Multi-Instance Clustering with Applications to Multi-Instance Prediction

Min-Ling Zhang and Zhi-Hua Zhou

Abstract. In the setting of multi-instance learning, each object is represented by a *bag* composed of multiple instances instead of by a single instance in traditional learning setting. Previous works in this area only concern multi-instance *prediction* problems where each bag is associated with a binary (classification) or real-valued (regression) label. However, *unsupervised* multi-instance learning where bags are without labels have not been studied. In this paper, the problem of unsupervised multi-instance learning is addressed where a multi-instance clustering algorithm named BAMIC is proposed. Briefly, by regarding bags as atomic data items and using some form of distance metric to measure distances between bags, BAMIC adapts the popular k-MEDOIDS algorithm to partition the unlabeled training bags into k disjoint *groups of bags*. Furthermore, based on the clustering results, a novel multi-instance prediction algorithm named BARTMIP is developed. Firstly, each bag is re-represented by a k-dimensional feature vector, where the value of the i-th feature is set to be the distance between the bag and the medoid of the i-th group. After that, bags are transformed into feature vectors so that common supervised learners are used to learn from the transformed feature vectors each associated with the original bag's label. Extensive experiments show that BAMIC could effectively discover the underlying structure of the data set and BARTMIP works quite well on various kinds of multi-instance prediction problems.

1 Introduction

Multi-instance learning originated from the study of drug activity prediction problem. The goal is to endow learning systems with the ability of predicting that whether a new molecule could be used to make some drug, through analyzing a collection of known molecules. The qualification of the molecule to make some drug is determined by some of its shapes with low energy. The main difficulty lies in that each molecule may have many alternative low-energy shapes while biochemists only know that whether a known molecule is qualified to make some drug instead of knowing that which of its alternative shapes is responsible for the qualification. In order to solve this problem, Dietterich et al. [1] proposed the notion of *multi-instance learning*, where the training set is composed of many *bags* each containing many instances.

Previous works on multi-instance learning only deal with prediction tasks where each bag is as-

sociated with a binary or real-valued label. Dietterich et al. [1] studied the standard multi-instance prediction (StdMIP) problem where a bag is positively labeled if it contains at least one positive instance; otherwise, it is labeled as a negative bag. The task is to learn some concept from the training set for correctly labeling unseen bags. Recently, in addition to the above definition of StdMIP, two kinds of generalized multi-instance prediction (GenMIP) problems are formalized by Weidmann et al. [2] and Scott et al. [3], respectively. In GenMIP problems, a bag is qualified to be positive iff instances in the bag satisfy some sophisticated constraints other than simply having at least one positive instance. Moreover, besides the above multi-instance classification problems with discrete-valued (in particular, binary) outputs, multi-instance regression (MIR) problems with real-valued outputs have also been investigated [4–6].

However, unsupervised multi-instance learning where training bags are without labels have not been thoroughly studied even though it should not be considered less important than multi-instance prediction, i.e. supervised multi-instance learning. There are two main reasons for the necessity of unsupervised multi-instance learning. Firstly, in some cases, it is hard or costly to obtain labels for the bags. For instance, biochemists may easily design various drug molecules each represented by a bag whose instances correspond to the low-energy shapes of the molecule [1], while it is quite costly to conduct biochemical experiments in order to determine the functionality (label) of the molecule (bag). Therefore, unsupervised multi-instance learning may help the biochemists identify similar molecules which may have similar properties so as to facilitate the drug design process. Secondly, it is well-known that unsupervised learning could help find the inherent structure of a data set. Therefore, even for multi-instance prediction tasks where bags are with labels, it is instructive to initially perform unsupervised multi-instance learning. In this way, some useful information about the data set may be acquired which could benefit the following prediction procedure.

Cluster analysis (clustering) is one of the most popular unsupervised learning methods, which is the process of grouping a set of physical or abstract objects into classes of similar objects [49]. A good clustering will produce high quality clusters with high intra-class similarity and low inter-class similarity. It can be used as a stand-alone tool to gain insight into the distribution of data or to serve as a preprocessing step for other tasks. In this paper, the problem of unsupervised multi-instance learning is directly investigated where a multi-instance clustering algorithm named Bamic, i.e. BAg-level Multi-Instance Clustering, is proposed. Concretely, Bamic tries to partition the unlabeled training bags into k disjoint *groups of bags*, where several forms of Hausdorff metric [7] is utilized to measure distances between bags with which the popular k-Medoids algorithm is adapted to fulfill the clustering task. Experimental results show that Bamic could effectively reveal the inherent structure

of the multi-instance data set.

In addition, based on the clustering results of BAMIC, a novel multi-instance prediction algorithm named BARTMIP, i.e. BAg-level Representation Transformation for Multi-Instance Prediction, is also developed. Specifically, BARTMIP performs representation transformation by converting each bag into a corresponding k-dimensional feature vector whose i-th feature equals the distance between the bag and the medoid of the i-th group. Consequently, bags are transformed into feature vectors meaning that the original task of multi-instance prediction is reduced to the task of traditional supervised learning. In this paper, support vector machines are employed to learn from the reduced learning problems. Extensive experiments show that the performance of BARTMIP is highly competitive to those of other multi-instance prediction algorithms on various multi-instance prediction problems, such as STDMIP, MIR and two kinds of GENMIP models[1].

The rest of this paper is organized as follows. Section 2 briefly reviews previous literatures on multi-instance learning. Section 3 presents BAMIC and reports its experimental results on real-world and artificial data sets. Section 4 presents BARTMIP and reports its experimental results on various multi-instance prediction problems. Section 5 discusses the relationships between BARTMIP and other related works. Finally, Section 6 concludes and raises several issues for future work.

2 Literature Review

Multi-instance learning originates from the research of drug activity prediction [1], where the PAC-learnability of this learning framework has been studied by many researchers. Long and Tan [8] showed that if the instances in the bags are independently drawn from product distribution, then the APR (Axis-Parallel Rectangle) proposed by Dietterich et al. [1] is PAC-learnable. Auer et al. [9] showed that if the instances in the bags are not independent then APR learning under the multi-instance learning framework is NP-hard. Moreover, they presented a theoretical algorithm that does not require product distribution, which has been transformed to a practical algorithm named MULTINST [10]. Blum and Kalai [11] described a reduction from PAC-learning under the multi-instance learning framework to PAC-learning with one-sided or two-sided random classification noise. They also presented a theoretical algorithm with smaller sample complexity than that of the algorithm of Auer et al. [9]. Goldman et al. [12] presented an efficient on-line agnostic multi-instance learning algorithm for learning the class of constant-dimension geometric patterns, which tolerated both noise and concept shift. Later, this algorithm was extended so that it could deal with real-valued output [13].

Many practical STDMIP algorithms have been developed during the past years, such as APR algorithms [1], DIVERSE DENSITY [14] and its variants [15,16], multi-instance lazy learning algorithms [17],

multi-instance tree learners [18–20], multi-instance rule inducer [19], multi-instance logistic regression methods [21], multi-instance neural networks [22–26], multi-instance kernel methods [27–29], and multi-instance ensembles [21, 30–33], etc. StdMIP techniques have already been applied to diverse applications including content-based image retrieval [34–38], scene classification [16, 39], stock selection [14], landmark matching [12, 13], computer security [18], subgoal discovery [40], web mining [41], etc.

In the early years of the research of multi-instance learning, most works were on multi-instance classification with discrete-valued outputs. Later, multi-instance regression with real-valued outputs caught the attention of many researchers [4–6].

Besides the StdMIP and MIR problems as introduced above, two kinds of GenMIP models have also been formalized recently. Firstly, Weidmann et al. [2] indicated that through employing different assumptions of how the instances' classifications determine their bag's label, different kinds of multi-instance prediction problems could be defined. Formally, let $\mathcal{X}$ denote the instance space and $\Omega = \{0, 1\}$ denote the set of class labels. A multi-instance concept is a function $\nu_{MI} : 2^{\mathcal{X}} \to \Omega$. In StdMIP, this function is defined as Eq.(1), where $c_I \in \mathcal{C}$ is a specific concept from a concept space $\mathcal{C}$ defined over $\mathcal{X}$, and $X \subseteq \mathcal{X}$ is a set of instances.

$$\nu_{MI}(X) \Leftrightarrow \exists x \in X : c_I(x) \tag{1}$$

Based on this characterization, Weidmann et al. [2] defined three kinds of GenMIP problems, i.e. *presence-based MI*[2], *threshold-based MI*, and *count-based MI*. Presence-based MI is defined in terms of the presence of instances of each concept in a bag, as shown in Eq.(2); threshold-based MI requires a certain number of instances of each concept to be present simultaneously, as defined in Eq.(3); count-based MI requires a maximum as well as a minimum number of instances of a certain concept in a bag, as defined in Eq.(4).

$$\nu_{PB}(X) \Leftrightarrow \forall c_i \in C : \Delta(X, c_i) \geq 1 \tag{2}$$

$$\nu_{TB}(X) \Leftrightarrow \forall c_i \in C : \Delta(X, c_i) \geq t_i \tag{3}$$

$$\nu_{CB}(X) \Leftrightarrow \forall c_i \in C : t_i \leq \Delta(X, c_i) \leq z_i \tag{4}$$

In Eqs.(2) to (4), ν_{PB}, ν_{TB} and ν_{CB} are functions defined on $2^{\mathcal{X}} \to \Omega$, $C \subset \mathcal{C}$ is a given set of concepts, Δ is a counting function $\Delta : 2^{\mathcal{X}} \times \mathcal{C} \to \mathbb{N}$ which counts the number of a given concept in a bag, $t_i \in \mathbb{N}$ and $z_i \in \mathbb{N}$ are respectively the lower and upper threshold for concept c_i. Two algorithms have been proposed for this kind of GenMIP model (GenMIP Model I), i.e. Tlc [2] and Cce [42], both of which could also work well in solving StdMIP problems.

Independently of Weidmann et al. [2], Scott et al. [3] defined another GENMIP model where the target concept is defined by two *sets* of points. Specifically, they defined their concepts by a set of q 'attraction' points $C = \{c_1, \ldots, c_q\}$ and a set of q' 'repulsion' points $\bar{C} = \{\bar{c}_1, \ldots, \bar{c}_{q'}\}$. Then the label for a bag $X = \{x_1, \ldots, x_p\}$ is positive iff there is a subset of r points $C' \subseteq C \cup \bar{C}$ such that each attraction point $c_i \in C'$ is near some point in X and each repulsion point $\bar{c}_j \in C'$ is not near any point in X. In other words, if one boolean attribute a_{c_i} is defined for each attraction point $c_i \in C$ which takes the value of 1 iff there exists some point $x \in X$ near it, and another boolean attribute $\bar{a}_{\bar{c}_j}$ is defined for each repulsion point $\bar{c}_j \in \bar{C}$ which takes the value of 1 iff there is no point from X near it, then the bag's label is an r-of-$(q + q')$ threshold function over these attributes. Several algorithms have been specially designed for this kind of GENMIP model (GENMIP Model II), such as GMIL-1 [3], GMIL-2 [43], $k_\wedge$ [44] and $k_{\min}$ [45]. The same as TLC [2] and CCE [42], these GENMIP algorithms could also be utilized to solve STDMIP problems. In addition, GENMIP Model II has also been applied to applications such as robot vision, content-based image retrieval, binding affinity, and biological sequence analysis [3].

Note that none of these two GENMIP models is more expressive than the other one. For instance, by setting $t_i \geq 2$ $(i \in \{1, \ldots q\})$ for each attraction point, then both the *threshold-based MI* and *count-based MI* models of Weidmann et al. [2] will not be possible to be represented by the GENMIP model of Scott et al. [3]. On the other hand, as indicated by Tao et al. [44], when $r < q + q'$, the representational ability of Scott et al.'s model will go beyond the generalization scope of Weidmann et al.'s. It is also worth noting that these two GENMIP models do overlap with each other under certain circumstances. For instance, by setting $t_i = z_i = 1$ $(i \in \{1, \ldots q\})$ for each attraction point and setting $t_j = z_j = 0$ $(j \in \{1, \ldots, q'\})$ for each repulsion point, then the *count-based MI* model of Weidmann et al. is just the one of Scott et al. with $r = q + q'$.

It is worth mentioning that multi-instance learning has also attracted the attention of the Inductive Logic Programming (ILP) community. De Raedt [46] suggested that multi-instance problems could be regarded as a bias on inductive logic programming, and the multi-instance paradigm could be the key between the propositional and relational representations, being more expressive than the former, and much easier to learn than the latter. Recently, Alphonse and Matwin [47] successfully employed multi-instance learning to help relational learning, where the expressive power of relational representation and the ease of feature selection on propositional representation are gracefully combined. It is also worth noting that although multi-instance learning was proposed initially based on propositional representation, a recent modification developed by McGovern and Jensen [48] allows multi-instance techniques to be used on relational representation. These works confirm that multi-instance learning

can really act as a bridge between propositional and relational learning.

All the works reviewed in this section concern supervised multi-instance learning where bags are associated with labels (discrete-valued or real-valued). In the next section, unsupervised multi-instance learning problem where bags are without labels is addressed by the first multi-instance clustering algorithm BAMIC.

