10, we give the classification accuracies of B-MIDA and 13 competing algorithms, of which 10 are multiple-instance classification algorithms, 3 are multiple-instance feature extraction algorithms (MIDR, MidLABS, and CLFDA). Note that there are two different versions for both SVM-based algorithms [10] and graph-based algorithms [21], one with a prefix "MI-" and the other with a prefix "mi-". The difference between MI-SVM and

Table 11
p-Values and *t*-test results on five benchmark datasets.

Algorithm	Musk1	Musk2	Elephant	Fox	Tiger
MIDR [26]+Citation-KNN	0.015 1	1.19e−5 1	5.85e−5 1	0.026 1	0.020 1
MidLABS [27]+Citation-KNN	0.058 0	4.55e−4 1	0.002 1	0.457 0	6.44e−4 1
CLFDA [28]+Citation-KNN	0.001 1	4.14e−6 1	1.40e−4 1	4.49e−7 1	6.08e−4 1

mi-SVM is that the former defines the classification margin in the bag level, while the latter defines the classification margin in the instance level. The difference between MI-Graph and mi-Graph is that the former explicitly maps each bag to an undirected graph and uses a graph kernel to distinguish positive and negative bags, while the latter implicitly constructs graphs by calculating affinity matrices and defines the graph kernel by considering the clique information.

Since the most competing algorithms are evaluated via 10-fold cross validations in their original papers, in order to make our results comparable to theirs, we also evaluate B-MIDA via 10-fold cross validations. There are two free parameters in B-MIDA, i.e., the trade-off parameter α and the subspace dimensionality d. We set the candidate set of α as {0.001, 0.01, 0.1, 1, 10, 100}, that of d as {5, 10, 15, …, 95, 100}, and then select α and d jointly by performing five-fold cross validations on the training set. All multiple-instance classification algorithms except for Citation-KNN [19] are evaluated via 10-fold cross validations in their original papers, and we simply replicate their results for reference. For Citation-KNN, we re-evaluate it via 10 times 10-fold cross validations and show the re-evaluated results. For B-MIDA and the three competing feature extraction algorithms, Citation-KNN is adopted to operate subsequent classifications. Moreover, since B-MIDA has the ability of discovering positive prototypes, we also utilize the KNN classifier [29] in our classifications, by treating positive prototypes as positive examples and negative prototypes as negative examples, respectively. The best results in Table 10 are represented in bold faces.

The results shown in Table 10 demonstrate that the feature extraction performance of B-MIDA is very promising, because B-MIDA+Citation-KNN achieves the highest classification accuracies on four benchmark datasets and the second highest classification accuracy on the other benchmark dataset (here we just compare the classification accuracies without considering the statistical significance of the difference in classification accuracies; the results of statistical significance analysis are given in following paragraphs). Moreover, the ambiguity elimination performance of B-MIDA is fine as well, since B-MIDA+KNN achieves moderate classification accuracies on most benchmark datasets.

In order to make a sufficient comparison, besides classification accuracies, we also assess the statistical significance of the difference on classification accuracies between B-MIDA and the other three competing feature extraction algorithms. We perform the statistical significance evaluation by operating the *t*-test and calculating the corresponding *p*-values. We give the *t*-test results at the 0.05 significance level (0.05 is a popularly adopted value in many real applications), where the result "1" indicates that the difference is statistically significant at the 0.05 level, while "0" indicates that there is no statistically significant difference at the 0.05 level. The *p*-values and *t*-test results are shown in Table 11, where each *p*-value is shown in the first line with the corresponding *t*-test result shown in the second line.

