mi-DS: Multiple-Instance Learning Algorithm

Dat T. Nguyen, Cao D. Nguyen, Rosalyn Hargraves, Lukasz A. Kurgan, *Member, IEEE*, and Krzysztof J. Cios, *Senior Member, IEEE*

***Abstract*—Multiple-instance learning (MIL) is a supervised learning technique that addresses the problem of classifying bags of instances instead of single instances. In this paper, we introduce a rule-based MIL algorithm, called mi-DS, and compare it with 21 existing MIL algorithms on 26 commonly used data sets. The results show that mi-DS performs on par with or better than several well-known algorithms and generates models characterized by balanced values of precision and recall. Importantly, the introduced method provides a framework that can be used for converting other rule-based algorithms into MIL algorithms.**

***Index Terms*—Multiple-instance learning (MIL), rule-based algorithms, supervised learning.**

I. Introduction

THE MULTIPLE-INSTANCE learning (MIL) problem was introduced by Dietterich *et al.* [13]. It is concerned with classifying bags of instances instead of single instances. A bag is labeled as positive if at least one of its instances is positive and as negative when all of its instances are negative. Several scenarios exist: A positive bag can include one or more true positive instances, the number of true positive instances in a positive bag may be greater or smaller than the number of false positive instances, etc. An example illustrating MIL is a problem in which a locksmith tries to decide whether a given keychain is useful: It is useful (positive) if at least one of its keys opens a door; otherwise, it is useless (negative) [13]. Many MIL methods were developed and used in applications such as image retrieval and annotation [11], [30], [38], failure prediction [27], and protein–ligand docking [28]. Key MIL algorithms are briefly reviewed next.

Several MIL methods are based on probabilistic models. Multiple-Instance Diverse Density (MI-DD) algorithm attempts to find a concept point in the feature space that is close to at least one instance from every positive bag but far away from instances in the negative bags. The optimal concept point has maximum diversity density, which is "a measure of how many different positive bags have instances near that point, and how far the negative instances are from that point" [23]. Thus, the concept point describes a region of the instance space that is dense in terms of instances from the positive bags. Another algorithm, the Multiple-Instance Expectation–Maximization and Diverse Density (MI-EMDD) extends the MI-DD and forms a generic framework that can be used to convert a MIL problem into a single-instance setting by using the expectation–maximization (EM) algorithm [39]. The label of a bag is determined by the instance with the highest likelihood of being positive among all instances in that bag. The authors use a set of hidden variables that are estimated using the EM approach to find out which instance determines the label of a given bag. Starting with an initial guess of the concept point h, obtained by checking points from the positive bags, the MI-EMDD iteratively performs the following two steps until it converges: *E-step*, i.e., the current hypothesis h is used to select one instance from each bag that is the most likely to be responsible for the label given to the bag, and *M-step*, i.e., a new instance is estimated to maximize the diverse density of the hypothesis h by using a gradient search. Although the standard logistic regression model cannot be directly used for solving MIL problems, an indirect estimate of the logistic model [Multiple-Instance Logistic Regression (MI-LR)] was proposed in [37]. The authors extended the standard instance-level model by indicating how the instance-level class probabilities were combined from the bag-level probability so that the actual class label for each instance was not required.

Two other approaches adapted support vector machine (SVM) to solve the MIL problems. In standard single-instance SVMs, training data are given as a set of instance–label pairs (b_i, y_i), where b_i is an instance and y_i is a label of b_i. Multiple-Instance Sequential Minimal Optimization (MI-SMO) method [29] extended the standard SVM by using a bag-level multi-instance kernel function. First, the bag-level kernel is defined, and the bag–label pairs (B_i, Y_i) are then used instead of instance–label pairs (b_i, y_i). In [1], another SVM-based algorithm was introduced: Multiple-Instance SVM (MI-SVM). The main idea was to transform the multi-instance data setting into a single-instance setting by properly assigning the

Manuscript received October 16, 2011; revised February 20, 2012; accepted April 20, 2012. Date of publication June 18, 2012; date of current version January 11, 2013. The work of D. T. Nguyen was supported by a National Science Foundation Graduate Assistance in Areas of National Need Grant. The work of C. D. Nguyen was supported by The Diabetes Research Foundation of Western Australia. The work of K. J. Cios was supported by the National Institutes of Health under Grant 1R01HD056235-01A1. This paper was recommended by Editor J. Basak.

D. T. Nguyen and R. Hargraves are with Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: nguyendt22@vcu.edu; rhobson@vcu.edu).

C. D. Nguyen is with The Western Australian Institute for Medical Research Centre for Diabetes Research and the Centre for Medical Research, The University of Western Australia, Perth, W.A. 6009, Australia (e-mail: cao.nguyen@uwa.edu.au).

L. A. Kurgan is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: lkurgan@ece.ualberta.ca).

K. J. Cios is with Virginia Commonwealth University, Richmond, VA 23284 USA, and also with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 44-100 Gliwice, Poland (e-mail: kcios@vcu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2201468

unobserved class label to each individual instance in the positive bags. Then, the standard SVM learning scheme was used to assign the labels. The goal was to find the maximum-margin multi-instance separating hyperplane in which all instances in each negative bag were located on one side of the hyperplane and at least one positive instance from all positive bags was located on the other side of the hyperplane.

Still another approach is to use distance measures. The idea of the Multiple-Instance Optimal Ball (MI-OptimalBall) method [3] was to find an optimal ball in the feature space such that all negative bags were outside of this ball. In other words, the surface of the ball separates positive from negative instances. The center of the optimal ball is an instance from a positive bag, and the radius of the ball is determined based on training data. During classification, if all instances in a test bag lie outside of the optimal ball, then the bag is classified as negative; otherwise, it is classified as positive. Citation-K nearest neighbor (Citation-KNN) algorithm used the K-nearest neighbor (KNN) algorithm to compute the shortest Hausdorff distance between any two instances [32]

$$Dist(A, B) = Min\,(Dist(a_i, b_j)) = MinMin\|a - b\|,$$
$$1 \leq i \leq n; \quad 1 \leq j \leq m; \quad a \in A; \quad b \in B$$