3 Multi-Instance Clustering

3.1 BAMIC

Previous works show that multi-instance representation, i.e. bag of multiple instances, seems a natural and appropriate form for object description in many real-world problems. For instance, in drug activity prediction, due to rotations of internal bonds an drug molecule usually have many low-energy shapes each can be represented by an instance [1]; Another example is scene classification, where an image usually contains multiple regions each of which can be represented as an instance [16,39]; Web mining is a further example, where each of the links in the web page can be regarded as an instance [41]. However, although it is relatively easy to represent a large number of available objects (e.g. images in databases) by bags of instances via pre-defined bag generation process, obtaining labels for these objects will be a rather costly and tedious work. In this case, unsupervised multi-instance learning techniques could properly manifest its capability to summarize some useful information from unlabeled training bags which could be utilized for further analysis.

As introduced in Section 1, clustering is one of the most popular strategies for unsupervised learning. Traditional clustering schemes include partitioning methods (e.g. k-MEDOIDS), hierarchical methods (e.g. AGNES), density-based methods (e.g. DBSCAN), grid-based methods (e.g. STING), model-based methods (e.g. SOM), etc [49]. In this paper, BAMIC adapts the popular k-MEDOIDS algorithm to complete the desired multi-instance clustering task due to its simplicity and efficiency.

Note that at the first glance, multi-instance clustering may be regarded as merely a simple extension of traditional clustering task where the objects to be clustered are now sets of instances instead of single instances. However, the task of clustering multi-instance bags has its own characteristics. Specifically, in order to cluster objects described by sets of instances, the most intuitive strategy is to let the instances contained in one set contribute equally to the clustering process. While for multi-instance clustering, this kind of strategy may not be appropriate since the instances comprising the bag usually exhibit different functionalities. For example, only one or few of the low-energy shapes describing one molecule would be responsible for its qualification to make certain drugs. Similarly, only one or few of the regions describing one image would be useful for certain applications such as scene classification.

[*Groups*, *Medoids*] = BAMIC(U, k, Bag_dist)

Inputs:

U — unlabeled multi-instance training set $\{X_1, X_2, ..., X_N\}$ ($X_i \subseteq \mathcal{X}$)

k — number of clustered groups

Bag_dist — distance metric used to calculate distances between bags, which could take the form of maximal, minimal or average Hausdorff distance in this paper

Outputs:

Groups — clustering outputs $\{G_1, G_2, \ldots, G_k\}$ ($\bigcup_{i=1}^{k} G_i = U$, $G_i \bigcap_{i \neq j} G_j = \emptyset$)

Medoids — Medoids of clustered groups

Process:

(1) Randomly select k training bags as the initial medoids C_j ($1 \leq j \leq k$);

(2) **repeat**

(3) **for** $j \in \{1, 2, \ldots, k\}$ **do**

(4) $G_j = \{C_j\}$;

(5) **for** $i \in \{1, 2, \ldots, N\}$ **do**

(6) $index = \arg\min_{j \in \{1,2,\ldots,k\}} \text{Bag\_dist}(X_i, C_j)$;

(7) $G_{index} = G_{index} \bigcup \{X_i\}$;

(8) **for** $j \in \{1, 2, \ldots, k\}$ **do**

(9) $C_j = \arg\min_{A \in G_j} \left(\sum_{B \in G_j} \text{Bag\_dist}(A, B) / |G_j| \right)$;

(10) **until** {the clustering results do not change};

(11) $Groups = \{G_j | 1 \leq j \leq k\}$; $Medoids = \{C_j | 1 \leq j \leq k\}$;

Figure 1: Pseudo-code describing the BAMIC algorithm.

Therefore, although in multi-instance clustering the labels of the bags are missing, the bags should not be regarded as simple collections of independent instances while the idiosyncrasies and relationships of the instances in the bags should be carefully investigated.

Let $\mathcal{X}$ denote the domain of instances. Suppose the training set U is composed of N unlabeled bags, i.e. $U = \{X_1, X_2, \ldots, X_N\}$ where $X_i \subseteq \mathcal{X}$ is a set of instances. BAMIC clusters all the training bags into k disjoint groups G_i ($1 \leq i \leq k$) each containing a number of training bags, i.e. $\bigcup_{i=1}^{k} G_i = \{X_1, X_2 \ldots, X_N\}$ and $G_i \bigcap_{i \neq j} G_j = \emptyset$. By regarding each bag as an atomic object, the popular k-MEDOIDS algorithm can be easily adapted to fulfill the desired clustering task provided

that some kind of distance function is used to measure the distances between bags. Actually, there have been two types of such distances, i.e. *maximal Hausdorff distance* [7] and *minimal Hausdorff distance* [17]. Formally, given two bags of instances $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$, the maximal and minimal Hausdorff distances are defined as Eq.(5) and Eq.(6) respectively, where $\|a-b\|$ measures the distance between instances a and b which usually takes the form of Euclidean distance.

$$\text{maxH}(A, B) = \max\{\max_{a\in A}\min_{b\in B}\|a-b\|, \max_{b\in B}\min_{a\in A}\|b-a\|\} \tag{5}$$

$$\text{minH}(A, B) = \min_{a\in A, b\in B}\|a-b\| \tag{6}$$

Both maximal and minimal Hausdorff distances have been successfully applied to the StdMIP problems [17] and MIR problems [4]. However, in our preliminary experiments, it was found that neither of these two distances could work well on the GenMIP problems. The reason that maximal Hausdorff distance did not work well may be its sensitivity to outlying points [17], while the reason for the unsatisfactory performance of minimal Hausdorff distance may be that it only considers the distance between the nearest pair of instances in A and B. Therefore, in this paper, another distance called *average Hausdorff distance*[3] is proposed to measure the distance between two bags:

$$\text{aveH}(A, B) = \frac{\sum_{a\in A}\min_{b\in B}\|a-b\| + \sum_{b\in B}\min_{a\in A}\|b-a\|}{|A|+|B|} \tag{7}$$

where $|\cdot|$ measures the cardinality of a set. In words, $\text{aveH}(\cdot,\cdot)$ averages the distances between each instance in one bag and its nearest instance in the other bag. Conceptually speaking, average Hausdorff distance takes more geometric relationships between two bags of instances into consideration than those of maximal and minimal Hausdorff distances.

Fig. 1 illustrates the complete description of Bamic. Based on some specific form of Hausdorff distance (maximal, minimal or average), Bamic partitions the unlabeled training set into k disjoint groups each containing a number of training bags. The medoid of each group is the one which has the minimum average distance to the other bags in the same group.

3.2 Experiments

3.2.1 Experimental Setup

To the best of the authors' knowledge, Bamic is the first work on unsupervised multi-instance learning, and there is no benchmark data set for the evaluation of unsupervised multi-instance learning algorithms. In this paper, several supervised multi-instance data sets are employed instead to test the effectiveness of Bamic. Traditionally, the quality of the clustering results is usually measured by various cluster validation indices [50], where the desired grouping properties such as the labeling

information are not available. However, for the experiments reported in this subsection, the labels associated with the bags are known, which can be utilized to give more realistic estimate of the clustering quality other than using cluster validation indices.

Considering that no other multi-instance clustering methods are available for comparative studies, the performance of BAMIC is evaluated against two traditional clustering methods, i.e. the partitioning method k-MEDOIDS and the hierarchical method AGNES (AGglomerative NESting, also known as the single-linkage clustering) [49].

Let S be a set of N binary labeled bags, i.e. $S = \{(X_1, BL_{X_1}), (X_2, BL_{X_2}), \ldots, (X_N, BL_{X_N})\}$ where $X_i \subseteq \mathcal{X}$ is a set of instances and $BL_{X_i} \in \{0, 1\}$ is the binary label associated with X_i. As shown in Fig. 1, given parameters k (number of clustered groups) and Bag_dist$(\cdot, \cdot)$ (distance metric between bags), BAMIC will partition the set of unlabeled bags $\{X_1, X_2, \ldots, X_N\}$ into k disjoint groups G_j each containing a number of bags. However, as for k-MEDOIDS and AGNES, both of which cluster the data set at the level of instances instead of at the level of bags. In this way, the bag X_i may be broken into several parts each belonging to different group when the clustering process is completed. In order to account for the effect of fractional bags, each instance in the bag X_i is now assigned with a weight $\frac{1}{|X_i|}$. By doing this, instances from small bags will have higher weights than instances from large bags while bags are considered to be of equal importance (i.e. with weight 1). Note that this kind of weighting strategy has been shown to be reasonable for multi-instance learning as in the literatures [2, 20].

For each group G_j $(1 \leq j \leq k)$, let W_j^l $(l \in \{0, 1\})$ denote the sum of weights of those instances in G_j which are coming from bags (may be fractional for k-MEDOIDS and AGNES) with label l. Furthermore, let W_j denote the sum of weights of all the instances in G_j, i.e. $W_j = W_j^0 + W_j^1$. It is easy to see that $\sum_{j=1}^{k} W_j = N$. Based on this, the following two criteria are defined in this paper to assess the quality of the clustering results:

$$avgpurity(\{G_1, G_2, \ldots, G_k\}) = \sum_{j=1}^{k} \frac{W_j}{N} \cdot \frac{\max\{W_j^0, W_j^1\}}{W_j} \tag{8}$$

$$avgentropy(\{G_1, G_2, \ldots, G_k\}) = \sum_{j=1}^{k} \frac{W_j}{N} \cdot \left(\sum_{l \in \{0,1\}} -\frac{W_j^l}{W_j} \log_2 \frac{W_j^l}{W_j} \right) \tag{9}$$

The first criterion $avgpurity(\cdot)$ measures the weighted *average purity* of the clustering results. Here, purity of G_j refers to the ratio of the weights of (fractional) bags belonging to the dominating class in G_j to the total weights of (fractional) bags in G_j, which is then weighted by $\frac{W_j}{N}$. The performance is perfect when $avgpurity(\{G_1, G_2, \ldots, G_k\}) = 1$. The *bigger* the value of this criterion, the *better* the performance of the clustering algorithm; The second criterion $avgentropy(\cdot)$ measures

the *average entropy* of the clustering results, i.e. the information needed to correctly classify all the (fractional) bags in the clustered groups. It resembles the entropy definition used in traditional decision tree building [51]. The performance is perfect when $avgentropy(\{G_1, G_2, \ldots, G_k\}) = 0$. The *smaller* the value of this criterion, the *better* the performance of the clustering algorithm.

Experimental results on several StdMIP and GenMIP data sets are reported in the following two subsections respectively.

3.2.2 STDMIP Data Sets

Musk data is a real-world benchmark test data for StdMIP algorithms, which was generated in the research of drug activity prediction [1]. Here each molecule is regarded as a bag, and its alternative low-energy shapes are regarded as the instances in the bag. A positive bag corresponds to a molecule qualified to make a certain drug, that is, at least one of its low-energy shapes could tightly bind to the target area of some larger protein molecules such as enzymes and cell-surface receptors. A negative bag corresponds to a molecule unqualified to make a certain drug, that is, none of its low-energy shapes could tightly bind to the target area. In order to represent the shapes, a molecule was placed at a standard position and orientation and then a set of 162 rays emanating from the origin was constructed so that the molecule surface was sampled approximately uniformly. There were also four features representing the position of an oxygen atom on the molecular surface. Therefore each instance in the bags was represented by 166 continuous attributes.

There are two data sets, i.e. Musk1 and Musk2, both publicly available at the UCI machine learning repository [52]. Musk1 contains 47 positive bags and 45 negative bags, and the number of instances contained in each bag ranges from 2 to 40. Musk2 contains 39 positive bags and 63 negative bags, and the number of instances contained in each bag ranges from 1 to 1,044. Detailed information on the Musk data is tabulated in Table 1.

Table 1: The Musk data (72 molecules are shared in both data sets).

Data Set	Dim.	Bags			Instances	Instances per Bag		
		Total	Musk	Non-Musk		Min	Max	Ave.
Musk1	166	92	47	45	476	2	40	5.17
Musk2	166	102	39	63	6,598	1	1,044	64.69

Figs. 2 and 3 show how the performance of each clustering algorithm on the Musk data changes as the number of clustered groups increasing in terms of average purity and average entropy respectively. The number of clustered groups ranges from 5 to 90 on Musk1 while from 5 to 100 on Musk2, both with an interval of 5. Bamic-w-minH denotes the version of Bamic when minimal Hausdorff

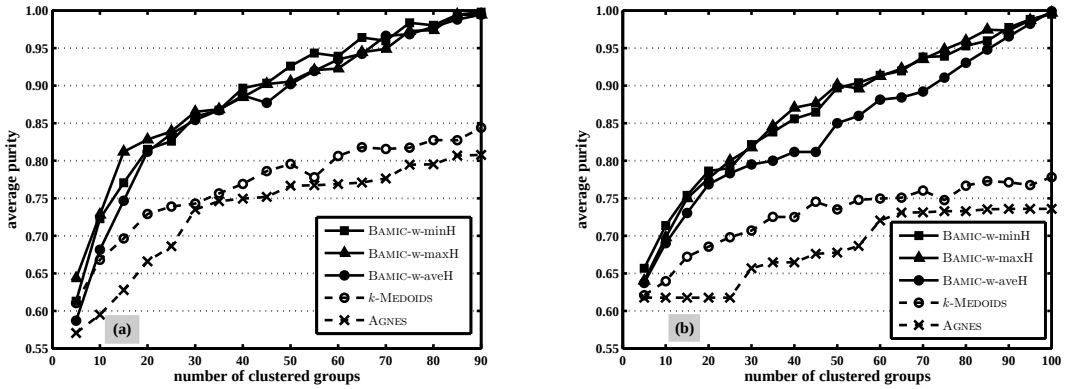

Figure 2: The average purity of the clustering algorithm changes as the number of clustered groups increasing. (a) MUSK1; (b) MUSK2.