The *p*-values and *t*-test results shown in Table 11 demonstrate that at the 0.05 significance level, the performance of B-MIDA and that of MIDR, as well as the performance of B-MIDA and that of CLFDA, are both statistically different from each other on all five datasets, whereas the performance of B-MIDA and that of MidLABS are statistically different from each other only on three datasets (except Musk1 and Fox). One possible reason for the difference between B-MIDA and MIDR is that they are designed according to different principles, i.e., B-MIDA aims at simultaneously maximizing the between-class scattering and minimizing the within-class one, whereas MIDR aims at keeping the consistency of the posterior probability of each bag being positive to the corresponding bag label. CLFDA adopts local Fisher criterion and also aims at maximizing the between-class scattering and minimizing the within-class one, but it pre-labels all instances by simply utilizing bag labels to represent instance labels before detecting false positive instances, the unreasonableness of the pre-labeling process may affect the feature extraction performance of CLFDA and lead to significant differences between CLFDA and B-MIDA on classification accuracies. The design principles of MidLABS and B-MIDA are very close to each other, because both of them are derived from LDA and can be treated as multiple-instance extensions of LDA. However, they also have some differences, mainly in that they construct the scattering matrices from different levels, i.e., MidLABS is from the bag level, and in contrast B-MIDA is from the instance level.

Table 12
Dimensionality reduction results on five benchmark datasets.

Dimensions	Musk1	Musk2	Elephant	Fox	Tiger
Original	166.0	166.0	230.0	230.0	230.0
MIDR [26]	55.6	37.2	48.8	41.0	62.9
MidLABS [27]	28.9	17.4	19.8	27.2	29.6
CLFDA [28]	73.6	69.1	56.7	49.1	65.2
B-MIDA	10.6	12.2	21.5	23.4	31.7

Besides classification accuracies, in Table 12, we also give the dimensionality reduction results of different feature extraction algorithms. We show the original dimensions and the average reduced dimensions of different feature extraction algorithms when they obtain the best classification results. Through Table 12, we find that the dimensionality reduction ability of B-MIDA is very remarkable, which is comparable to that of MidLABS and more powerful than that of MIDR and CLFDA on most benchmark datasets.

To evaluate the impact of the trade-off parameter α on the classification accuracy of B-MIDA (plus Citation-KNN), in Fig. 2 we show the variation of the classification accuracies of B-MIDA w.r.t. different α values on five benchmark datasets, respectively. Besides classification accuracies, we also give the dimensionality reduction results of B-MIDA, i.e., we show the subspace dimensionality of B-MIDA in brackets right after the classification accuracies. Note that the best results shown in Fig. 2 are not exactly the same to that shown in Table 10, because we conduct experiments via 10-fold cross validations and give the averaged results, thus for a given value of α, we may get the optimal results for some folds, and sub-optimal results for the other folds. Through Fig. 2, we find that the performance of B-MIDA is insensitive to α, which implies that