where A and B denote bags, a_i and b_j are instances, and n and m are the total numbers of instances in bags A and B, respectively. The new bags are labeled using the *KNN* algorithm. However, in the MIL scenario, the majority label of the KNNs of an unlabeled bag may not always be the true label of that bag. This is because the majority voting scheme may not work in the presence of false positive instances in the positive bags. This weakness was overcome by adding a citation approach which considered not only the bags as the nearest neighbors (known as *References*) of a bag B but also the bags that count B as their neighbors (known as *Citations*). Hence, the Citation-KNN predicts the label of a bag based on both *References* and *Citations* of that bag. Unfortunately, Citation-KNN cannot predict labels of all individual instances. The Multiple-Instance Nearest Neighbor with Distribution Learner (MI-NND) method assumes that each bag contains enough instances and that all dimensions of the data are equally relevant to classification [36]. Under these assumptions, a distribution is derived for each dimension of each bag, and the obtained distributions are used directly for classification, instead of doing it on the original data. MI-NND operates in two steps. The first formulates distribution for each bag based on training data by deriving a Gaussian model for each dimension of each bag. These Gaussian distributions are then used to represent the original data. The second step finds the nearest neighbors of a test bag where the test bag is also represented by the Gaussian distributions for each dimension. Next, the testing and training distributions are compared using Kullback–Leibler distance, and the category is decided based on the closest match (the same as in the KNN algorithm).

One of the more recent algorithms is Multiple-Instance Learning via Embedded instance Selection (MILES) [8], which converts MIL problem to standard supervised learning by mapping each bag into a feature space defined by instances in the training bags, using instance similarity measure and the one-norm SVM [42] to solve it. Two other methods are also based on the SVM but use deterministic annealing to identify all labels, namely, the annealing for identifying all labels-SVM (AL-SVM) and annealing for identifying the witness-SVM (AW-SVM) algorithms [17]. Extended Random Forest (mi-Forest) algorithm [6] defines labels of all instances in every positive bag as random variables and uses deterministic annealing to find true labels [21]. All of the aforementioned methods were developed using either the probabilistic EM, nearest neighbor, regression, or SVM approach.

In this paper, we propose a new algorithm, called mi-DS, which is based on standard rule classifier DataSqueezer [19]. mi-DS uses rules generated by DataSqueezer as a metric for measuring distance between two bags in training data. Importantly, the proposed here approach can be used as a generic framework for transforming other rule-based algorithms for solving MIL problems.

II. mi-DS Algorithm

First, we briefly overview DataSqueezer and then introduce the mi-DS algorithm. The former is a supervised inductive rule learner, whose main advantage is efficient rule induction process (log linear) that involves two steps [19]. In the first, it performs data reduction via the use of the prototypical concept learning. In the second, it generates rules by performing greedy hill-climbing search on the reduced data. A rule, while being generated, is checked against the reduced data, and if it covers any data in the negative (reduced) data set, then a new selector is added. The maximum depth of the search is equal to the number of features in the training data. All instances covered by the generated rule are then removed, and the second step is repeated until all data points are covered by the generated rules. Because some of the data sets that we will use in the experiments are continuous, we perform front-end discretization using information-theoretic class-attribute interdependence maximization (CAIM) algorithm that maximizes the class-attribute interdependence [20], [26]. The mi-DS algorithm operates in three phases.

Phase I: Rule Generation

Step 1) The training data are divided into positive/negative tables that include only instances with the positive/negative labels. Inconsistent instances (those that appear in both negative and positive bags) are removed from the positive table.

Step 2) RULEPOS and RULENEG tables are generated. After performing data reduction, a rule is generated by incrementally adding features by checking the positive table against the negative table for generating the RULEPOS table and vice versa for the RULENEG table.

Phase II: Construction of the Similarity Matrix

Having generated the RULEPOS and RULENEG tables, a similarity matrix M is built to measure the similarity between

TABLE I
SIMILARITY MATRIX CONSTRUCTED IN PHASE II. M_{ij} IS THE NUMBER OF THE RULES COVERING BAGS b_i AND b_j; $(n+1)$TH ROW AND $(n+1)$TH COLUMN ARE USED FOR MAKING PREDICTIONS

Row/Col	1	2	3	...	n	n+1
1	0	M_{12}	M_{13}	...	M_{1n}	
2	M_{21}	0	M_{23}	...	M_{2n}	
....	...	...	0	...	...	
n	M_{n1}	M_{n2}	M_{n3}	...	0	
n+1						

```
ConstructMatrix:
  Initialize the matrix M with n+1 rows and n+1
columns
  FOR (k=1 to n)
      ScanInRuleTable($b_k$, RULEPOS)
      ScanInRuleTable($b_k$, RULENEG)
  END
  OUTPUT (M)

ScanInRuleTable($b_k$,RULE):
  FOR (every instance i  in bag $b_k$)
   FOR (each rule r in RULE)
     IF (instance i  covered by rule r)
      FOR (each bag $b_j$ IN covered bags of the
      rule r) AND
         (k<>j)
                        ⇒ M[k,j]++
         END
      END
  END
```

Fig. 1. Pseudocode for creating similarity matrix in phase II.

```
Set (n+1)th row and (n+1)th column in the
  similarity matrix M to 0
ScanInRuleTable(B,RULEPOS)
ScanInRuleTable(B,RULENEG)
Build list REF of bags which include the
  top R nearest references to B
Build list CITER of bags which include
  the top C nearest citers of B
Calculate P and N from REF and CITER
```

Fig. 2. Pseudocode for calculating similarity indices in phase III for a test bag B.

```
IF (P > N)                      ⇒     B is positive
ELSE IF (P < N)                 ⇒     B is negative
ELSE IF ($R_p$ > $R_n$)         ⇒     B is positive
ELSE IF ($R_p$ < $R_n$)         ⇒     B is negative
ELSE IF ($C_p$ > $C_n$)         ⇒     B is positive
ELSE IF ($C_p$ < $C_n$)         ⇒     B is negative
ELSE IF (total number of positive bags > total
   number of negative bags in {$b_1$,$b_2$,..,$b_n$})
                                ⇒     B is positive
  ELSE                          ⇒     B is negative
```

Fig. 3. Pseudocode for classification of a test bag B.

two bags, as shown in Table I. The similarity matrix has $n+1$ rows and $n+1$ columns, where n is the number of bags in the training data. The value M_{ij} stored in this matrix represents the total number of positive and negative rules such that the bags b_i and b_j are covered by the rules. In other words, the bag b_i refers M_{ij} times to the bag b_j, while the bag b_j is cited M_{ij} times by bag b_i. The $(n+1)$th row and the $(n+1)$th column are used to store the numbers of *References* (R) and *Citations* (C), which are later used for assigning a label to a test bag. Fig. 1 shows the pseudocode for generation of the similarity matrix.