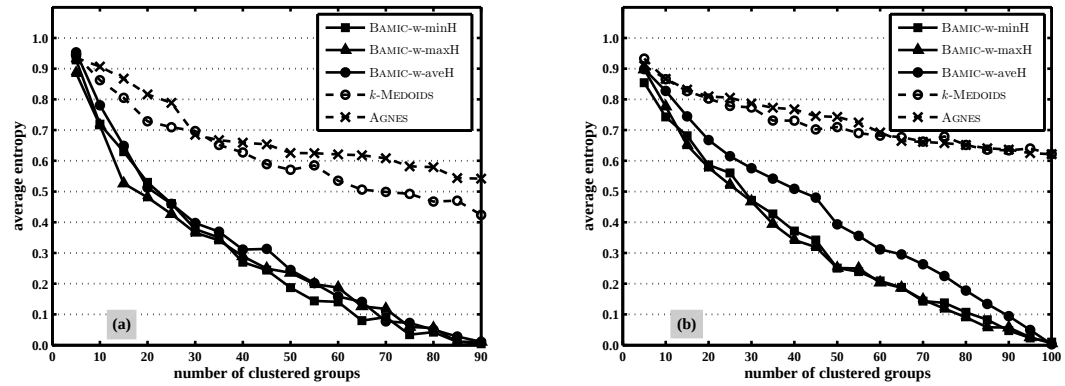

Figure 3: The average entropy of the clustering algorithm changes as the number of clustered groups increasing. (a) MUSK1; (b) MUSK2.

distance is used to measure distance between bags. The same naming rule goes for BAMIC-w-maxH and BAMIC-w-aveH. For each of the five clustering algorithms, when the number of clustered groups is fixed, experiments are repeated for ten times where the averaged evaluation value out of ten runs is depicted as a point in the figure. The concrete experimental results of each clustering algorithm in terms of average purity and average entropy are also reported in Tables 2 and 3 respectively, where the number of clustered groups ranges from 5 to 80 with an interval of 15. The value following '$\pm$' gives the standard deviation and the best result out of the five clustering algorithms on each number of clustered groups is shown in bold face.

Fig. 2 and Table 2 show that in terms of average purity, on both MUSK1 and MUSK2, all the three versions of BAMIC significantly and consistently outperform k-MEDOIDS and AGNES. Furthermore,

Table 2: Experimental results of each clustering algorithm (mean±std) on MUSK1 and MUSK2 with different number of clustered groups in terms of average purity.

DATA SET	CLUSTERING ALGORITHM	NUMBER OF CLUSTERED GROUPS					
		$k = 5$	$k = 20$	$k = 35$	$k = 50$	$k = 65$	$k = 80$
MUSK1	BAMIC-w-minH	.613±.043	.815±.029	.867±.016	**.926±.022**	**.964±.019**	**.980±.014**
	BAMIC-w-maxH	**.644±.038**	**.828±.036**	**.869±.028**	.905±.017	.945±.020	.974±.013
	BAMIC-w-aveH	.587±.016	.812±.050	.867±.027	.902±.026	.942±.015	.978±.013
	k-MEDOIDS	.611±.030	.729±.025	.757±.015	.796±.025	.818±.020	.828±.019
	AGNES	.571±.000	.666±.000	.746±.000	.767±.000	.771±.000	.795±.000
MUSK2	BAMIC-w-minH	**.657±.032**	**.786±.026**	.838±.014	.897±.017	.920±.030	.953±.008
	BAMIC-w-maxH	.649±.020	.777±.022	**.846±.021**	**.901±.022**	**.923±.017**	**.960±.019**
	BAMIC-w-aveH	.637±.013	.769±.012	.800±.012	.850±.037	.884±.020	.930±.018
	k-MEDOIDS	.621±.010	.686±.027	.725±.020	.735±.020	.751±.019	.767±.012
	AGNES	.618±.000	.618±.000	.665±.000	.678±.000	.731±.000	.733±.000

Table 3: Experimental results of each clustering algorithm (mean±std) on MUSK1 and MUSK2 with different number of clustered groups in terms of average entropy.

DATA SET	CLUSTERING ALGORITHM	NUMBER OF CLUSTERED GROUPS					
		$k = 5$	$k = 20$	$k = 35$	$k = 50$	$k = 65$	$k = 80$
MUSK1	BAMIC-w-minH	.928±.054	.530±.049	.350±.034	**.187±.044**	**.080±.038**	**.042±.032**
	BAMIC-w-maxH	**.886±.060**	**.481±.082**	**.341±.067**	.236±.047	.127±.045	.056±.026
	BAMIC-w-aveH	.953±.034	.513±.088	.369±.062	.245±.056	.141±.041	.050±.030
	k-MEDOIDS	.943±.035	.729±.028	.652±.020	.571±.066	.506±.038	.467±.029
	AGNES	.935±.000	.816±.000	.667±.000	.626±.000	.618±.000	.579±.000
MUSK2	BAMIC-w-minH	**.854±.044**	.587±.068	.427±.027	**.251±.032**	.188±.072	.107±.019
	BAMIC-w-maxH	.897±.018	**.578±.053**	**.395±.045**	**.251±.046**	**.187±.038**	**.091±.037**
	BAMIC-w-aveH	.899±.031	.667±.027	.542±.021	.393±.070	.296±.030	.178±.036
	k-MEDOIDS	.933±.016	.803±.040	.731±.023	.710±.029	.677±.027	.652±.028
	AGNES	.910±.000	.810±.000	.773±.000	.743±.000	.665±.000	.651±.000

there is no significant difference among the performance of the three versions of BAMIC on MUSK1, while the performance of BAMIC-w-aveH is slightly inferior to those of BAMIC-w-minH and BAMIC-w-maxH on MUSK2. Fig. 3 and Table 3 reveal the same phenomena as shown above when the performance of clustering algorithm is evaluated in terms of average entropy. The above results indicate that BAMIC could work well on the real-world MUSK data.

3.2.3 GENMIP Data Sets

In addition to the above STDMIP data sets, several GENMIP data sets are employed to further evaluate the performance of each clustering algorithm. Specifically, the *presence-based* MI of GENMIP Model I proposed by Weidmann et al. [2] is used.

In generating presence-based MI data sets, $|C|$ concepts were used. To generate a positive bag,

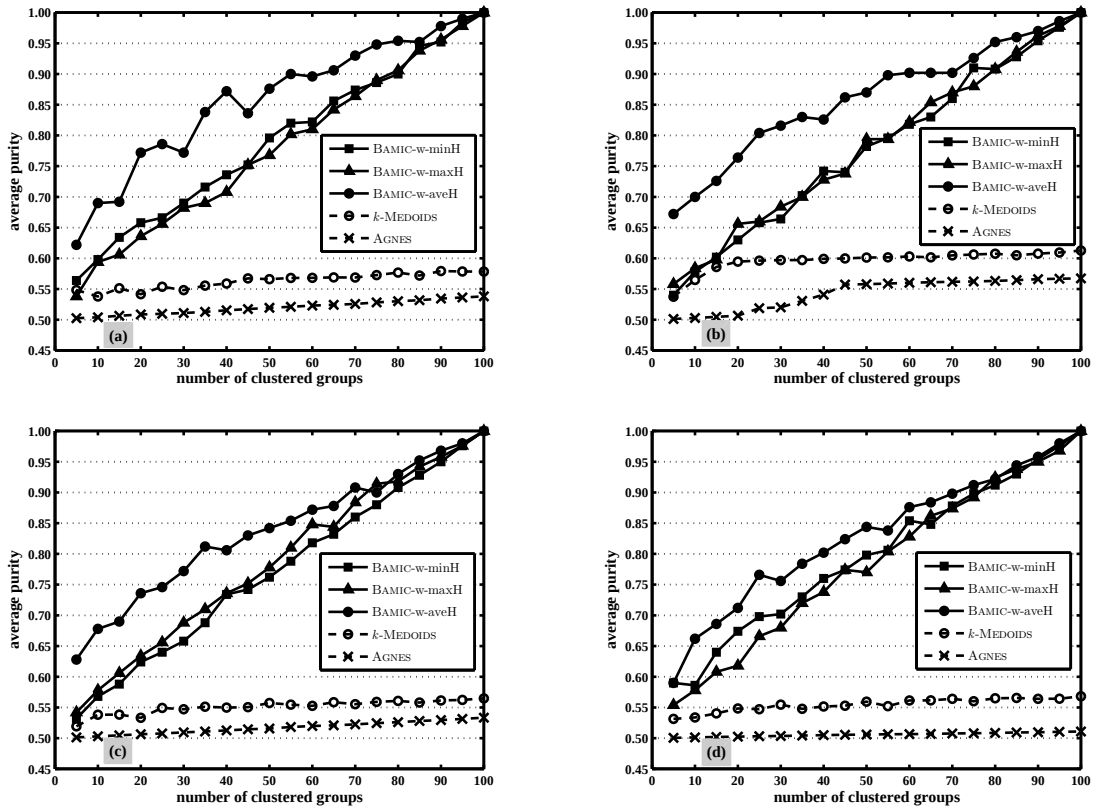

Figure 4: The average purity of the clustering algorithm changes as the number of clustered groups increasing. (a) 2-5-5; (b) 2-10-0; (c) 3-5-5; (d) 3-5-10.

the number of instances in a concept c_i was chosen randomly from $\{1, \ldots, 10\}$ for each concept. The number of random instances was selected with equal probability from $\{10|C|, \ldots, 10|C|+10\}$. To form a negative bag, they replaced all instances of a concept c_i by random instances. Hence the minimal bag size in this data set was $|C| + 10|C|$ and the maximal bag size was $20|C| + 10$. It is obvious that STDMIP problem is just a special case of presence-based MI by using only one underlying concept (i.e. $|C| = 1$). Therefore, learning from presence-based MI data sets would be inevitably more difficult than learning from STDMIP data sets such as the MUSK data.

In this paper, four presence-based MI data sets are used, i.e. 2-5-5, 2-10-0, 3-5-5, 3-5-10. Here the name of the data set '2-5-5' means it was generated with 2 concepts, 5 relevant and 5 irrelevant attributes. The same naming rule goes for the other data sets. In each data set, there are five different training sets containing 50 positive bags and 50 negative bags, and a big test set containing 5,000 positive bags and 5,000 negative bags. When the data set is used to evaluate the performance of a supervised multi-instance algorithm (as shown in Subsection 4.2.4), the average test set accuracy of

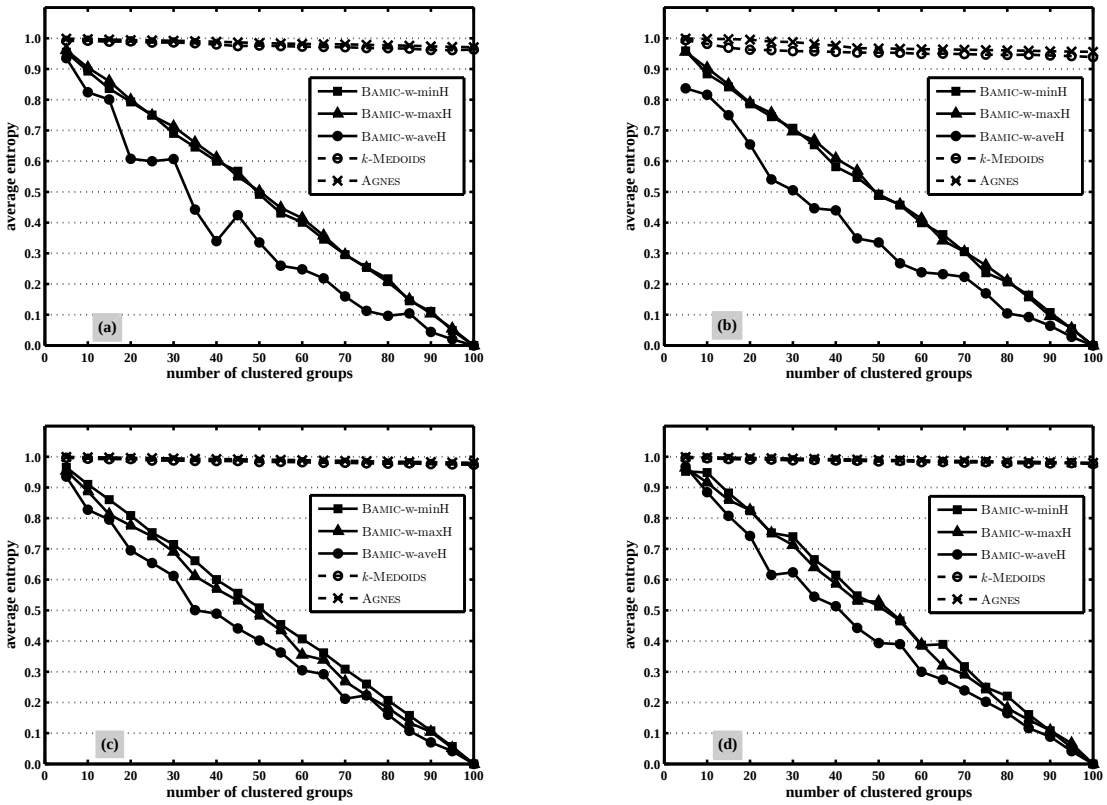

Figure 5: The average entropy of the clustering algorithm changes as the number of clustered groups increasing. (a) 2-5-5; (b) 2-10-0; (c) 3-5-5; (d) 3-5-10.

the classifiers trained on each of these five training sets is recorded as the predictive accuracy of the tested algorithm on that data set. In this subsection, however, only the five training sets contained in each data set are used to evaluated the performance of the unsupervised multi-instance clustering algorithms.