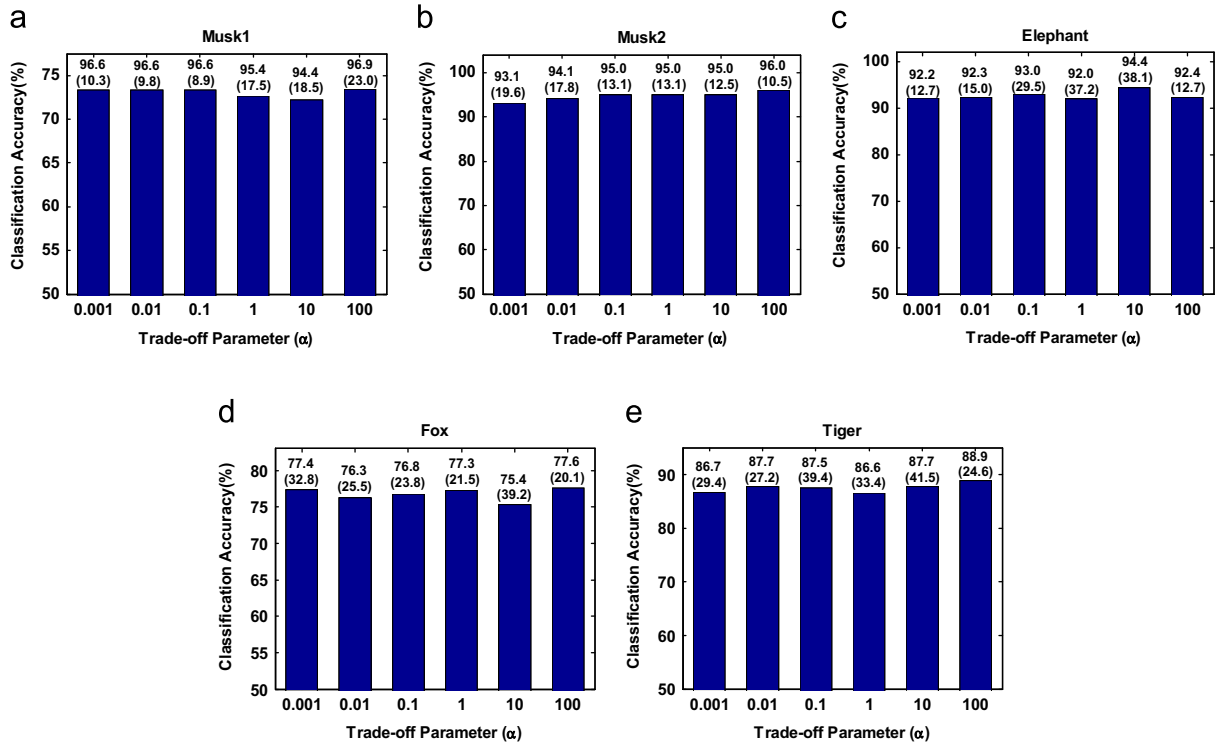

Fig. 2. Classification performances and dimensionality reduction results of B-MIDA w.r.t. trade-off parameters on five benchmark datasets. The dimensionality reduction results are given in brackets.

maximizing the between-class scattering plays a more important role than minimizing the within-class one in improving discriminations, and the latter may just accommodate the discriminative ability of B-MIDA slightly.

5.4. On Corel dataset

In this subsection we utilize an image categorization task to test the feature extraction performance of M-MIDA. Image categorization refers to labeling images into predefined categories and is an important application domain of multiple-instance learning. The Corel dataset consists of 2000 images taken from 20 CD-ROMs published by the Corel Corporation. Each CD-ROM contains 100 images representing a distinct concept. Therefore, the dataset has 20 thematically diverse image categories (classes), each containing 100 images (bags). Images are in JPEG format of size 384×256 or 256×384. The category names are listed in Table 13 along with the identifiers (IDs) for 20 classes.

To segment an image, the system first partitions the image into non-overlapping blocks of size 4×4 pixels. A six-dimensional feature vector is then extracted for each block. Three of them are the average color components in a block. The other three represent the square root of energy in the high-frequency bands of the wavelet transforms, which may show the variations in different directions and hence capture the texture properties.

The *k*-means algorithm is applied to group the feature vectors into clusters each corresponding to a region (an instance) in the segmented image [7]. The number of regions in an image can vary depending on the complexity of the image content. Table 13 lists the average number of regions per image (average number of instances per bag) for each category. After segmentation, three extra features are computed for each region to describe the shape properties. As a result, each region in any image (each instance in any bag) is characterized by a nine-dimensional feature vector, with three characterizing the color properties, three characterizing the texture properties, and three characterizing the shape properties of the region.

Table 13
Brief description of the Corel dataset.

Category ID	Category name	# Instances per bag (# regions per image)
0	African people & villages	4.84
1	Beach	3.54
2	Historical building	3.10
3	Buses	7.59
4	Dinosaurs	2.00
5	Elephants	3.02
6	Flowers	4.46
7	Horses	3.89
8	Mountains & glaciers	3.38
9	Food	7.24
10	Dogs	3.80
11	Lizards	2.80
12	Fashion models	5.19
13	Sunset scenes	3.52
14	Cars	4.93
15	Waterfalls	2.56
16	Antique furniture	2.30
17	Battle ships	4.32
18	Skiing	3.34
19	Desserts	3.65