Phase III: Prediction of a Label of a Test Bag

A measure to quantify similarity between a test bag and training bags was designed to decide a label for a new bag. The measure is a modification of the one introduced in [32]. Specifically, instead of using the metric distances between the bags, we use a number u_{ij} of rules that cover all the bags.

For a given test bag, we first fill the $(n+1)$th row and the $(n+1)$th column in the similarity matrix, in the same way as in phase II but only for the last row and the last column of the similarity matrix (see Fig. 2). To make a decision about the class of a test bag, we use the number of *References* R and the number of *Citations* C. To do so, we first count, in row $(n+1)$, the bags that have R maximum values to find the bags (both positive and negative) that the test bag most often referred to. Similarly, we scan all rows of the similarity matrix and count the C maximum values with the restriction that one of them must come from column $(n+1)$—the rule that covers the test bag. The purpose is to find bags which the test bag most often cited. Then the P and N values are calculated:

$$P = R_p + C_p$$
$$N = R_n + C_n$$

where $R_p(R_n)$ is the number of positive (negative) bags in the R selected bags and $C_p(C_n)$ is the number of positive (negative) bags in the selected C bags.

The pseudocode for the aforementioned procedure is shown in Fig. 2. The test bag is predicted as positive if $P > N$ and as negative otherwise. When $P = N$, we use an algorithm shown in Fig. 3.

mi-DS uses three thresholds: one for rule pruning and two for rule generalization. The pruning threshold is used to decide whether to add a new feature to the rule. A feature is added if

$$\max_{1\le j\le K}\{a_{ij}\cdot v_j\} > pruning\ threshold$$

where K is the total number of features, a_{ij} is the count of occurrence of value a_i in the jth feature, and v_j is the total number of values of the jth feature; otherwise, generation of a new rule starts.

The second and third generalization thresholds specify the following: 1) the number of bags to be considered as the nearest neighbors of a given bag (R) and 2) the number of bags that a given bag considers as its nearest neighbors (C). The range for each of the latter two thresholds is $[1, \min(\#positive\ bags, \#negative\ bags)]$. The values of the three thresholds are optimized for acceptable accuracy using tenfold cross-validation (10FCV).

Computational complexity of the mi-DS is analyzed next. In phase I, the mi-DS requires $O(N_{\text{pos}}N_{\text{neg}})$ computations

TABLE II
BAGS OF DATA

Bag #	Features			Class label
	Shape	Color	Width	
1	Rect	Green	200	+
	Circle	Blue	400	
	Circle	**Green**	**300**	
2	**Circle**	**Green**	**300**	-
	Rect	Blue	200	
	Triangle	**Blue**	**200**	
3	Rect	Green	200	+
	Circle	Red	300	
	Triangle	**Blue**	**200**	
4	**Rect**	**Blue**	**200**	-
	Triangle	**Blue**	**200**	
5	Rect	Green	200	+
	Circle	Blue	300	
	Rect	**Blue**	**200**	

TABLE III
POSITIVE AND NEGATIVE TABLES OBTAINED IN STEP 1

Positive table

Features			Number of times occurred	Bag #
Shape	Color	Width		
Rect	Green	200	1	1
Circle	Blue	400	1	1
Rect	Green	200	1	3
Circle	Red	300	1	3
Rect	Green	200	1	5
Circle	Blue	300	1	5

Negative table

Features			Number of times occurred	Bag #
Shape	Color	Width		
Circle	Green	300	1	2
Rect	Blue	200	1	2
Triangle	Blue	200	1	2
Rect	Blue	200	1	4
Triangle	Blue	200	1	4

for step 1, where N_{pos} is the number of instances in the positive bags and N_{neg} is the number of instances in the negative bags. The complexities of step 2, after [19], are $O(R_{\mathrm{pos}} K N_{\mathrm{pos}} \log N_{\mathrm{pos}})$ for generating the RULEPOS and $O(R_{\mathrm{neg}} K N_{\mathrm{neg}} \log N_{\mathrm{neg}})$ for generating the RULENEG tables, where R_{pos} and R_{neg} are the total numbers of rules in RULEPOS and RULENEG, respectively, N is the total number of instances, and K is the number of features. In phase II, the construction of the similarity matrix takes $O(N\ R_{\mathrm{total}}\ K)$, where R_{total} is the total number of rules. Therefore, the computational complexity of the algorithm is approximately $O(R_{\mathrm{total}}\, KN\ \log N)$.

mi-DS minimizes the number of rules and the number of selectors in each rule, while, at the same time, minimizing the number of false positives. Although it may not find a global optimum, it produces a locally optimal solution with respect to the aforementioned criteria.

The data shown in Table II are used to illustrate the working of mi-DS algorithm.

Phase I: Rule Generation

Step 1) Inconsistent instances shown in bold in Table II [i.e., (Circle, Green, 300), (Triangle, Blue, 200), and (Rect, Blue, 200)] are removed from the positive table but are kept in the negative table, as shown in Table III. The bag number and the count of the number of occurrences of each instance are also shown in Table III.

Step 2) In this step, RULEPOS and RULENEG tables are generated as shown in Table IV); the table also shows bag numbers covered by these rules.

TABLE IV
RULEPOS AND RULENEG TABLES; * INDICATES ANY VALUE OF A FEATURE

RULEPOS

Features			Number of times occurred	Bags
Shape	Color	Width		
Rect	Green	*	3	{1,3,5}
Circle	Blue	*	2	{1,5}
Circle	Red	*	1	{3}

RULENEG

Features			Number of times occurred	Bags
Shape	Color	Width		
*	Blue	200	4	{2,4}
Circle	Green	*	1	{2}

Phase II: Construction of the Similarity Matrix

Table V shows the similarity matrix at the end of phase II. For example, bag 1 in Table II has three instances. Instance 1 (Rect, Green, 200) is covered by rule (Rect, Green, *) in the RULEPOS table; because this rule also covers bags 3 and 5, we increase u_{13} and u_{15}. Instance 2 (Circle, Blue, 400) is covered by rule (Circle, Blue, *) in RULEPOS; because it also covers bag 5, we increase u_{15}. Instance 3 (Circle, Green, 300) is covered by rule (Circle, Green, *) in RULENEG; because it also covers bag 2, we increase u_{12}. This results in the first row in Table V being (0, 1, 1, 0, 2).