Figs. 4 and 5 show how the performance of each clustering algorithm on the presence-based MI data sets changes as the number of clustered groups increasing in terms of average purity and average entropy respectively. The number of clustered groups ranges from 5 to 100 with an interval of 5 on all data sets. When the number of clustered groups and the clustering algorithm are fixed, the averaged evaluation value out of five training sets is depicted as a point in the figure. The concrete experimental results of each clustering algorithm in terms of average purity and average entropy are also reported in Tables 4 and 5 respectively, where the number of clustered groups ranges from 5 to 80 with an interval of 15. The value following '$\pm$' gives the standard deviation and the best result out of the five clustering algorithms on each number of clustered groups is shown in bold face.

Table 4: Experimental results of each clustering algorithm (mean±std) on presence-based MI data sets with different number of clustered groups in terms of average purity.

Data Set	Clustering Algorithm	Number of Clustered Groups					
		$k = 5$	$k = 20$	$k = 35$	$k = 50$	$k = 65$	$k = 80$
2-5-5	Bamic-w-minH	.564±.021	.658±.042	.716±.027	.796±.024	.856±.023	.900±.007
	Bamic-w-maxH	.538±.019	.636±.013	.690±.012	.768±.016	.842±.008	.906±.006
	Bamic-w-aveH	**.622±.048**	**.772±.042**	**.838±.051**	**.876±.015**	**.906±.020**	**.954±.017**
	k-Medoids	.548±.008	.542±.012	.556±.001	.566±.008	.569±.005	.577±.009
	Agnes	.503±.001	.509±.001	.513±.001	.520±.003	.524±.004	.530±.004
2-10-0	Bamic-w-minH	.540±.021	.630±.017	.702±.049	.782±.028	.830±.017	.908±.008
	Bamic-w-maxH	.558±.026	.656±.025	.700±.014	.794±.020	.854±.018	.908±.005
	Bamic-w-aveH	**.672±.062**	**.764±.051**	**.830±.020**	**.870±.027**	**.902±.025**	**.952±.013**
	k-Medoids	.538±.021	.595±.007	.597±.007	.601±.007	.602±.009	.607±.007
	Agnes	.501±.001	.507±.001	.531±.027	.558±.010	.561±.011	.563±.010
3-5-5	Bamic-w-minH	.532±.008	.624±.017	.688±.013	.762±.008	.832±.013	.908±.008
	Bamic-w-maxH	.542±.015	.634±.018	.710±.022	.778±.016	.844±.021	.918±.015
	Bamic-w-aveH	**.628±.043**	**.736±.011**	**.812±.024**	**.842±.039**	**.878±.023**	**.930±.016**
	k-Medoids	.519±.026	.533±.016	.551±.014	.557±.007	.559±.011	.561±.008
	Agnes	.501±.001	.506±.001	.511±.001	.516±.002	.521±.001	.526±.002
3-5-10	Bamic-w-minH	**.590±.024**	.674±.034	.730±.036	.798±.022	.848±.026	.912±.016
	Bamic-w-maxH	.554±.017	.618±.018	.720±.026	.770±.024	.862±.034	**.924±.027**
	Bamic-w-aveH	**.590±.024**	**.712±.046**	**.784±.049**	**.844±.032**	**.884±.033**	.922±.018
	k-Medoids	.531±.011	.548±.010	.548±.006	.560±.003	.562±.007	.565±.006
	Agnes	.501±.000	.503±.001	.504±.001	.506±.001	.507±.001	.509±.000

As shown by Figs. 4 and 5 together with Tables 4 and 5, on all data sets and evaluation criteria, all the three versions of Bamic significantly and consistently outperform k-Medoids and Agnes. Specifically, both k-Medoids and Agnes totally fail to find the inherent structure of the concerned GenMIP data. Furthermore, on all data sets and evaluation criteria, the performance of Bamic-w-aveH is superior to those of Bamic-w-minH and Bamic-w-maxH, while there is no significant difference between the performance of the latter two versions of Bamic. The above results show that in addition to the StdMIP data as shown in Subsection 3.2.2, Bamic (especially Bamic-w-aveH) could also work well on the more complicated GenMIP data.

4 Applications to Multi-Instance Prediction

4.1 Bartmip

The difficulty of multi-instance prediction (whether StdMIP, MIR or GenMIP) mainly lies in that although the training bags are labeled, the labels of their instances are unknown. Actually, a bag corresponds to a real-world object while the instances correspond to feature vectors describing the object.

Table 5: Experimental results of each clustering algorithm (mean±std) on presence-based MI data sets with different number of clustered groups in terms of average entropy.

Data Set	Clustering Algorithm	Number of Clustered Groups					
		$k = 5$	$k = 20$	$k = 35$	$k = 50$	$k = 65$	$k = 80$
2-5-5	BAMIC-w-minH	.955±.021	.793±.042	.646±.027	.492±.024	.346±.023	.217±.007
	BAMIC-w-maxH	.961±.019	.799±.013	.661±.012	.502±.016	.358±.008	.206±.006
	BAMIC-w-aveH	**.935±.048**	**.608±.042**	**.443±.051**	**.336±.015**	**.219±.020**	**.097±.017**
	k-MEDOIDS	.991±.008	.990±.012	.984±.001	.977±.008	.972±.005	.968±.009
	AGNES	.999±.001	.994±.001	.991±.001	.985±.003	.981±.004	.977±.004
2-10-0	BAMIC-w-minH	.959±.021	.786±.017	.653±.049	.492±.028	.361±.017	.207±.008
	BAMIC-w-maxH	.956±.026	.790±.025	.668±.014	.487±.020	.341±.018	.211±.005
	BAMIC-w-aveH	**.837±.062**	**.655±.051**	**.447±.020**	**.335±.027**	**.232±.025**	**.105±.013**
	k-MEDOIDS	.994±.021	.963±.007	.958±.007	.953±.007	.950±.009	.946±.007
	AGNES	.999±.001	.996±.001	.981±.027	.967±.010	.964±.011	.961±.010
3-5-5	BAMIC-w-minH	.966±.008	.809±.017	.662±.013	.508±.008	.362±.013	.207±.008
	BAMIC-w-maxH	.954±.015	.776±.018	.611±.022	.483±.016	.339±.021	.182±.015
	BAMIC-w-aveH	**.935±.043**	**.695±.011**	**.501±.024**	**.402±.039**	**.292±.023**	**.160±.016**
	k-MEDOIDS	.997±.026	.992±.016	.986±.014	.983±.007	.980±.011	.978±.008
	AGNES	.999±.001	.996±.001	.994±.001	.991±.002	.988±.001	.985±.002
3-5-10	BAMIC-w-minH	**.952±.024**	.824±.034	.665±.036	.514±.022	.389±.026	.221±.016
	BAMIC-w-maxH	.964±.017	.826±.018	.640±.026	.528±.024	.320±.034	.182±.027
	BAMIC-w-aveH	.968±.024	**.742±.046**	**.545±.049**	**.394±.032**	**.274±.033**	**.165±.018**
	k-MEDOIDS	.996±.011	.991±.010	.990±.006	.985±.003	.983±.007	.980±.006
	AGNES	.999±.000	.996±.001	.993±.001	.990±.001	.987±.001	.984±.000

In contrast to typical machine learning settings where an object is represented by one feature vector, in multi-instance learning an object is represented by a set of feature vectors. Therefore, common supervised machine learning methods can hardly be applied. In fact, Dietterich et al. [1] showed that learning algorithms ignoring the characteristics of multi-instance problems, such as popular decision trees and neural networks, could not work well in this scenario[4].

Until now, there are two distinct strategies to deal with the representation difficulty encountered in multi-instance prediction [42]. The most straightforward strategy is to modify supervised learning algorithms to meet the representation of feature vector sets. Actually, Zhou and Zhang [30] showed that supervised learning algorithms could be adapted to multi-instance prediction as long as their focuses are shifted from the discrimination on the instances to the discrimination on the bags. Most current multi-instance prediction algorithms, especially those for solving STDMIP and MIR problems, can be viewed as going along this way. On the other hand, the opposite strategy, i.e. transforming the representation to meet the requirement of common supervised learning algorithms, can also be considered. In fact, several multi-instance prediction algorithms, especially those for solving GENMIP problems, have been proposed along this way.

In this paper, with the help of BAMIC, a new solution to the problem of multi-instance prediction named BARTMIP is proposed along the second strategy. In detail, training bags are firstly clustered into k disjoint *groups of bags* using BAMIC as introduced in Section 3. Intuitively, since clustering could help find the inherent structure of a data set, the clustered k groups might implicitly encode some information on the spacial distribution of different bags. Therefore, BARTMIP then tries to re-represent the bags based on the clustering results. Specifically, each bag is re-represented by a k-dimensional feature vector whose i-th feature corresponds to the distance between the bag and the medoid of the i-th group. When all the bags are transformed into the k-dimensional feature vectors, common supervised learners could be used to train on the generated feature vectors to distinguish the bags.

When an unseen bag is given for prediction, BARTMIP firstly re-represents it by querying the clustering results, then feeds the generated feature vector to the trained predictor to obtain the predicted label. This means the clustering results, at least the medoids of each clustered groups, should be stored for prediction so that the unseen bag can be transformed into the corresponding feature vector through measuring its distance to different group medoids. However, although BARTMIP is not so efficient as multi-instance prediction algorithms that do not query on training bags in prediction, such as RELIC [18], it is more efficient than algorithms that require storing and querying all the training bags in prediction, such as CITATION-KNN and BAYESIAN-KNN [17]. As will be shown in the next subsection, a prominent advantage of BARTMIP is that it can be easily applied to MIR problems by simply replacing the binary classifier built on the transformed feature vectors with a regression estimator, and applied to GENMIP problems without any modification, while most current multi-instance prediction algorithms can not.

Let D be the set of labeled training bags, i.e. $D = \{(X_1, L_{X_1}), (X_2, L_{X_2}), \ldots, (X_N, L_{X_N})\}$ where $X_i \subseteq \mathcal{X}$ is a set of instances and L_{X_i} is the label (binary or real-valued) associated with X_i. The complete description of BARTMIP is illustrated in Fig. 6. Based on BAMIC, steps (1) to (2) cluster the training set into k disjoint groups[5]. After that, steps (3) to (8) firstly convert each training bag into one feature vector associated with the bag's label, and then use the given learning algorithm to train a predictor from the transformed feature vectors. Note that many algorithms can be used to implement the learning algorithm. Finally, steps (9) to (11) output the label for the test bag by feeding the generated feature vector to the learned predictor.

Note that although BARTMIP is designed along one existing strategy, i.e. performing representation transformation to meet the requirement of traditional supervised learning algorithms, it has its own characteristics. Firstly, for all the other multi-instance prediction algorithms based on representation

$Label$ = BARTMIP(D, k, Bag_dist, $TBag$, $Learner$)

Inputs:

D — labeled multi-instance training set $\{(X_1, L_{X_1}), (X_2, L_{X_2}), ..., (X_N, L_{X_N})\}$ ($X_i \subseteq \mathcal{X},\ L_{X_i} \in \{0, 1\}$ or $[0, 1]$)

k — number of clustered groups

Bag_dist — distance metric used to calculate distances between bags, which could take the form of maximal, minimal or average Hausdorff distance in this paper

$TBag$ — test bag ($TBag \subseteq \mathcal{X}$)

$Learner$ — common supervised learner for the transformed feature vectors

Outputs:

$Label$ — predicted label for the test bag ($Label \in \{0, 1\}$ or $[0, 1]$)

Process:

(1) Set $U = \{X_1, X_2, \ldots, X_N\}$;

(2) $[\{G_1, G_2, \ldots, G_k\}, \{C_1, C_2, \ldots, C_k\}]$ = BAMIC(U, k, Bag_dist);

(3) Set $Tr = \emptyset$;

(4) **for** $i \in \{1, 2, \ldots, N\}$ **do**

(5) **for** $j \in \{1, 2, \ldots, k\}$ **do**

(6) $y_j^i = \text{Bag\_dist}(X_i, C_j)$;

(7) $Tr = Tr \bigcup \{(< y_1^i, y_2^i, \ldots, y_k^i >, L_{X_i})\}$;

(8) Predictor($\cdot$) = $Learner(Tr)$;

(9) **for** $j \in \{1, 2, \ldots, k\}$ **do**

(10) $y_j = \text{Bag\_dist}(TBag, C_j)$;

(11) $Label$ = Predictor($< y_1, y_2, \ldots, y_k >$);

Figure 6: Pseudo-code describing the BARTMIP algorithm.

transformation [2,3,42–45], the bags are transformed into corresponding feature vectors by structuring the input space at the level of *instances*. While for BARTMIP, through unsupervised clustering of training bags, representation transformation is performed by structuring the input space at the level of *bags*, which seems more pertinent to the original representation of multi-instance learning. On the other hand, most existing multi-instance prediction algorithms have only been applied to one or two kinds of multi-instance prediction problems. In this paper, BARTMIP is utilized to solve various kinds

of multi-instance prediction problems, including StdMIP, MIR and two kinds of GenMIP problems.