In most previous studies, the original Corel dataset was utilized as two datasets. The first dataset contains only 10 categories and is usually termed as the 1000-Image dataset, while the second one contains all 20 categories and is termed as the 2000-Image dataset. We utilize the same experimental routine to that described in [7]. In particular, for both datasets (1000-Image and 2000-Image), images of each category are randomly separated into two parts with the same size, with one part utilized for training and the other part for

testing. M-MIDA has two free parameters, i.e., the trade-off parameter α and the subspace dimensionality d. We set the candidate set of α as {0.001, 0.01, 0.1, 1, 10, 100}, the candidate set of d as {1, 2, 3, 4, 5, 6, 7, 8, 9}, and then select α and d jointly by performing five-fold cross validations on the training set. We repeat the above random separation independently for five times, and report the average results for five random separations. For M-MIDA, we adopt the DD-SVM classifier [7] to perform subsequent classifications. Moreover, since M-MIDA has the ability of finding positive prototypes for each class, we adopt the SVM classifier to perform classifications as well (by utilizing the positive prototypes for each class as samples of this class). Among MIDR, MidLABS and CLFDA, only CLFDA can be directly applied for multi-class multiple-instance feature extraction, thus we only give the classification results of CLFDA+DD-SVM. The one-vs-one strategy is adopted to perform classifications for both SVM and DD-SVM. For reference, we also give the classification results of several multi-class multiple-instance learning algorithms reported in the literatures [3,4,7,10,20,21,44,45]. Following the style of most of these competing algorithms, besides classification accuracies, we also give the 95% confidence intervals of these accuracies. The classification results are shown in Table 14, of which the best results are represented in bold faces. Besides classification accuracies, in Table 15 we also give the dimensionality reduction results of M-MIDA and CLFDA, and the results demonstrate that both M-MIDA and CLFDA can reduce the dimensions to about half of the original ones.

Since the original data consist of only nine dimensions, we have to acknowledge that state-of-the-art feature extraction for visual extraction cannot be obtained on the Corel dataset. However, it still has some use for reference by comparing the classification results w.r.t. the M-MIDA preprocessed data and that w.r.t. the original data. Through Table 14, we see that M-MIDA+DD-SVM obtains the highest classification accuracy on the 1000-Image dataset, and the third highest classification accuracy (it is very close to the highest classification accuracy) on the 2000-Image dataset among all competing algorithms. Moreover, M-MIDA+DD-SVM obtains obvious performance improvements over DD-SVM on both 1000-Image and 2000-Image datasets, and these improvements demonstrate the effectiveness of M-MIDA in extracting discriminative features. Finally, the classification performance of M-MIDA+SVM is comparable to that of most SVM-related algorithms, which shows that the ambiguity elimination ability of M-MIDA is fine as well.

Similar to the experiments conducted on benchmark datasets, in Fig. 3 we show the variation of M-MIDA's classification accuracies w.r.t. different trade-off parameters on the Corel dataset, and give the dimensionality reduction results of M-MIDA in brackets right after the accuracies. Since the accuracies are averaged over five random separations and each α may not correspond to the best results for all separations, the highest classification accuracies shown in Fig. 3 are not exactly the same to that shown in Table 14. The results shown in Fig. 3 are very similar to that in Fig. 2, both of them show the superiority of the between-class scattering over the within-class one in extracting discriminative features and the slight accommodating ability of the within-class scattering in improving discriminations.

Table 14
Classification accuracies (%) on the Corel dataset.

Algorithm	1000-Image	2000-Image
DD-SVM [7]	81.5 [78.5, 84.5]	67.5 [66.1, 68.9]
MI-SVM [10]	74.7 [74.1, 75.3]	54.6 [53.1, 56.1]
MissSVM [44]	78.0 [75.8, 80.2]	65.2 [62.0, 68.3]
k-means-SVM [45]	69.8 [67.9, 71.7]	52.3 [51.6, 52.9]
MILES [4]	82.6 [81.4, 83.7]	68.7 [67.3, 70.1]
MI-Kernel [20]	81.8 [80.1, 83.6]	72.0 [71.2, 72.8]
MI-Graph [21]	83.9 [81.2, 85.7]	**72.1 [71.0, 73.2]**
mi-Graph [21]	82.4 [80.2, 82.6]	70.5 [68.7, 72.3]
MIForests [3]	59.2 [57.5, 60.9]	66.8 [64.8, 68.8]
CLFDA[28]+DD-SVM	83.0 [81.3, 84.7]	68.3 [66.1, 70.5]
M-MIDA+DD-SVM	**85.1 [83.2, 87.0]**	71.8 [70.2, 73.4]
M-MIDA+SVM	74.9 [73.0, 76.8]	64.1 [61.9, 66.3]

Table 15
Dimensionality reduction results on the Corel dataset.