TABLE V
SIMILARITY MATRIX OBTAINED IN PHASE II

Bag# (label)	1(+)	2(-)	3(+)	4(-)	5(+)	Test bag
1(+)	0	1	1	0	2	
2(-)	0	0	0	2	0	
3(+)	1	1	0	1	2	
4(-)	0	2	0	0	0	
5(+)	2	1	1	1	0	
Test bag						

TABLE VI
TEST BAG B WITH FIVE INSTANCES

Shape	Color	Width
Rect	Green	300
Circle	Green	100
Rect	Blue	200
Circle	Blue	100
Rect	Green	100

Phase III: Prediction of Unseen Test Bags

Next, we perform classification of a test bag B that is shown in Table VI. First, every instance in the bag is scanned to see whether it is covered by any rule from either RULEPOS or RULENEG tables. If an instance in bag B is covered by a rule, then we increment the values of $M[n+1, j]$ and $M[j, n+1]$ for every bag j in the bag lists (shown in the last column of Table IV) for both RULEPOS and RULENEG tables.

For example, the instance (Rect, Green, 300) in bag B is covered by rule (Rect, Green, *) from the RULEPOS table, so we increase $M[n+1, j]$ and $M[j, n+1]$ by one, with bag numbers being $j = 1, 3, 5$. The last row and the last column of the similarity matrix are updated, as shown in Table VII. Finally, the algorithm shown in Fig. 2 is used with $R = 2$ and $C = 2$; the resulting R_{set} list (it contains bags which test bag B refers to) and C_{set} list (it contains bags which refer to test bag B) values are

$$R_{\text{set}} = \{bag1(+), bag5(+)\}$$
$$C_{\text{set}} = \{bag1(+), bag2(-), bag3(+), bag4(-), bag5(+)\}.$$

The label of the test bag B is then predicted based on the labels of bags in R_{set} and C_{set}. Since $R_p = 2, R_n = 0, C_p = 3, C_n = 2, P = R_p + C_p = 5$, and $N = R_n + C_n = 2$, the test bag B is predicted as positive.

III. EXPERIMENTAL RESULTS

The mi-DS algorithm performance is evaluated by comparing it with many existing MIL algorithms, using accuracy and rank, on a diverse set of 26 benchmark data sets, which are briefly described below and summarized in Table VIII. Musk1 and Musk2 are used to predict whether a given molecule emits a musky odor, where each bag describes one molecule [13]. Three mutagenicity data sets concern analysis of drug activity with the goal of predicting whether a given drug molecule is mutagenic or nonmutagenic; the molecules are represented by atoms, atomic bonds, and chains [31]. The next three data sets concern identification of the following target objects in images: elephants, foxes, and tigers. They include bags representing photographs of animals [14]. An image is represented by a set of segments (pixel regions) characterized by color, texture, and shape. A bag represents an image, and instances in a bag represent individual segments of that image. A positive bag is an image that contains a given animal. Data set Trx was used to solve protein identification problem [14]. The data sets EastWest and WestEast are used to predict whether a train is eastbound or westbound [14]. There are two artificial data sets. Artificial 1 includes 200 bags with 100 positive and 100 negative bags; each bag has 20 instances with 12 features uniformly assigned as 0 or 1 [10]. The concept is described by the first three features, (1, 0, 1), namely, if there exists in a bag at least one instance with the format of (1, 0, 1) while the remaining features take on any value, then the bag is positive. Artificial 2 includes 100 positive and 100 negative bags, each with 20 instances. Each instance has two features which are drawn randomly from area [0, 100] $\times$ [0, 100] in R^2. An instance is labeled positive if the features fell in the square 5 $\times$ 5 in the middle, i.e., in the square [48, 52] $\times$ [48, 52]. The bag is labeled as positive if at least one instance fell within this region and as negative otherwise [23]. We also use five categories of preprocessed image data sets (Africa, Beach, Building, Bus, and Dinosaur) from the Corel Corporation; each category has 100 images of size 384 $\times$ 256 or 256 $\times$ 384 [7]. The next three text categorization data sets are as follows. Given a name of a protein and a full-text journal article, the task is to determine whether this protein–article pair can be annotated with a particular gene ontology (GO) term. For the MIL setting, each article is represented as a bag. An instance in a bag refers to a paragraph in an article. Each paragraph is represented as a set of word occurrence frequencies plus data about the nature of the protein–GO relationship. The assumption is that, if there is at least one instance (paragraph) that is related to

TABLE VII
SIMILARITY MATRIX WITH SCORES FOR THE TEST BAG B, WITH $R = 2$ AND $C = 2$

Rule covering bag# (label)	*1(+)*	*2(-)*	*3(+)*	*4(-)*	*5(+)*	Test bag B
1(+)	0	1	1	0	2	3
2(-)	0	0	0	2	0	2
3(+)	1	1	0	1	2	2
4(-)	0	2	0	0	0	1
5(+)	2	1	1	1	0	3
Test bag B	3	2	2	1	3	0

TABLE VIII
DATA SETS USED IN TESTING

Data sets	# Features	# of bags			# of instances			ratio = # instances / # bags		
		Total	Pos	Neg	Total	Pos	Neg	Total	Pos	Neg
Musk1	167	92	47	45	476	207	269	5.2	4.4	6
Musk2	167	102	39	63	6598	1017	5581	64.7	26.2	88.6
Mutagenesis atoms	11	188	125	63	1618	1073	545	8.6	8.6	8.7
Mutagenesis bonds	17	188	125	63	3995	2955	1040	21.3	23.6	16.5
Mutagenesis chains	25	188	125	63	5349	4116	1233	28.5	32.9	19.6
Image Elephant	231	200	100	100	1391	762	629	7	7.6	6.3
Image Fox	231	200	100	100	1320	647	673	6.6	6.5	6.7
Image Tiger	231	200	100	100	1220	676	544	6.1	6.8	5.4
Trx	8	193	25	168	26611	3341	23270	137.9	133.6	138.5
East-West	24	20	10	10	213	81	132	8.9	8.1	13.2
West-East	24	20	10	10	213	81	132	8.9	8.1	13.2
Artificial 1	13	200	100	100	4000	2000	2000	20	20	20
Artificial 2	3	200	100	100	4000	2000	2000	20	20	20
Corel Corp. images	9	250	50	200	variable	variable	variable	variable	variable	variable
Component	200	3130	423	2707	36894	9104	27790	11.8	21.5	10.3
Function	200	5242	443	4799	55536	5543	49993	10.6	12.5	10.4
Process	200	11718	757	10961	118417	9272	109145	10.1	12.3	10