4.2 Experiments

4.2.1 Experimental Setup

As shown in Fig. 6, several parameters should be determined in advance to make Bartmip be a concrete prediction algorithm. In this subsection, support vector machines (SVM)[6], backpropagation neural networks (BNN) and K-nearest neighbors (KNN) are used respectively as the base learners to learn from the transformed feature vectors. Considering that after representation transformation, different multi-instance learning problems will give rise to different common supervised learning problems, thus the parameters used by each base learner are coarsely tuned for different data sets to yield comparable performance. For SVM, gaussian kernels are used with γ-parameter set to be 0.25 and C-parameter set to be 3. For BNN, the number of hidden neurons is set to be the number of input neurons. For KNN, the number neighbors considered is set to be 3. The following experiments (as shown in Table 6) on the Musk data, whose detailed characteristics have already been introduced in Subsection 3.2.2, are performed to determine other parameters of Bartmip.

Ten times of *leave-one-bag-out* tests are performed on both Musk1 and Musk2, where the parameter k (i.e. the number of clustered groups) ranges from 20% to 100% of $|D|$ (i.e. the number of training bags) with an interval of 20%. Bartmip-SVM-minH denotes the version of Bartmip where SVM is employed to learn from the transformed feature vectors and minimal Hausdorff distance is used to measure distance between bags. The same naming rule goes for the other algorithms. The value following '$\pm$' gives the standard deviation and the best result out of three different distance measures on each number of clustered groups is shown in bold face.

Table 6 shows that the number of clustered groups does not significantly affect the performance of each prediction algorithm. When the base learner is fixed, in most cases better performance is achieved when average Hausdorff distance is used as the bag distance metric (except Bartmip-KNN on Musk1 and Bartmip-BNN on Musk2). Furthermore, when the bag distance metric is fixed, in all cases better performance is achieved when SVM is used as the base learner. Therefore, in the rest of this paper, the reported results of Bartmip are all obtained with μ set to be the moderate value of 60%, where SVM is employed to learn from the transformed feature vectors and average Hausdorff distance is utilized to measure distance between bags.

Experimental results of Bartmip on all kinds of multi-instance prediction formalizations, including StdMIP, MIR and GenMIP, are reported sequentially in the following subsections. Note that the performance of different algorithms have been reported on different data sets. For each data set used

Table 6: Experimental results of BARTMIP (mean±std) on the MUSK data with different base learners, different number of clustered groups and different bag distance metrics.

DATA SET	PREDICTION ALGORITHM	Number of Clustered Groups (= μ×NUMBER OF TRAINING BAGS)				
		$\mu = 20\%$	$\mu = 40\%$	$\mu = 60\%$	$\mu = 80\%$	$\mu = 100\%$
MUSK1	BARTMIP-SVM-minH	91.1±0.9	91.2±1.5	90.4±0.9	88.7±1.7	**90.2±0.0**
	BARTMIP-SVM-maxH	85.3±1.6	85.1±2.4	84.4±1.3	82.0±2.5	81.5±0.0
	BARTMIP-SVM-aveH	**92.6±1.0**	**94.7±1.0**	**94.1±1.0**	**92.0±1.4**	**90.2±0.0**
	BARTMIP-BNN-minH	77.2±3.7	76.5±2.7	80.8±1.7	76.5±1.3	79.7±0.0
	BARTMIP-BNN-maxH	74.6±1.7	77.9±1.3	79.0±4.1	76.8±3.8	77.5±0.0
	BARTMIP-BNN-aveH	**79.7±4.1**	**78.9±4.4**	**84.4±1.7**	**78.6±3.5**	**80.8±0.0**
	BARTMIP-KNN-minH	**87.4±1.8**	**88.9±2.0**	86.7±1.8	**87.3±1.7**	**89.1±0.0**
	BARTMIP-KNN-maxH	81.1±2.7	80.0±2.4	80.8±2.1	81.0±2.2	81.5±0.0
	BARTMIP-KNN-aveH	85.7±3.2	86.3±2.8	**86.9±1.5**	**87.3±1.3**	88.0±0.0
MUSK2	BARTMIP-SVM-minH	86.7±2.4	88.0±1.1	86.6±1.9	82.6±1.5	82.4±0.0
	BARTMIP-SVM-maxH	85.9±1.9	88.2±1.7	87.6±1.2	85.4±1.3	83.3±0.0
	BARTMIP-SVM-aveH	**90.7±1.7**	**91.2±1.0**	**89.8±1.2**	**87.3±1.4**	**88.2±0.0**
	BARTMIP-BNN-minH	74.2±2.8	76.5±2.6	76.8±3.7	78.1±3.2	74.8±0.0
	BARTMIP-BNN-maxH	**76.5±2.9**	**79.4±2.5**	**82.0±3.2**	**79.1±4.4**	**80.4±0.0**
	BARTMIP-BNN-aveH	69.9±2.3	73.2±3.8	75.5±1.0	75.8±3.3	73.9±0.0
	BARTMIP-KNN-minH	83.0±1.5	**84.6±0.6**	83.0±1.5	82.0±0.6	81.4±0.0
	BARTMIP-KNN-maxH	80.7±3.2	82.7±0.6	82.0±1.5	81.1±2.0	81.4±0.0
	BARTMIP-KNN-aveH	**84.6±3.2**	**84.6±1.5**	**85.3±1.7**	**86.6±2.0**	**85.3±0.0**

in this paper, the experimental results of other algorithms ever reported on that data set are included for comparative studies. Furthermore, in Subsection 4.2.3, time complexities of BARTMIP and some other algorithms are also compared. All the algorithms studied in this paper take the propositional (i.e. attribute-value) representation language.

4.2.2 STDMIP Data Sets

Experimental results on the most widely used STDMIP data sets, i.e. the MUSK data, are shown in this subsection. The performance of BARTMIP (γ=0.25, C=3) are presented in the first part of Table 7. The first part also contains the results of those algorithms designed for GENMIP Model I (i.e. TLC [2] and CCE [42]) and GENMIP Model II (i.e. $k_\wedge$ [44] and $k_{\min}$ [45]), which can also work on the MUSK data. The second part contains the results of other learning algorithms designed for the STDMIP problems. Note that Zhou and Zhang [30] have shown that ensembles of multi-instance learners could achieve better results than single multi-instance learners. However, considering that BARTMIP is a single multi-instance learner, the performance of ensembles of multi-instance learners are not included in Table 7 for fair comparison. The empirical results shown in Table 7 have been obtained by multiple runs of tenfold cross-validation (10CV), by 1,000 runs of randomly leaving out

Table 7: Comparison of the performance (%correct ± std) of each multi-instance prediction algorithm on the MUSK data.

PREDICTION ALGORITHM	MUSK1	EVAL.	PREDICTION ALGORITHM	MUSK2	EVAL.
BARTMIP	94.1±1.0	LOO	BARTMIP	89.8±1.2	LOO
CCE [42]	92.4	LOO	CCE [42]	87.3	LOO
TLC [2]	88.7±1.6	10CV	TLC [2]	83.1±3.2	10CV
$k_\wedge$ [44]	82.4	10CV	$k_\wedge$ [44]	77.3	10CV
$k_{\min}$ [45]	82.4	10CV	$k_{\min}$ [45]	77.3	10CV
ID APR [1]	92.4	10CV	DD-SVM [16]	91.3	10CV
CITATION-KNN [17]	92.4	LOO	ID APR [1]	89.2	10CV
BOOSTING [32]	92.0	10CV	MITI [20]	88.2	10CV
GFS KDE APR [1]	91.3	10CV	MI SVM [27]	88.0±1.0	10OUT
BAG UNIT-BASED RBF [25]	90.3	10CV	RBF-MIP [26]	88.0±3.5	LOO
GFS EC APR [1]	90.2	10CV	RELIC [18]	87.3	10CV
BAYESIAN-KNN [17]	90.2	LOO	BOOSTING [32]	87.1	10CV
RBF-MIP [26]	90.2±2.6	LOO	BAG UNIT-BASED RBF [25]	86.6	10CV
DIVERSE DENSITY [14]	88.9	10CV	CITATION-KNN [17]	86.3	LOO
MI-NN [22]	88.0	N/A	EM-DD [15]	84.9	10CV
RIPPER-MI [19]	88.0	N/A	MI-SVM [28]	84.3	10CV
BP-MIP-PCA [24]	88.0	LOO	MILOGREGARITH [21]	84.1±1.3	10CV
MIBOOSTING [21]	87.9±2.0	10CV	MULTINST [10]	84.0±3.7	10CV
mi-SVM [28]	87.4	10CV	MIBOOSTING [21]	84.0±1.3	10CV
MILOGREGARITH [21]	86.7±1.8	10CV	mi-SVM [28]	83.6	10CV
MI SVM [27]	86.4±1.1	10OUT	BP-MIP-PCA [24]	83.3	LOO
MILOGREGGEOM [21]	85.9±2.2	10CV	DIVERSE DENSITY [14]	82.5	10CV
BP-MIP-DD [24]	85.9	LOO	BAYESIAN-KNN [17]	82.4	LOO
DD-SVM [16]	85.8	10CV	MILOGREGGEOM [21]	82.3±1.2	10CV
EM-DD [15]	84.8	10CV	MI-NN [22]	82.0	N/A
MITI [20]	83.7	10CV	BP-MIP [23]	80.4	LOO
RELIC [18]	83.7	10CV	BP-MIP-DD [24]	80.4	LOO
BP-MIP [23]	83.7	LOO	GFS KDE APR [1]	80.4	10CV
MI-SVM [28]	77.9	10CV	RIPPER-MI [19]	77.0	N/A
MULTINST [10]	76.7±4.3	10CV	GFS EC APR [1]	75.7	10CV

10 bags and training on the remaining ones (10OUT), or by leave-one-bag-out test (LOO)[7].

Table 7 shows that, among the algorithms shown in the first part, BARTMIP is comparable to CCE but apparently better than TLC, $k_\wedge$ and $k_{\min}$ on both MUSK1 and MUSK2. Compared with those algorithms shown in the second part, BARTMIP is comparable to ID APR, CITATION-KNN and BOOSTING but apparently better than the other algorithms on MUSK1, while it is comparable to DD-SVM, ID-APR, MITI, MI SVM and RBF-MIP but apparently better than the other algorithms on MUSK2. These observations show that BARTMIP works quite well on the STDMIP problems.

4.2.3 MIR Data Sets

In 2001, Amar et al. [4] presented a method for creating artificial MIR data, which was further studied by Dooly et al. [6] in 2002. This method generates an artificial receptor (target point) t at first. Then, artificial molecules (denoted as X_i) with 3-5 instances per bag are generated, with each feature value considered as the distance from the origin to the molecular surface when all molecules are in the same orientation. Let X_{ij} denote the j-th instance of X_i and X_{ijk} denote the value of the k-th feature of X_{ij}. Each feature has a scale factor $s_k \in [0, 1]$ to represent its importance in the binding process. The binding energy E_{X_i} of X_i to t is calculated as:

$$E_{X_i} = \max_{X_{ij} \in X_i} \left\{ \sum_{k=1}^{n} s_k \cdot V(t_k - X_{ijk}) \right\} \tag{10}$$

where n is the number of features and

$$V(u) = 4\epsilon \left(\left(\frac{\sigma}{u} \right)^{12} - \left(\frac{\sigma}{u} \right)^{6} \right) \tag{11}$$

is called the *Lennard-Jones potential* [53] for intermolecular interactions with parameters ϵ (the depth of the potential well) and σ (the distance at which $V(u) = 0$), and u is the distance between the bag instances and the target point on a specific feature (they assumed that $u > \sigma$ which means $V(u) < 0$). Based on this, the real-valued label $RL_{X_i} \in [0, 1]$ for molecule X_i is the ratio of E_{X_i} to the maximum possible binding energy E_{max}:

$$RL_{X_i} = \frac{E_{X_i}}{E_{max}} = \max_{X_{ij} \in X_i} \left\{ \frac{\sum_{k=1}^{n} s_k \cdot V(t_k - X_{ijk})}{E_{max}} \right\} \tag{12}$$

where $E_{max} = -\epsilon \cdot \sum_{k=1}^{n} s_k$. Thresholding at 1/2 converts RL_{X_i} to a binary label BL_{X_i}.