Dimensions	1000-Image	2000-Image
Original	9.0	9.0
CLFDA [28]	5.4	3.8
M-MIDA	4.2	4.8

6. Conclusions

In this paper we propose two MIDA dimensionality reduction algorithms, i.e., B-MIDA and M-MIDA, to cope with the feature extraction problem for binary-class and multi-class multiple-instance learning tasks, respectively. The feature extraction and

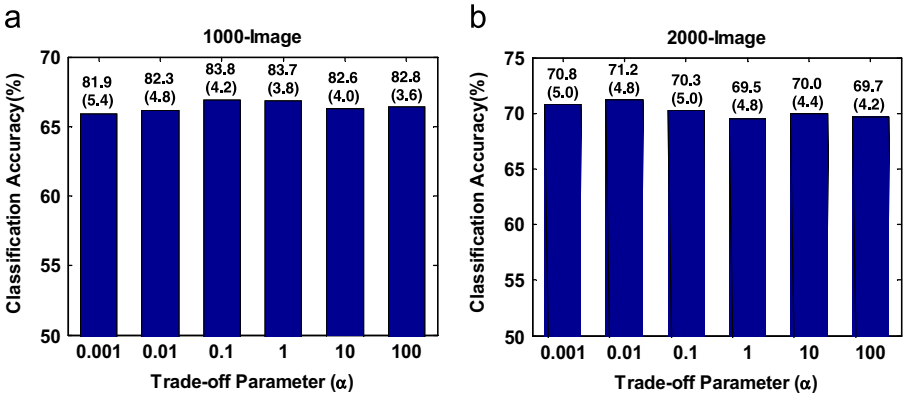

Fig. 3. Classification performances and dimensionality reduction results of M-MIDA w.r.t. trade-off parameters on the Corel dataset. The dimensionality reduction results are given in brackets.

ambiguity elimination abilities of the above two MIDA algorithms are empirically justified via extensive experiments.

There are still some limits in the current work. Firstly, MIDA is an extension of LDA, LDA can find the Bayesian optimal projection when data are Gaussian distributed (in fact, LDA also works well for many non-Gaussian distributed data, but the projection of LDA for non-Gaussian distributed data is not Bayesian optimal), and some limits of Gaussian distribution such as the unavailability for multi-modal data still exist in MIDA. To make up for this limit, trying to use local learning models to construct scattering matrices may be a feasible choice. Secondly, LDA assumes that instances derived from the same class are i.i.d. variables, and MIDA has the same assumption as well. The i.i.d. assumption ignores the structural information among instances within a bag, and we may adopt the structural information to improve the feature extraction performance of MIDA in the future. Thirdly, MIDA is just a linear algorithm, and there are requirements to extend it to nonlinear versions (e.g., kernel versions) to obtain more powerful discriminative abilities.

Conflict of interest

None declared.

Acknowledgments

The authors would like to thank Ming Li from Nanjing University and Christian Leistner from Graz University of Technology for their valuable feedback. The project is supported by Special/Youth Foundation of Taiyuan University of Technology (No. 2012L088), and Natural Science Foundation of Shanxi Province (No. 2013011035-4).