the protein–GO relationship, the bag is positive; otherwise, it is negative. There are three data sets corresponding to these GO terms: Component, Function, and Process [14]. All these data sets (see Table VIII) are characterized by different numbers of features ranging from 3 to 231, different numbers of instances ranging from 213 to 118 417, and different ratios of positive to negative instances ranging from 0.08 to 3.3.

mi-DS algorithm is compared with 21 MIL algorithms, including the following eight that are implemented in Waikato Environment for Knowledge Analysis (WEKA): MI-DD [23], MI-EMDD [39], Modified Diverse Density (MDD) [13], MI-LR [37], MI-SVM [1], MI-SMO [29], MI-OptimalBall [3], Citation-KNN [32], and MI-NND [36]. Experiments are performed on a 2.4-GHz CPU with 3-GB RAM, and each algorithm is parameterized to maximize its accuracy using 10FCV.

Table IX shows the results in terms of classification accuracy. Discretization, using CAIM algorithm [20], was performed on 8 (out of 13) data sets, namely, Musk1, Musk2, Atoms, Bonds, Chains, Elephants, Fox, and Tigers. Results are ranked according to accuracy. Notice that no method is universally best. The average rank, over all data sets, of mi-DS is 6.4, which is the second best, after MI-DD at 6.2. The third best method, MI-SMO, has a rank of 7.3. More importantly, mi-DS results were the best four times among the 13 data sets, while the other methods performed as the best at most once. It is worth to mention that Citation-KNN algorithm (on the Elephants, Tigers, and Fox data sets) and the MI-SVM algorithm (on the three mutagenesis data sets) generated only positive labels (using default hypothesis approach), which resulted in their very low accuracies.

For comparison with the algorithms listed on positions 11–22 in Table IX (Multiple-Instance Graph (MIGraph) [41], multiple-instance Graph (miGraph) [41], Multiple-Instance Kernel (MI-Kernel) [16], p-Posterior Mixture-Model Kernels for Multiple Instance Learning (PPMM) [33], Random Forest [6], mi-Forest [21], MILES [8], AW-SVM [17], AL-SVM [17], Multiple Instance Classification Algorithm (MICA) [24], Convex-Hull MIL Fisher's Discriminant (CH-FD) [15], and Multi-Instance Learning by Semi-Supervised Support Vector Machine (MissSVM) [40]), we could show only the published results as neither they were implemented in WEKA nor their codes were available.

To investigate the statistical significance of the performance differences between mi-DS and nine MIL algorithms, we use Wilcoxon signed-ranked test [12] with the results shown in Table X. The Wilcoxon test ranks differences in the accuracy of two classifiers for each data set. Using confidence level $\alpha = 0.05$ with the number of data sets $N = 13$, the null hypothesis that a given pair of algorithms performs equally well should be rejected when $T \leq 17$. Table X shows that mi-DS is significantly better than the Citation-KNN, MDD, MI-EMDD, MI-NND, MI-SVM, and MI-LR algorithms, while performing on par with the MI-DD, MI-SMO, and MI-OptimalBall algorithms.