The artificial data sets are named as LJ-$r.f.s$ where r is the number of relevant features, f is the number of features, and s is the number of different scale factors used for the relevant features. To partially mimic the MUSK data, some data sets only use labels that are not near 1/2 (indicated by the 'S' suffix) and all scale factors for the relevant features are randomly selected between [0.9, 1]. Note that these data sets are mainly designed for multi-instance regression, but they can also be used for multi-instance classification through rounding the real-valued label to 0 or 1.

Leave-one-bag-out test is performed on four artificial data sets, i.e. LJ-80.166.1, LJ-80.166.1-S, LJ-160.166.1 and LJ-160.166.1-S. Each data set contains 92 bags. The performance of BARTMIP (γ=1, C=1) is compared with those of BP-MIP, RBF-MIP, DIVERSE DENSITY and CITATION-KNN, where the performance of BP-MIP, RBF-MIP and those of DIVERSE DENSITY and CITATION-KNN are reported in the literatures [23], [26] and [4] respectively. Table 8 and Table 9 gives the predictive error and squared loss of each comparing algorithm.

Table 8: Comparison of the predictive error of each multi-instance prediction algorithm on the MIR data sets.

Data Set	Bartmip	Bp-mip	Rbf-mip	Diverse Density	Citation-knn
LJ-80.166.1	3.3	18.5	6.6	N/A	8.6
LJ-80.166.1-S	0.0	18.5	18.5	53.3	0.0
LJ-160.166.1	5.4	16.3	5.1	23.9	4.3
LJ-160.166.1-S	0.0	18.5	1.1	0.0	0.0

Table 9: Comparison of the squared loss of each multi-instance prediction algorithm on the MIR data sets.

Data Set	Bartmip	Bp-mip	Rbf-mip	Diverse Density	Citation-knn
LJ-80.166.1	0.0102	0.0487	0.0167	N/A	0.0109
LJ-80.166.1-S	0.0075	0.0752	0.0448	0.1116	0.0025
LJ-160.166.1	0.0051	0.0398	0.0108	0.0852	0.0014
LJ-160.166.1-S	0.0039	0.0731	0.0075	0.0052	0.0022

Table 10: Comparison of the training time (mean±std, measured in seconds) of each multi-instance prediction algorithm on the StdMIP and MIR data sets.

Data Set	Bartmip	Bp-mip	Rbf-mip	Diverse Density	Citation-knn
Musk1	1.5±0.4	359.5±93.3	1.9±0.6	1183.1±225.7	0.0±0.0
Musk2	88.2±5.5	209.1±14.2	78.1±4.2	44817.7±680.9	0.0±0.0
LJ-80.166.1	0.6±0.1	215.4±11.2	1.1±0.2	492.3±44.4	0.0±0.0
LJ-80.166.1-S	0.6±0.1	154.0±44.3	1.1±0.1	526.4±84.1	0.0±0.0
LJ-160.166.1	0.6±0.1	182.5±27.0	1.1±0.1	403.8±57.5	0.0±0.0
LJ-160.166.1-S	0.8±0.3	214.9±152.5	1.6±0.7	688.7±73.1	0.0±0.0

Table 11: Comparison of the testing time (mean±std, measured in seconds) of each multi-instance prediction algorithm on the StdMIP and MIR data sets.

Data Set	Bartmip	Bp-mip	Rbf-mip	Diverse Density	Citation-knn
Musk1	0.008±0.005	0.006±0.006	0.002±0.005	0.036±0.037	1.428±0.238
Musk2	0.344±0.044	0.591±0.058	0.546±0.086	0.102±0.011	88.555±0.564
LJ-80.166.1	0.006±0.006	0.008±0.005	0.004±0.006	0.008±0.005	0.431±0.007
LJ-80.166.1-S	0.006±0.006	0.008±0.005	0.004±0.003	0.004±0.005	0.431±0.007
LJ-160.166.1	0.006±0.006	0.004±0.006	0.004±0.003	0.008±0.005	0.427±0.008
LJ-160.166.1-S	0.008±0.008	0.010±0.007	0.006±0.003	0.012±0.005	0.741±0.012

In terms of predictive error (as shown in Table 8), Bartmip is comparable to Rbf-mip and Citation-knn but significantly outperforms Bp-mip and Diverse Density on all data sets. In terms of squared loss (as shown in Table 9), Bartmip is comparable to Citation-knn on LJ-80.166.1

and is worse than CITATION-KNN on LJ-80.166.1-S, LJ-160.166.1 and LJ-160.166.1-S, but significantly outperforms BP-MIP, RBF-MIP and DIVERSE DENSITY on all data sets. These results indicate that, in addition to STDMIP problems, BARTMIP could also work well on MIR problems.

In addition, Table 10 and Table 11 report respectively the training time and testing time of those algorithms studied in this subsection on both STDMIP and MIR (thresholding real-valued label into binary one) data sets. Experiments are conducted on an HP server equipped with 4G RAM and four Intel Xeron^{TM} CPUs each running at 2.80GHz. As shown in Table 10, the training complexity of BARTMIP is comparable to that of RBF-MIP while much smaller than that of BP-MIP and DIVERSE DENSITY (As a lazy learning style algorithm, CITATION-KNN takes no training phase and thus leads to the null training complexity). As shown in Table 11, the testing complexity of BARTMIP is comparable to that of BP-MIP, RBF-MIP and DIVERSE DENSITY while much smaller than that of CITATION-KNN. The above results show that BARTMIP is quite efficient in both training phase and testing phase compared to other multi-instance prediction algorithms.

4.2.4 GENMIP Model I

Weidmann et al. [2] designed some methods for artificially generating data sets of GENMIP Model I. In each data set they generated, there are five different training sets containing 50 positive bags and 50 negative bags, and a big test set containing 5,000 positive bags and 5,000 negative bags. The average test set accuracy of the classifiers trained on each of these five training sets is recorded as the predictive accuracy of a tested learning algorithm on that data set. Note that in this subsection, the results of MI SVM and the TLC algorithms are from the literature [2], while the results of CCE are from the literature [42].

The way on how to generate presence-based MI data sets has already been introduced in Subsection 3.2.3. Experimental results on this kind of data sets are shown in Table 12, here the name of the data set '2-10-5' means this data set was generated with 2 concepts, 10 relevant and 5 irrelevant attributes.

In generating threshold-based MI data sets, Weidmann et al. [2] also used $|C|$ concepts. They chose thresholds $t_1 = 4$ and $t_2 = 2$ for data sets with $|C| = 2$, and $t_1 = 2$, $t_2 = 7$ and $t_3 = 5$ for data sets with $|C| = 3$. For positive bags, the number of instances of concept c_i was chosen randomly from $\{t_i, \ldots, 10\}$. To form a negative bag, they replaced at least $(\Delta(X, c_i) - t_i + 1)$ instances of a concept c_i in a positive bag X by random instances. The minimal bag size in this data set is $\sum_i t_i + 10|C|$, the maximal size is $20|C| + 10$. In this paper, four threshold-based MI data sets are used. The results are shown in Table 13, here the name of the data set '42-10-5' means this data set has at least 4 instances of the first concept and 2 instances of the second concept in a positive bag, with 10 relevant and 5

Table 12: Comparison of the performance (%correct ± std) of each multi-instance prediction algorithm on presence-based MI data sets.

Data Set	Bartmip	Bartmip-AS	MI Svm	Cce	Tlc	Tlc-AS
2-10-5	92.53±0.81	96.51±1.89	82.00±1.53	88.95±0.31	85.18±10.07	96.67±1.58
2-10-10	87.30±3.17	97.07±1.27	80.74±0.79	87.67±0.79	86.63± 8.69	94.45±4.54
3-10-0	96.72±1.08	94.58±1.66	84.39±1.25	88.39±0.97	95.68± 3.78	94.73±2.91
3-10-5	84.66±1.01	91.96±5.14	84.27±1.44	84.93±0.90	78.07± 0.91	87.41±6.24

Table 13: Comparison of the performance (%correct ± std) of each multi-instance prediction algorithm on threshold-based MI data sets.

Data Set	Bartmip	Bartmip-AS	MI Svm	Cce	Tlc	Tlc-AS
42-10-5	90.05±2.28	90.38±0.91	85.36±0.92	87.64±1.34	88.65±10.12	96.58±1.91
42-10-10	87.84±0.50	88.71±0.91	83.93±0.36	88.04±0.38	84.59± 8.08	95.89±2.05
275-5-10	77.87±1.90	84.68±1.35	82.73±0.85	79.59±0.26	86.42± 5.39	93.75±6.63
275-10-5	88.22±0.34	88.36±1.55	87.05±0.75	84.39±0.65	86.92± 6.56	90.44±4.63

Table 14: Comparison of the performance (%correct ± std) of each multi-instance prediction algorithm on count-based MI data sets.

Data Set	Bartmip	Bartmip-AS	MI Svm	Cce	Tlc	Tlc-AS
42-10-5	84.06±1.49	84.17±1.76	54.62±0.50	58.30±1.18	57.80±8.55	92.78± 2.45
42-10-10	78.56±3.09	81.28±2.72	55.59±2.81	57.79±1.72	51.05±1.60	65.10±20.35
275-5-10	52.58±0.63	56.55±1.89	52.34±0.50	52.74±0.81	50.33±0.72	56.94±11.64
275-10-5	76.45±0.92	77.01±2.02	54.50±1.81	56.64±1.94	54.11±4.79	83.50±17.88

irrelevant attributes. The same naming rule goes for the other data sets.

In generating count-based MI data sets, Weidmann et al. [2] still used $|C|$ concepts. They used the same value for both threshold t_i and z_i. Hence, the number of instances of concept c_i is exactly z_i in a positive bag. They set $z_1 = 4$ and $z_2 = 2$ for data sets with $|C| = 2$, and $z_1 = 2$, $z_2 = 7$ and $z_3 = 5$ for data sets with $|C| = 3$. A negative bag could be created by either increasing or decreasing the required number z_i of instances for a particular c_i. They chose a new number from $\{0, \ldots, z_i - 1\} \bigcup \{z_i + 1, \ldots, 10\}$ with equal probability. If this number was less than z_i, they replaced instances of concept c_i by random instances; if it was greater, they replaced random instances by instances of concept c_i. The minimal bag size in this data set is $\sum_i z_i + 10|C|$, and the maximal possible bag size is $\sum_i z_i + 10|C| + 10$. In this paper, four count-based MI data sets are used. The results are shown in Table 14, here the name of the data set '42-10-5' means this data set requires exactly 4 instances of the first concept and 2 instances of the second concept in a positive bag, with 10 relevant and 5 irrelevant attributes. The same naming rule goes for the other data sets.

For these GenMIP data sets, the γ-parameter and the C-parameter are coarsely tuned to be 10 and 5 respectively for the Gaussian kernel SVM. Considering that attributed selection (AS) has been applied to refine the Tlc algorithm [2], for fair comparison, this strategy has also been adopted to improve the performance of Bartmip. In detail, backward attribute selection [54] was used. It starts with all attributes and in each round eliminating one attribute where the remaining attribute subset will mostly improve the performance of the algorithm. The effectiveness of a given subset of attributes is evaluated by tenfold cross-validation on the training set using this subset, and the attribute selection process terminates when eliminating attributes will no longer improve the algorithm's performance or there is only one attribute remaining. Of course, this method is computationally expensive and increases the runtime of Bartmip considerably.

On presence-based MI data sets (as shown in Table 12), the performance of Bartmip is better or at least comparable to those of MI Svm, Cce and Tlc, while the performance of Bartmip with AS (i.e. Bartmip-AS) is also better or at least comparable to that of Tlc with AS (i.e. Tlc-AS) when the strategy of attribute selection is incorporated. On threshold-based MI data sets (as shown in Table 13), the performance of Bartmip is better or at least comparable to those of MI Svm, Cce and Tlc except on '275-5-10', while the performance of Bartmip with AS is not so good as that of Tlc with AS as attribute selection has not significantly enhanced the generalization ability of Bartmip. On count-based MI data sets (as shown in Table 14), the performance of Bartmip is significantly better than those of MI Svm, Cce and Tlc on '42-10-5', '42-10-10' and '275-10-5' except on '275-5-10' where all the algorithms fail due to the very high ratio of irrelevant attributes, while Bartmip with AS performs better on '42-10-10' but worse on '42-10-5' and '275-10-5' than Tlc with AS. It is worth noting that, although the Tlc algorithms (with and without AS) achieve high average accuracies on almost all these data sets of GenMIP Model I, the variance of their results are higher or much higher than those of Bartmip (with and without AS), MI Svm and Cce in most cases. The experimental results show that Bartmip could also work well on GenMIP Model I as well as on StdMIP and MIR problems, even though it is not a specially designed learner of GenMIP Model I such as Tlc and Cce.