References

[1] T.G. Dietterich, R.H. Lathrop, T. Lozano-Perez, Solving the multiple instance problem with axis-parallel rectangles, Artif. Intell. 89 (1) (1997) 31–71.
[2] O. Maron, A.L. Ratan, Multiple-instance learning for natural scene classification, in: Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 341–349.
[3] C. Leistner, A. Saffari, H. Bischof, MIForests: multiple-instance learning with randomized trees, in: Proceedings of the 11th European Conference on Computer Vision, 2010, pp. 29–42.
[4] Y. Chen, J. Bi, J.Z. Wang, MILES: multiple-instance learning via embedded instance selection, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 1931–1947.
[5] Z.H. Zhou, M.L. Zhang, Multi-instance multi-label learning with application to scene classification, Adv. Neural Inf. Process. Syst. (2007) 1609–1616.
[6] Q. Zhang, S.A. Goldman, W. Yu, J. Fritts, Content-based image retrieval using multiple-instance learning, in: Proceedings of the 19th International Conference on Machine Learning, 2002, pp. 682–689.
[7] Y. Chen, J.Z. Wang, Image categorization by learning and reasoning with regions, J. Mach. Learn. Res. 5 (2004) 913–939.
[8] W.J. Li, D.Y. Yeung, MILD: multiple-instance learning via disambiguation, IEEE Trans. Knowl. Data Eng. 22 (1) (2010) 76–89.
[9] J.Y. Chiang, S.R. Cheng, Multiple-instance content-based image retrieval employing isometric embedded similarity measure, Pattern Recognit. 42 (1) (2009) 158–166.
[10] S. Andrews, I. Tsochantaridis, T. Hofmann, Support vector machines for multiple-instance learning, Adv. Neural. Inf. Process. Syst. (2003) 561–568.
[11] S. Ray, M. Craven, Supervised versus multiple instance learning: an empirical comparison, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 697–704.
[12] O. Maron, T. Lozano-Perez, A framework for multiple-instance learning, Adv. Neural Inf. Process. Syst. (1998) 570–576.
[13] Q. Tao, S. Scott, N.V. Vinodchandran, T.T. Osugi, SVM-based generalized multiple-instance learning via approximate box counting, in: Proceedings of the 21st International Conference on Machine Learning, 2004, pp. 799–806.
[14] J. Bi, 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, pp. 1–8.
[15] G. Fung, M. Dundar, B. Krishnapuram, R.B. Rao, Multiple instance learning for computer aided diagnosis, Adv. Neural Inf. Process. Syst. (2007) 425–432.
[16] G. Ruffo, Learning single and multiple decision trees for security applications (PhD thesis), University of Turin, Italy, 2000.
[17] Z.H. Zhou, Multi-instance Learning: A Survey, AI Lab, Department of Computer Science and Technology, Nanjing University, Technical Report, 2004.
[18] Q. Zhang, S.A. Goldman, EM-DD: an improved multiple-instance learning technique, Adv. Neural Inf. Process. Syst. (2002) 1073–1080.
[19] J. Wang, J.D. Zucker, Solving the multiple-instance problem: a lazy learning approach, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 1119–1125.
[20] T. Gartner, P.A. Flach, A. Kowalczyk, A.J. Smola, Multi-instance kernels, in: Proceedings of the 19th International Conference on Machine Learning, 2002, pp. 179–186.
[21] Z.H. Zhou, Y.Y. Sun, Y.F. Li, Multi-instance learning by treating instances as non-i.i.d. samples, in: Proceedings of the 26th International Conference on Machine Learning, 2009, pp. 1249–1256.
[22] A. Zafra, M. Pechenizkiy, S. Ventura, ReliefF-MI: an extension of ReliefF to multiple-instance learning, Neurocomputing 75 (1) (2012) 210–218.
[23] I. Kononenko, Estimating attributes: analysis and extension of relief, in: Proceedings of the 7th European Conference on Machine Learning, 1994, pp. 171–182.
[24] Y. Li, D.M.J. Tax, R.P.W. Duin, M. Loog, Multiple-instance learning as a classifier combining problem, Pattern Recognit. 46 (3) (2013) 865–874.
[25] T.Z. Zhang, S. Liu, C.S. Xu, H.Q. Lu, M^4L: maximum margin multi-instance multi-cluster learning for scene modeling, Pattern Recognit. 46 (10) (2013) 2711–2723.
[26] Y.Y. Sun, M.K. Ng, Z.H. Zhou, Multi-instance dimensionality reduction, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, 2010, pp. 587–592.