TABLE IX
COMPARISON WITH 21 MIL ALGORITHMS USING 10FCV AND RANK [ ]; THE BEST RESULTS ARE SHOWN IN BOLD ITALIC

#	Data set / Algorithm	Musk 1	Musk 2	Ele-phant	Fox	Tiger	Atoms	Bonds	Chains	Trx	East West	West East	Artifi-cial 1	Artifi-cial 2	Avg. Rank
1	mi DS	86.7 [14]	77 [18]	79.5 [13]	64.5 [3]	73.4 [18]	***80.0 [1]***	74.7 [3]	***79.5 [1]***	89.0 [2]	64.2 [4]	60.0 [4]	***100.0 [1]***	***68.8 [1]***	6.4
2	Citation KNN	94.4 [2]	85 [7]	50* [21]	50* [20]	50* [21]	75.3 [2]	73.2 [5]	74.7 [6]	87.6 [4]	55 [7]	55.0 [6]	60.5 [3]	67.0 [2]	8.2
3	MI DD	92.2 [3]	73.0 [21]	83.0 [4]	65.5 [2]	74.0 [17]	72.6 [4]	75.3 [2]	76.8 [3]	***90.0 [1]***	76 [2]	25.0 [9]	76.1 [2]	43.2 [10]	6.2
4	MDD	78.9 [21]	74.0 [20]	80.0 [12]	***70.0 [1]***	75.5 [15]	72.1 [6]	73.2 [5]	76.8 [3]	87.1 [5]	55 [7]	55.0 [6]	51.3 [9]	51.8 [4]	8.8
5	MI EMDD	88.9 [6]	90 [3]	73.0 [20]	59.0 [14]	74.5 [16]	73.2 [3]	72.1 [7]	73.7 [7]	87.9 [3]	45 [10]	30.0 [8]	56 [5]	51.8 [4]	8.2
6	MI NND	75.1 [22]	72.8 [22]	74.8 [18]	58.5 [16]	66.5 [20]	45.6 [10]	31.2 [10]	41.2 [10]	87.0 [6]	57 [6]	***74.0 [1]***	55.2 [7]	45.9 [8]	12.0
7	MI-SMO	87.8 [11]	84 [9]	79.0 [14]	58.0 [17]	82.5 [5]	70 [7]	***84.2 [1]***	78.4 [2]	86.1 [8]	75 [3]	65.0 [3]	54.3 [8]	49.1 [7]	7.3
8	MI-SVM	89.2 [5]	83.9 [11]	81.6 [8]	49.4 [21]	75.7 [14]	66.5* [9]	66.5* [9]	66.5* [9]	87.0 [6]	52 [9]	22.0 [10]	55.5 [6]	50.0 [6]	9.5
9	MI-LR	87.8 [11]	85 [7]	78.0 [16]	56.0 [18]	77.0 [12]	70 [7]	71.1 [8]	76.8 [3]	85.3 [9]	60 [5]	60.0 [4]	57.8 [4]	52.7 [3]	8.2
10	MI-Optimal Ball	80 [20]	77 [18]	77.5 [17]	53.0 [19]	67.0 [19]	72.6 [4]	74.2 [4]	71.6 [8]	84.9 [10]	***79 [1]***	70.0 [2]	30 [10]	43.6 [9]	10.8
11	MI Graph	90 [4]	90 [3]	85.1 [2]	61.2 [10]	81.9 [9]	-	-	-	-	-	-	-	-	-
12	mi Graph	88.9 [6]	90.3 [2]	***86.8 [1]***	61.6 [9]	***86.0 [1]***	-	-	-	-	-	-	-	-	-
13	MI-Kernel	88 [9]	89.3 [5]	81.4 [9]	60.3 [11]	84.2 [2]	-	-	-	-	-	-	-	-	-
14	PPMM	***95.6 [1]***	81.2 [15]	82.4 [5]	60.3 [11]	82.4 [6]	-	-	-	-	-	-	-	-	-
15	Random Forest	85 [17]	78 [17]	74.0 [19]	60.0 [13]	77.0 [12]	-	-	-	-	-	-	-	-	-
16	mi-Forest	85 [17]	82 [14]	84.0 [3]	64.0 [4]	82.0 [8]	-	-	-	-	-	-	-	-	-
17	MILES	88 [9]	83 [12]	81.0 [10]	62.0 [7]	80.0 [10]	-	-	-	-	-	-	-	-	-
18	AW-SVM	86 [15]	84 [9]	82.0 [7]	64.0 [4]	83.0 [3]	-	-	-	-	-	-	-	-	-
19	AL-SVM	86 [15]	83 [12]	79.0 [14]	63.0 [6]	78.0 [11]	-	-	-	-	-	-	-	-	-
20	MICA	84.4 [19]	***90.5 [1]***	80.5 [11]	58.7 [15]	82.6 [4]	-	-	-	-	-	-	-	-	-
21	CH-FD	88.8 [8]	85.7 [6]	82.4 [5]	62.0 [7]	82.2 [7]	-	-	-	-	-	-	-	-	-
22	MissSVM	87.6 [13]	80 [16]	-	-	-	-	-	-	-	-	-	-	-	-

**Algorithms predicted bags belong to only one class*

TABLE X
WILCOXON TEST COMPARISONS OF ACCURACIES ON 13 DATA SETS. BOLD INDICATES ALGORITHMS FOR WHICH mi-DS WAS SIGNIFICANTLY BETTER

Pair of compared algorithms	R+	R–	T=min(R+,R-)
mi-DS vs. Citation-KNN	**76**	**15**	**15**
mi-DS vs. MI-DD	57	34	34
mi-DS vs. MDD	**78**	**13**	**13**
mi-DS vs. MI-EMDD	**77.5**	**13.5**	**13.5**
mi-DS vs. MI-NND	**83**	**8**	**8**
mi-DS vs. MI-SMO	48.5	42.5	42.5
mi-DS vs. MI-SVM	**77**	**14**	**14**
mi-DS vs. MI-LR	**74**	**16**	**16**
mi-DS vs. MI-Optimal Ball	70.5	20.5	20.5

Friedman test with $N = 13$ data sets, $k = 10$ algorithms, and $\alpha = 0.025$ was also used. The calculated F value is 20.396, which is greater than the table value of $df = 19.023$, which means that mi-DS is significantly different from the nine algorithms.

Next, we run mi-DS on Musk1 data 100 times. The order of bags was changed randomly, and then, 10FCV was performed. The minimum, maximum, and average accuracies were 0.84, 0.99, and 0.93, respectively. Note that the average from this experiment is better than the one reported in Table IX; the reason is that, for the results reported in Table IX, we used the training and testing data sets as specified in [14]. The purpose of this experiment was to show that the performance of mi-DS was not adversely affected by the order of instances in the training data set.

In the following experiments, we use three textual data sets. Note that class distributions in these data sets are extremely unbalanced. We used the same number of positive and negative bags in training: 359 for Component, 385 for Function, and 620 for Process data sets. The remaining bags were used for testing, namely, 64 positive and 2348 negative bags for Component, 58 positive and 4414 negative bags for Function, and 137 positive and 10341 negative bags for Process.

mi-DS was additionally tested on preprocessed image data sets. Images were considered as bags, each containing from 3 to 13 instances of the same object (e.g., dinosaur), and described by nine features [7]. Images belong to ten categories, with 100 images per category; the size of the images is 384 $\times$ 256 or 256 $\times$ 384. Dinosaurs' example images, original and preprocessed, are shown in Fig. 4 (Fig. 5).

The training and testing data were created as follows: Choose randomly five categories, and then, select randomly 50 images from each category. This results in a data set of 250 bags for training and 250 bags for testing. Next, 50 images from one category are designated as positive, and the remaining 200 are designated as negative. Table XI summarizes the performance of the ten MIL algorithms on Africa, Beach, Building, Bus, and Dinosaur images.

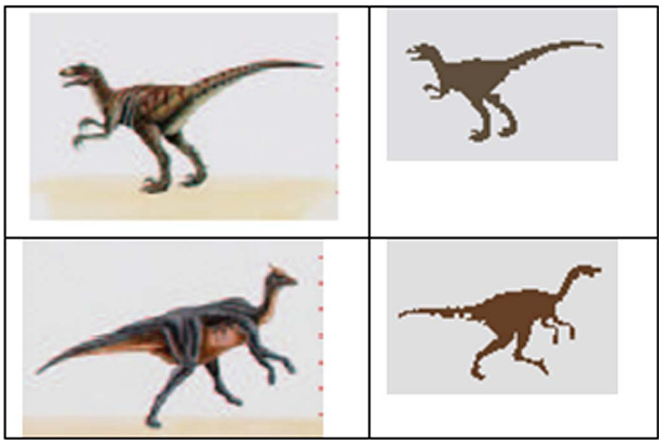
Fig. 4. Dinosaur images from Corel's data: Before (left column) and After (right column) preprocessing.