4.2.5 GENMIP Model II

In presenting GenMIP Model II, Scott et al. [3] also designed an algorithm (referred to here as Gmil-1) for their model and successfully applied it to various areas such as robot vision, content-based image retrieval, biological sequence analysis and molecular binding. However, this algorithm requires explicitly enumerating all axis-parallel boxes in the discrete space $\{0, \ldots, s\}^d$ (where d is the

dimensionality and s is the number of discrete values in each dimension), where each box is assigned with a boolean attribute. This inevitably causes the time complexity of GMIL-1 to be exponential in d and prohibits its applicability to high dimensional learning problems (e.g. the MUSK data as shown in Subsection 4.2.2). Later on, Tao and Scott [43] developed some heuristics to significantly speed up GMIL-1 in certain circumstances, yielding the algorithm named GMIL-2. Unfortunately, GMIL-2 still has exponential time complexity and cannot scale well to higher dimensions.

In order to overcome the above problem of exponential time complexity, Tao et al. [44] reformulated GMIL-1 using a kernel that can be used with a support vector machine to efficiently learn geometric multi-instance concepts (i.e. an r-of-$(q+q')$ threshold function as shown in Section 2). They showed that computing such a kernel on two bags X_1 and X_2 is equivalent to count the number of axis-parallel boxes in a discrete, bounded space that contain at least one point from each of X_1 and X_2. They proved that the counting problem is #P-complete and gave a fully polynomial randomized approximation scheme (FPRAS) for it. Later, this kernel is further generalized along the line of Weidmann et al.'s count-based MI [45]. In addition to the widely used MUSK data, they also tested their kernels (i.e. $k_\wedge$ [44] and $k_{\min}$ [45]) on content-based image retrieval data [3], protein superfamily identification data [55] and multi-site drug binding affinity data [3]. In this paper, two of these data sets, i.e. the protein superfamily identification data and multi-site drug binding affinity data are used to further evaluate the effectiveness of BARTMIP[8]. The γ-parameter and the C-parameter are coarsely tuned to be 1 and 10 respectively for the Gaussian kernel SVM, the results of other comparing algorithms are all from the literatures [44] and [45].

The protein superfamily identification data[9] deals with the problem of identifying new Trx-fold (Thioredoxin-fold) proteins based on each protein's primary sequence. However, the low conservation of primary sequence in the Trx-fold superfamily makes conventional modeling methods such as Hidden Markov Models (HMMs) difficult to use. Therefore, Wang et al. [55] proposed to using multi-instance learning techniques for identification of new Trx-fold proteins. They mapped each protein's primary sequence to a bag in the following way: Firstly, they found the primary sequence motif (typically CxxC) in each (positive and negative) sequence that is known to exist in all Trx-fold proteins. Secondly, they extracted a window of size 204 around it (20 residues upstream, 180 downstream, which is a region known to contain the entire Trx-fold) and aligned these windows around the motif. Finally, they mapped all sequences to 8-dimensional profiles based on the numeric properties of Kim et al. [56] and used them as inputs to the multi-instance learning algorithms.

The resulting data set contains 193 bags out of which 180 bags (20 positive and 160 negative) have been studied by Wang et al. [55] and Tao et al. [44, 45]. In this paper, the same experimental setup is

Table 15: Performance of each comparing algorithm on the protein superfamily identification data in terms of predictive error, false positive rate and false negative rate.

Evaluation Criterion	Bartmip	$k_\wedge$	$k_{\min}$	Gmil-1	Gmil-2	Emdd	DD
Predictive Error	0.235	0.218	0.215	N/A	0.250	0.365	0.664
False Positive Rate	0.235	0.218	0.215	N/A	0.250	0.365	0.668
False Negative Rate	0.244	0.169	0.144	N/A	0.250	0.360	0.125

Table 16: Predictive error of each comparing algorithm on the multi-site drug binding affinity data.

Data Set	Bartmip	$k_\wedge$	$k_{\min}$	Gmil-1	Gmil-2	Emdd	DD
5-dim	0.190	0.205	N/A	0.212	0.218	0.191	0.196
10-dim	0.178	0.175	N/A	N/A	N/A	0.223	0.216
20-dim	0.195	0.207	N/A	N/A	N/A	0.268	0.255

used to evaluate the performance of Bartmip on this data set. In detail, 20-fold cross-validation on 20 positive bags and 8-fold cross-validation on 160 negative bags are performed. In each round, Bartmip is trained on 19 positive proteins plus one of 8 sets of negative proteins, and then tested on the held-out positive protein plus the remaining 7 sets of negative proteins. The above process is repeated for each positive protein and each of the 8 sets of negative proteins. The average predictive error, false positive rate and false negative rate of Bartmip over 160 (20×8) times of runs are compared with those of other multi-instance prediction algorithms as shown in Table 15 (Diverse Density is abbreviated as DD).

The second data studied in this subsection, i.e. the multi-site drug binding affinity data, is a generalization of the synthetic data of Amar et al. [4] and Dooly et al. [6] which has been introduced in Subsection 3.2.3. The purpose of generalization is to reflect the notion of 'antagonist' drugs [1] where a molecule must bind at multiple sites to be labeled positive. Adopting the same notation as used in Subsection 3.2.3, in the generalized case, there is a set of target points T ('subtargets') instead of a single target point. A bag X_i is qualified to be positive iff each target point $t \in T$ must bind to some instance in X_i. In other words, similar to Eq.(12), the label of X_i is determined by thresholding the following generalized real-valued label GRL_{X_i} at 1/2:

$$GRL_{X_i} = \min_{t \in T} \left\{ \max_{X_{ij} \in X_i} \left\{ \frac{\sum_{k=1}^{n} s_k \cdot V(t_k - X_{ijk})}{E_{max}} \right\} \right\} \tag{13}$$

Using the above generalized data generator, Scott et al. [3] build ten 5-dimensional data sets (200 training bags and 200 testing bags) each with 4 subtargets and tested their algorithm Gmil-1 on them. Tao et al. [44] also generated data with dimension 10 and 20, each with 5 subtargets, to further evaluate how well their algorithm $k_\wedge$ handles higher-dimensional data. As with the 5-dimensional data, ten

sets were generated, each containing 200 training bags and 200 testing bags. The average predictive errors of BARTMIP on these data sets (with dimensionality of 5, 10 and 20) are compared with those of other multi-instance learning algorithms as shown Table 16 (DIVERSE DENSITY is abbreviated as DD).

On the protein superfamily identification data (as shown in Table 15), in terms of all evaluation criteria, the performance of BARTMIP is worse than those of $k_\wedge$ and $k_{\min}$, but significantly better than those of GMIL-2 and EMDD. Furthermore, DIVERSE DENSITY significantly outperforms all the other comparing algorithms in terms of false negative rate. On the multi-site drug binding affinity data (as shown in Table 16), the performance of BARTMIP is comparable, and in most cases superior to the those of $k_\wedge$, GMIL-1, GMIL-2, EMDD and DIVERSE DENSITY. These experimental results show that BARTMIP could also work well on GENMIP Model II as well as on STDMIP and MIR problems, even though it is not a specially designed learner of GENMIP Model II such as GMIL-1, GMIL-2, $k_\wedge$ and $k_{\min}$.

5 Discussion

As mentioned in Subsection 4.1, BARTMIP solves multi-instance prediction problems by transforming the representation to meet the requirement of traditional supervised learning algorithms. Actually, several algorithms, especially those for GENMIP problems, such as TLC [2], CCE [42], GMIL-1 [3], GMIL-2 [43], $k_\wedge$ [44] and $k_{\min}$ [45] have also been designed along this way. In this section, the relationships between BARTMIP and those algorithms are discussed in more detail.

Weidmann et al. [2] proposed TLC to tackle GENMIP Model I problems, which constructs a *meta-instance* for each bag and then passes the meta-instance and the class label of the corresponding bag to a common classifier. TLC uses a standard decision tree for imposing a structure on the instance space, which is trained on the set of all instances contained in all bags where the instances are labeled with their bag's class label, so that a meta-instance is generated for each bag. However, simply labeling the instances with their bag's class label for decision tree building may be misleading. For instance, in STDMIP problems (special case of presence-based MI problems with $|C| = 1$), not all the instances in a positive bag should also be labeled positive. In contrast to TLC, BARTMIP performs *unsupervised* clustering at the level of bags for representation transformation, where the label of each bag is not necessarily shared by its instances.

Scott et al. [3] proposed GMIL-1 to tackle GENMIP Model II problems, which operates in a discretized feature space $\mathcal{X} = \{0, \ldots, s\}^d$ (as shown in Subsection 4.2.5). GMIL-1 works by enumerating the set of all possible axis-parallel boxes $B_\mathcal{X}$ (including degenerate ones) in $\mathcal{X}$ (so that

$|B_{\mathcal{X}}| = \left(\binom{s+1}{2} + (s+1)\right)^d = \binom{s+2}{2}^d$), and then creating two boolean attributes a_b and $\bar{a}_b$ ($\bar{a}_b = 1 - a_b$) for each box $b \in B_{\mathcal{X}}$. Given a bag $X = (x_1, \ldots, x_p)$, the algorithm set $a_b = 1$ if some point from X lies in b and $a_b = 0$ otherwise. Using the above remapping of each bag to $2|B_{\mathcal{X}}|$ boolean attributes, the target concept defined by some C, $\bar{C}$ and r (as shown in Section 2) can be represented by an r-of-$(q + q')$ threshold function over the attributes a_{c_i} for $c_i \in C$ and $\bar{a}_{\bar{c}_j}$ for $\bar{c}_j \in \bar{C}$. As shown in the literature [3], such a threshold function can be easily learned by WINNOW [57] while making only $O(r(q+q')d \log s)$ mistakes in an online learning setting. Unfortunately, the time complexity of GMIL-1 is exponential in both $\log s$ and d. Although Tao and Scott [43] have developed some heuristics (GMIL-2) to significantly improve the computation efficiency of GMIL-1 in certain situations, GMIL-2 still has exponential time complexity which limits its application to problems with higher dimensions. In contrast to GMIL-1 and GMIL-2, BARTMIP could scale quite well to learning problems with higher dimensions, such as those problems shown in Subsections 4.2.2 and 4.2.3.

For those algorithms mentioned above, i.e. TLC [2], GMIL-1 [3] and GMIL-2 [43], together with $k_\wedge$ [44] and $k_{\min}$ [45] (kernel reformulations of GMIL-1), bags are transformed into corresponding feature vectors in a *supervised* way. That is, the *boolean* label of each bag is essential for the process of representation transformation. While for BARTMIP, this process is accomplished in an unsupervised way by *unsupervised* clustering at the level of bags. One advantage of unsupervised clustering is that BARTMIP could be easily adapted to solve MIR problems with *real-valued* outputs, which is much more difficult for TLC, GMIL-1, GMIL-2, $k_\wedge$ and $k_{\min}$ to do so. Another advantage of unsupervised clustering is that additional unlabeled bags, if available, could be incorporated with the labeled training bags in the clustering process. For instance, in the problem of content-based image retrieval using multi-instance learning techniques [35,36], only a few images (bags) will be labeled by the users while plenty of unlabeled images are available in the database which could be combined with the labeled ones to get more reliable clustering results.

CCE [42] also performs representation transformation in an unsupervised way. Firstly, all the instances contained in all the bags are collected, and the popular clustering algorithm k-MEANS [58] is employed to cluster the instances into d groups. Secondly, each bag is transformed into a d-dimensional feature vector, where if the bag has some instances being clustered to the i-th group, then the value of the i-th feature is set to 1 and 0 otherwise. They also use support vector machines to build a classifier on the transformed feature vectors. Finally, considering that there is no criterion available for judging which kind of clustering result is the best for re-representation of the bags, the above procedures are repeated with different values of d where an ensemble of m classifiers are produced and their outputs are combined by *majority voting* when making predictions for unseen bags. Although in this way CCE

could utilize the power of ensemble learning [59] to achieve strong generalization ability, experimental results reported in Subsection 4.2.4 show that its generalization ability is still worse than that of BARTMIP, which is only a single multi-instance learner. The superiority of BARTMIP may attribute to the clustering results obtained by BAMIC which successfully reveal the underlying distribution of training bags.

As already mentioned several times, a unique characteristic of BARTMIP is that it performs representation transformation by structuring the input space at the level of *bags* instead of at the level of *instances*. Specifically, TLC [2] structures the input space by building a decision tree from the set of all instances contained in all bags by sharing the bag's class label to its component instances; GMIL-1 [3], GMIL-2 [43], $k_\wedge$ [44] and $k_{\min}$ [45] structures the input space by enumerating all the axis-parallel boxes at instance level and then constructing boolean attributes from those boxes; CCE [42] structures the input space by dividing the instances into many disjoint regions using the popular k-MEANS clustering algorithm. However, BARTMIP structures the input space at the level of bags by clustering the training bags into several disjoint *groups of bags*. We suppose that this kind of representation transformation strategy is more pertinent to the original representation of multi-instance learning, i.e. each object represented by a bag (a set of feature vectors) instead of one feature vector.