[27] W. Ping, Y. Xu, K.X. Ren, C.H. Chi, F.R. Shen, Non-i.i.d. multi-instance dimensionality reduction by learning a maximum bag margin subspace, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, 2010, pp. 551–556.
[28] S. Kim, S. Choi, Local dimensionality reduction for multiple instance learning, in: IEEE International Workshop on Machine Learning for Signal Processing, 2010, pp. 13–29.
[29] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., John Wiley and Sons, New York, USA, 2000.
[30] Y.Q. Jia, F.P. Nie, C.S. Zhang, Trace ratio problem revisited, IEEE Trans. Neural Networks 20 (4) (2009) 729–735.
[31] Y. Guo, S. Li, J. Yang, T. Shu, L. Wu, A generalized Foley–Sammon transform based on generalized fisher discriminant criterion and its application to face recognition, Pattern Recogn. Lett. 24 (1–3) (2003) 147–158.
[32] M. Sugiyama, Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis, J. Mach. Learn. Res. 8 (2007) 1027–1061.
[33] W.K. Yang, J.G. Wang, M.W. Ren, J.Y. Yang, Feature extraction based on Laplacian bidirectional maximum margin criterion, Pattern Recognit. 42 (11) (2009) 2327–2334.
[34] W.K. Yang, C.Y. Sun, L. Zhang, A multi-manifold discriminant analysis method for image feature extraction, Pattern Recognit. 44 (8) (2011) 1649–1657.
[35] J. Chai, H.W. Liu, Z. Bao, Generalized re-weighting local sampling mean discriminant analysis, Pattern Recognit. 43 (10) (2010) 3422–3432.
[36] D.G. Luenberger, Y.Y. Ye, Linear and Nonlinear Programming, 3nd ed., Springer, New York, USA, 2008.
[37] J. Nocedal, S. Wright, Numerical Optimization, Series in Operations Research and Financial Engineering, 2nd ed., Springer, New York, USA, 2006.
[38] D. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, MA, 1999.
[39] G.R. Terrell, D.W. Scott, Variable kernel density estimation, Ann Stat 20 (3) (1992) 1236–1265.
[40] M. Kristan, A. Leonardis, D. Skočaj, Multivariate online kernel density estimation with Gaussian kernels, Pattern Recognit. 44 (10) (2011) 2630–2642.
[41] A.R. Webb, Statistical Pattern Recognition, 2nd ed, John Wiley and Sons, West Sussex, UK, 2002.
[42] R.A. Amar, D.R. Dooly, S.A. Goldman, Q. Zhang, Multiple-instance learning of real-valued data, in: Proceedings of the 18th International Conference on Machine Learning, 2001, pp. 3–10.
[43] Z.H. Zhou, K. Jiang, M. Li, Multi-instance learning based web mining, Appl. Intell. 22 (2) (2005) 135–147.
[44] Z.H. Zhou, J.M. Xu, On the relation between multi-instance learning and semi-supervised learning, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1167–1174.
[45] G. Csurka, C. Bray, C. Dance, L. Fan, Visual categorization with bags of keypoints, in: ECCV Workshop on Statistical Learning in Computer Vision, 2004, pp. 59–74.

Jing Chai received his B.Eng. and Ph.D. degrees in electronic engineering from the Xidian University, Xi'an, China, in 2005 and 2010, respectively. He is now a lecturer with the College of Information Engineering, Taiyuan University of Technology. His research interests include machine learning, pattern recognition, multiple-instance learning, dimensionality reduction, and metric learning.

Xinghao Ding received his Ph.D. degree from the Hefei University of Technology, Hefei, China, in 2003. He is now a professor with the School of Information Science and Engineering, Xiamen University. His research interests include image processing, machine learning, and compressed sensing.

Hongtao Chen received his Ph.D. degree from the Xidian University, Xi'an, China, in 2012. He is now a lecturer with the College of Information Engineering, Taiyuan University of Technology. His research interests include pattern recognition and medical image processing.

Tingyu Li received her Ph.D. degree from Kobe University, Kobe, Japan, in 2007. She is now a lecturer with the College of Information Engineering, Taiyuan University of Technology. Her research interest is signal processing.