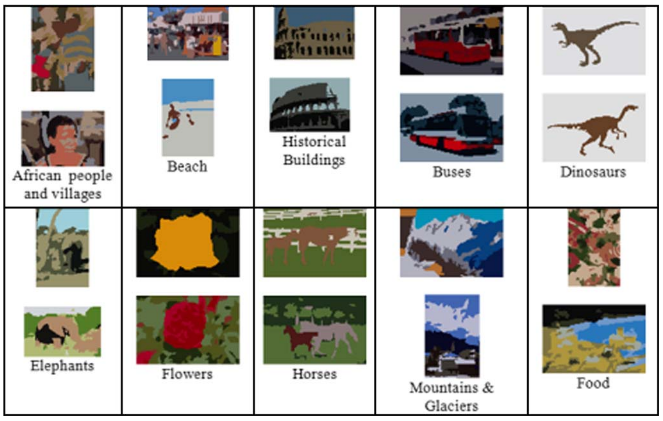

Fig. 5. Images from Corel's data after preprocessing.

Text categorization experiments, shown in the first three rows of Table XI, show that mi-DS performs on par with MDD, MI-SMO, and MI-OptimalBall algorithms, in terms of accuracy.

Table XII compares the running times of the ten MIL algorithms required for building models of the 21 data sets. Note that, in terms of average rank, mi-DS performed better than five algorithms and performed comparable to or worse than MI-LR, MI-SMO, MI-OptimalBall, and Citation-KNN. Table IX, however, shows that the latter four algorithms are outperformed by mi-DS in terms of predictive accuracy. Although MI-DD achieves better rank with respect to accuracy, it is outranked by mi-DS in terms of running time.

This prompted us to use eight larger data sets (Musk2, Elephants, Fox, Tiger, Trx, Component, Function, and Process) for more detailed analysis of running times and accuracies. They are summarized in Table XIII, which shows the minimum, maximum, and average of accuracies and of running times. Notice that mi-DS obtains a similar level of accuracy to those of MI-DD, MDD, and MI-SMO, but these three methods needed much longer running times (on average, by a factor of five), while two methods that have relatively short running times (MI-EMDD and MI-LR) have lower accuracies than mi-DS. This analysis shows that mi-DS strikes a good tradeoff between

TABLE XI
COMPARISON WITH NINE MIL ALGORITHMS ON TEXT (THE FIRST THREE) AND IMAGE DATA SETS; THE BEST RESULTS ARE SHOWN IN BOLD ITALIC

Data set	mi-DS	Citation-KNN	MI-DD	MDD	MI-EMDD	MI-NND	MI-SMO	MI-SVM	MI-LR	MI-Optimal Ball
Component Train: 359 Pos and 359 Neg bags Test: 64 Pos and 2348 Neg bags	84.1 [3]	63.8 [9]	74.8 [7]	***91 [1]***	61.4 [10]	77.5 [6]	82.3 [5]	67.1 [8]	86.2 [2]	84.1 [3]
Function Train: 385 Pos and 385 Neg bags Test: 58 Pos and 4414 Neg bags	***97.1 [1]***	71.9 [8]	92.2 [5]	95.6 [3]	78.1 [7]	93.8 [4]	91.4 [6]	62.5 [10]	68.2 [9]	***97.1 [1]***
Process Train: 620 Pos and 620 Neg bags Test: 137 Pos and 10341 Neg bags	87.3 [5]	80.1 [9]	91.4 [3]	***96.8 [1]***	86.2 [7]	85.7 [8]	91.3 [4]	78.6 [10]	93.8 [2]	87.3 [5]
Africa Test and Train has 50 Positive bags and 200 Negative bags	87.6 [5]	85.2 [6]	***92.8 [1]***	84 [7]	90.8 [2]	78.4 [10]	83.6 [8]	80 * [9]	90.4 [3]	88.4 [4]
Beach Test and Train has 50 Positive bags and 200 Negative bags	82.8 [7]	88 [2]	86 [4]	84 [6]	***88.8 [1]***	32.8 [10]	80 * [8]	80 * [8]	86 [4]	88 [2]
Buiding Test and Train has 50 Positive bags and 200 Negative bags	84 [5]	85.6 [3]	***91.2 [1]***	84.4 [4]	87.6 [2]	43.6 [10]	80 * [8]	80 * [8]	82 [7]	82.4 [6]
Bus Test and Train has 50 Positive bags and 200 Negative bags	92.4 [3]	91.2 [5]	93.2 [2]	80.8 [9]	89.2 [7]	89.2 [7]	***94.4 [1]***	80 * [10]	92.4 [3]	90 [6]
Dinosaur Test and Train has 50 Positive bags and 200 Negative bags	98 [6]	99.6 [3]	***100 [1]***	80 * [8]	100 [1]	58.8 [10]	88.4 [7]	80 * [8]	99.6 [3]	99.2 [5]
Average Rank	4.4	5.6	3.0	4.9	4.6	8.1	5.9	8.9	4.1	4.0

* *Algorithms predicted bags belong to only one class*

running time and accuracy. Note that we were unable to evaluate running times for the following 12 algorithms because their implementations were not available: MIGraph, miGraph, MI-Kernel, PPMM, Random Forest, mi-Forest, MILES, AW-SVM, AL-SVM, MICA, CH-FD, and MissSVM. However, comparison of the asymptotic complexity between mi-DS and MIGraph, miGraph, MI-Kernel, and PPMM showed that the latter four were competitive with mi-DS in terms of accuracy (Table IX).

As shown in Section II, mi-DS has computational complexity of $O(K^* n^* \log(n))$. This is better than the complexities of MIGraph, miGraph, MI-Kernel, and PPMM. In particular, the computational complexities of MIGraph and miGraph are $O(n^2 + K^2)$ and $O(n^2)$, respectively. MI-Kernel's complexity is $O(K^* n^2)$, while PPMN performs exhaustive search that is prohibitive in practice. This means that their running times are much longer than mi-DS's.