6 Conclusion

In this paper, the problem of unsupervised multi-instance learning is addressed where a novel multi-instance clustering algorithm named BAMIC is proposed. BAMIC seeks to cluster training bags into a number of disjoint groups of bags, where the popular k-MEDOIDS algorithm is adapted by using some form of distance metric to measure distance between bags. Based on the information provided by the clustering results of BAMIC, a novel multi-instance prediction algorithm named BARTMIP is proposed. BARTMIP re-represents each bag by a feature vector whose feature values are set to be the distances between the bag and the medoids of clustered groups. Therefore, the bags are transformed into feature vectors so that common supervised learners (support vector machines in this paper) are used to learn from the induced data set. Extensive experiments show that BAMIC could effectively reveal the underlying structure of multi-instance data sets while BARTMIP is highly comparable to other multi-instance prediction algorithms on a wide range of multi-instance prediction problems, including STDMIP, MIR and two kinds of GENMIP models.

BAMIC is the first work toward unsupervised multi-instance learning, it is obvious that investigating other kinds of unsupervised multi-instance learning methods to yield better learning ability is an important and interesting issue worth further study.

For BAMIC, there will exist some clusters containing only one bag when the number of clustered groups is large. Investigating some appropriate methods to eliminate these 'trivial' clusters is an important issue for future work. Further evaluating the performance of BARTMIP on more available multi-instance data sets will be another interesting future work.

In addition to several well-established algorithms [2,3,42–45], the success of BARTMIP further shows the feasibility of solving multi-instance prediction problems through representation transformation. Actually, there are many possible ways for this purpose. For examples, referring to Fig. 6, each bag can be transformed into a feature vector whose i-th feature equals the average distance of it to all the bags in G_i instead of the distance of it to the medoid of G_i. Furthermore, besides support vector machines, other supervised learning methods such as neural networks can also be used as the classifier training algorithms. Exploring other schemes for modifying the representation of multi-instance prediction problems is also an interesting issue for future work.

Acknowledgement

The authors wish to thank N. Weidmann for the artificial data sets of GENMIP Model I, T. Osugi for the code of $k_\wedge$, Q. Tao for the multi-site drug binding affinity data, and S. D. Scott and S. A. Goldman for their kind help.

References

[1] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 39(1-2):31–71, 1997.

[2] N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problem. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Lecture Notes in Artificial Intelligence 2837*, pages 468–579. Springer, Berlin, 2003.

[3] S. Scott, J. Zhang, and J. Brown. On generalized multiple-instance learning. Technical Report UNL-CSE-2003-5, Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE, 2003.

[4] R. A. Amar, D. R. Dooly, S. A. Goldman, and Q. Zhang. Multiple-instance learning of real-valued data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 3–10, Williamstown, MA, 2001.

[5] S. Ray and D. Page. Multiple instance regression. In *Proceedings of the 18th International Conference on Machine Learning*, pages 425–432, Williamstown, MA, 2001.

[6] D. R. Dooly, Q. Zhang, S. A. Goldman, and R. A. Amar. Multiple-instance learning of real-valued data. *Journal of Machine Learning Research*, 3(Dec):651–678, 2002.

[7] G. A. Edgar. *Measure, Topology, and Fractal Geometry, 3rd print.* Springer-Verlag, Berlin, 1995.

[8] P. M. Long and L. Tan. PAC learning axis-aligned rectangles with respect to product distribution from multiple-instance examples. *Machine Learning*, 20(1):7–21, 1998.

[9] P. Auer, P. M. Long, and A. Srinivasan. Approximating hyper-rectangles: learning and pseudo-random sets. *Journal of Computer and System Sciences*, 57(3):376–388, 1998.

[10] P. Auer. On learning from multi-instance examples: empirical evaluation of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, pages 21–29, Nashville, TN, 1997.

[11] A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30(1):23–29, 1998.

[12] S. A. Goldman, S. S. Kwek, and S. D. Scott. Agnostic learning of geometric patterns. *Journal of Computer and System Sciences*, 62(1):123–151, 2001.

[13] S. A. Goldman and S. D. Scott. Multiple-instance learning of real-valued geometric patterns. *Annals of Mathematics and Artificial Intelligence*, 39(3):259–290, 2003.

[14] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 570–576. MIT Press, Cambridge, MA, 1998.

[15] Q. Zhang and S. A. Goldman. EM-DD: an improved multiple-instance learning technique. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1073–1080. MIT Press, Cambridge, MA, 2002.

[16] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5(Aug):913–939, 2004.

[17] J. Wang and J.-D. Zucker. Solving the multiple-instance problem: a lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1125, San Francisco, CA, 2000.

[18] G. Ruffo. *Learning single and multiple decision trees for security applications*. PhD thesis, Department of Computer Science, University of Turin, Italy, 2000.

[19] Y. Chevaleyre and J.-D. Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem. In E. Stroulia and S. Matwin, editors, *Lecture Notes in Artificial Intelligence 2056*, pages 204–214. Springer, Berlin, 2001.

[20] H. Blockeel, D. Page, and A. Srinivasan. Multi-instance tree learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 57–64, Bonn, Germany, 2005.

[21] X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In H. Dai, R. Srikant, and C.-Q. Zhang, editors, *Lecture Notes in Computer Science 3056*, pages 272–281. Springer, Berlin, 2004.

[22] J. Ramon and L. De Raedt. Multi instance neural networks. In *Proceedings of ICML-2000, Workshop on Attribute-Value and Relational Learning*, San Francisco, CA, 2000.

[23] Z.-H. Zhou and M.-L. Zhang. Neural networks for multi-instance learning. Technical report, AI Lab, Computer Science & Technology Department, Nanjing University, Nanjing, China, 2002.

[24] M.-L. Zhang and Z.-H. Zhou. Improve multi-instance neural networks through feature selection. *Neural Processing Letters*, 19(1):1–10, 2004.

[25] A. Bouchachia. Multiple instance learning with radial basis function neural networks. In N. R. Pal, N. Kasabov, and R. K. Mudi, editors, *Lecture Notes in Computer Science 3316*, pages 440–445. Springer, Berlin, 2004.

[26] M.-L. Zhang and Z.-H. Zhou. Adapting RBF neural networks to multi-instance learning. *Neural Processing Letters*, 23(1):1–26, 2006.

[27] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186, Sydney, Australia, 2002.

[28] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, Cambridge, MA, 2003.

[29] P.-M. Cheung and J. T. Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23th International Conference on Machine Learning*, Pittsburgh, PE, 2006, to appear.

[30] Z.-H. Zhou and M.-L. Zhang. Ensembles of multi-instance learners. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Lecture Notes in Artificial Intelligence 2837*, pages 492–502. Springer, Berlin, 2003.

[31] S. Andrews and T. Hofmann. Multiple-instance learning via disjunctive programming boosting. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[32] P. Auer and R. Ortner. A boosting approach to multiple instance learning. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Lecture Notes in Artificial Intelligence 3201*, pages 63–74. Springer, Berlin, 2004.

[33] P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006.

[34] X. Huang, S.-C. Chen, M.-L. Shyu, and C. Zhang. Mining high-level user concepts with multiple instance learning and relevance feedback for content-based image retrieval. In O. R. Zaïane, S. J. Simoff, and C. Djeraba, editors, *Lecture Notes in Artificial Intelligence 2797*, pages 50–67. Springer, Berlin, 2002.

[35] C. Yang and T. Lozano-Pérez. Image database retrieval with multiple-instance learning techniques. In *Proceedings of the 16th International Conference on Data Engineering*, pages 233–243, San Diego, CA, 2000.

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

[37] Z.-H. Zhou, M.-L. Zhang, and K.-J. Chen. A novel bag generator for image database retrieval with multi-instance learning techniques. In *Proceedings of the 15th IEEE Intenational Conference on Tools with Artificial Intelligence*, pages 565–569, Sacramento, CA, 2003.

[38] R. Rahmani and S. A. Goldman. Missl: multiple-instance semi-supervised learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 705–712, Pittsburgh, PE, 2006.

[39] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 341–349, Madison, WI, 1998.

[40] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, pages 361–368, Williamstown, MA, 2001.

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

[42] Z.-H. Zhou and M.-L. Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems*, 11(2):155–170, 2007.

[43] Q. Tao and S. Scott. A faster algorithm for generalized multiple-instance learning. In *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference*, pages 550–555, Miami Beach, FL, 2004.

[44] Q. Tao, S. Scott, N. V. Vinodchandran, and T. T. Osugi. Svm-based generalized multiple-instance learning via approximate box counting. In *Proceedings of the 21st International Conference on Machine Learning*, pages 799–806, Banff, Canada, 2004.

[45] Q. Tao, S. Scott, N. V. Vinodchandran, T. T. Osugi, and B. Mueller. An extended kernel for generalized multiple-instance learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 272–277, Boca Raton, FL, 2004.

[46] L. De Raedt. Attribute-value learning versus inductive logic programming: the missing links. In D. Page, editor, *Lecture Notes in Artificial Intelligence 1446*, pages 1–8. Springer, Berlin, 1998.

[47] É. Alphonse and S. Matwin. Filtering multi-instance problems to reduce dimensionality in relational learning. *Journal of Intelligent Information Systems*, 22(1):23–40, 2004.

[48] A. McGovern and D. Jensen. Identifying predictive structures in relational data using multiple instance learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 528–535, Washington, DC, 2003.

[49] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, 2001.

[50] S. Günter and H. Bunke. Validation indices for graph clustering. *Pattern Recognition Letters*, 24(8):1107–1113, 2003.

[51] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[52] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/˜mlearn/mlrepository.html]. Technical report, Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[53] R. S. Berry, S. A. Rice, and J. Ross. *Physical Chemistry, Chapter 10 (Intermolecular Forces)*. John Wiley & Sons, New York, 1980.

[54] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Nowell, MA, 1998.

[55] C. Wang, S. D. Scott, J. Zhang, Q. Tao, D. Fomenko, and V. Gladyshev. A study in modeling low-conservation protein superfamilies. Technical Report UNL-CSE-2004-0003, Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE, 2004.

[56] J. Kim, E. N. Moriyama, C. G. Warr, P. J. Clyne, and J. R. Carlson. Identification of novel multi-transmembrane proteins from genomic databases using quasi-periodic structural properties. *Bioinformatics*, 16(9):767–775, 2000.

[57] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using winnow. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 147–156, Santa Cruz, CA, 1991.

[58] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[59] T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Lecture Notes in Computer Science 1867*, pages 1–15. Springer, Berlin, 2000.

[60] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[61] S. Ray and M. Craven. Supervised versus multiple instance learning: an empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 697–704, Bonn, Germany, 2005.

[62] X. Tan, S. Chen, Z.-H. Zhou, and J. Liu. Learning non-metric partial similarity based on maximal margin criterion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, NY, 2006.

Affiliation of author:

The authors are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China. Corresponding author: Z.-H. Zhou (E-mail: zhouzh@nju.edu.cn, Tel.: +86-25-8368-6268).

Footnotes:

1. In this paper, the generalized multi-instance prediction model proposed by Weidmann et al. [2] is denoted as GENMIP Model I; Accordingly, the other generalized multi-instance prediction model proposed by Scott et al. [3] is denoted as GENMIP Model II.
2. Here multi-instance is abbreviated as MI.
3. Actually, $\text{aveH}(\cdot,\cdot)$ doesn't satisfy the *triangle inequality* and thus can not be regarded as a metric distance [62]. For instance, let $A = \{1\}$, $B = \{2\}$ and $C = \{1, 2\}$, then $\text{aveH}(A, C) + \text{aveH}(C, B) = \frac{2}{3} < 1 = \text{aveH}(A, C)$. However, considering that $\text{minH}(\cdot,\cdot)$ is neither a metric distance ($\text{minH}(A, C) + \text{minH}(C, B) = 0 < 1 = \text{minH}(A, C)$) but is still called as *minimal Hausdorff distance* in the literature [17]. Following the same naming style, $\text{aveH}(\cdot,\cdot)$ is called as *average Hausdorff distance* in this paper.
4. Recently, Ray and Craven [61] empirically studied the relationship between traditional supervised learning and multi-instance learning. They considered four multi-instance algorithms and their supervised counterparts by simply sharing the bag's label to all the instances in the bag. Experiments on 12 multi-instance data sets show that, in most cases (8 out of 12 data sets), the best performance is obtained with the multi-instance algorithm rather than the traditional supervised learner.
5. In our experiments reported in the next subsection, the clustering process converges within 5 rounds in most cases.
6. Specifically, the Matlab® version of LIBSVM [60] is used, where the Matlab code is available at http://www.ece.osu.edu/~maj/osu_svm/ or http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
7. As pointed out by Andrews et al. [28], the EM-DD algorithm [15] seems to use the test data to select the optimal solution obtained from multiple runs of the algorithm. Thus, the experimental results of EM-DD shown in Table 7 are those given by Andrews et al. [28].
8. Unfortunately, due to intellectual property restriction, the content-based image retrieval data [3] used by Tao et al. [44, 45] (from Corel image Suite and www.webshots.com) are not publicly available.
9. Data set publicly available at http://cse.unl.edu/~qtao/datasets/mil_dataset__Trx_protein.html.