An important feature of the mi-DS algorithm, in contrast to other MIL algorithms, is that it works well on data with missing values, which are treated as "do not care" values. To test this scenario, we introduced missing values into Musk1, Atoms, Bonds, Elephants, Fox, and Artificial 1 data sets. It was done by randomly deleting 5%, 10%, 15%, 20%, 25%, and 30% of the feature values. mi-DS was then run using 10FCV, and the results are shown graphically in Fig. 6. They indicate that mi-DS performs quite well up to 10% of missing values. This result could not be compared with those of other MIL algorithms because no published results were available at the time of this writing.

IV. CONCLUSION

We have introduced a new rule-based MIL algorithm, called mi-DS, and compared it with 21 MIL algorithms on 26 data sets that ranged from numerical, through text, to images. The results indicated that, although there did not exist one generally best performing algorithm on all data sets, the mi-DS performed on average quite well and had desirable characteristics that

TABLE XII
COMPARISON OF TIME NEEDED BY TEN MIL ALGORITHMS TO BUILD A MODEL

Time to build a model (sec)	Algorithm									
Data set	mi-DS	Citation-KNN	MI-DD	MDD	MI-EMDD	NND	MI-SMO	MI-SVM	MI-LR	MI-Opti-malBall
Musk1	1.7 [2]	0.88 [1]	171 [9]	120.5 [8]	225.2 [10]	34.7 [7]	7.22 [4]	31.3 [6]	15.4 [5]	2.4 [3]
Musk2	379.6 [7]	154.3 [5]	1917.9 [9]	5230 [10]	934 [8]	27.3 [1]	74.3 [4]	318 [6]	57.3 [3]	55.9 [2]
Atoms	1.9 [7]	0.8 [4]	12.5 [9]	5 [8]	0.8 [4]	13 [10]	0.5 [3]	0.4 [2]	0.2 [1]	1.4 [6]
Bonds	8.1 [2]	56.5 [5]	476.8 [9]	262.1 [8]	42.2 [4]	673 [10]	31.3 [3]	219.9 [7]	3.54 [1]	84.6 [6]
Chains	41.6 [5]	17.2 [3]	276 [9]	176.6 [8]	10 [2]	341 [10]	17.8 [4]	56.2 [6]	0.9 [1]	128 [7]
Elephants	57.7 [5]	9.6 [3]	554.7 [10]	346.6 [8]	292.5 [7]	515.4 [9]	26 [4]	146.4 [6]	8.1 [2]	3.4 [1]
Fox	45.4 [6]	13.9 [4]	263.9 [8]	470 [9]	199.5 [7]	483 [10]	4.6 [2]	17.1 [5]	1.1 [1]	7.7 [3]
Tiger	69.1 [6]	10.1 [3]	674.7 [7]	801.7 [9]	862.1 [10]	709 [8]	5.1 [2]	13.8 [5]	2.8 [1]	10.7 [4]
Artificial 1	8.1 [3]	7.4 [2]	1166 [10]	575.4 [9]	8.2 [4]	61.1 [7]	34.8 [6]	94.1 [8]	1.4 [1]	16.9 [5]
Artificial 2	8.8 [5]	2.8 [2]	491.9 [10]	170 [8]	170.2 [9]	105.6 [7]	3 [3]	33.9 [6]	0.5 [1]	3.2 [4]
Trx	1337 [8]	200.5 [5]	1681 [9]	1874 [10]	21.6 [3]	923 [7]	89.8 [4]	732 [6]	2.1 [1]	3.97 [2]
EastWest	0.18 [5]	0.06 [3]	3.86 [9]	7.66 [10]	0.52 [8]	0.28 [7]	0.03 [1]	0.17 [4]	0.2 [6]	0.03 [1]
WestEast	0.17 [7]	0.02 [1]	1.86 [9]	3.03 [10]	0.13 [6]	0.25 [8]	0.02 [1]	0.08 [5]	0.05 [4]	0.02 [1]
Africa	0.25 [4]	0.49 [6]	0.63 [7]	1.61 [9]	0.66 [8]	12.1 [10]	0.06 [1]	0.06 [1]	0.13 [3]	0.3 [5]
Beach	0.32 [3]	0.38 [6]	1.44 [8]	1.88 [9]	1.02 [7]	15.6 [10]	0.06 [2]	0.36 [5]	0.05 [1]	0.33 [4]
Building	0.25 [4]	0.49 [8]	0.44 [6]	0.48 [7]	0.33 [5]	56.2 [10]	0.06 [1]	45.99 [9]	0.16 [3]	0.11 [2]
Bus	0.4 [6]	0.39 [5]	0.98 [7]	1.03 [8]	1.27 [9]	8.45 [10]	0.03 [1]	0.13 [3]	0.05 [2]	0.25 [4]
Dinasaur	0.07 [4]	0.39 [7]	7.17 [8]	10.1 [9]	0.14 [6]	17.6 [10]	0.03 [1]	0.03 [1]	0.13 [5]	0.06 [3]
Component	154.75 [2]	1615 [6]	437 [3]	736 [5]	602 [4]	7562 [10]	5022 [8]	6258 [9]	8.34 [1]	1936 [7]
Function	32.8 [1]	3670 [7]	1682 [6]	952 [5]	381 [3]	9278 [10]	8576 [9]	404.9 [4]	51.6 [2]	5187 [8]
Process	189 [2]	17130 [8]	2047 [4]	2107 [5]	3620 [6]	53210 [10]	50381 [9]	982 [3]	39.9 [1]	4577 [7]
Avg. rank	**4.48**	**4.48**	**7.90**	**8.19**	**6.19**	**8.62**	**3.48**	**5.10**	**2.19**	**4.05**

TABLE XIII
COMPARISON OF ACCURACIES AND RUNNING TIMES ON EIGHT DATA SETS (MUSK2, ELEPHANTS, FOX, TIGER, TRX, COMPONENT, FUNCTION, AND PROCESS)

	mi-DS	Citation-KNN	MI-DD	MDD	MI-EMDD	MI-NND	MI-SMO	MI-SVM	MI-LR	MI-OptimalBall
Min. Accuracy (%)	64.5	63.8	65.5	70	59	58.5	58	49.4	56	53
Max. Accuracy (%)	97.1	87.6	92.2	96.8	90	93.8	91.4	87	93.8	97.1
Avg. Accuracy (%)	**81.5**	**77.7**	**80.5**	**83.8**	**76.3**	**77.1**	**81.8**	**73.2**	**78.7**	**78.5**
Min. modeling time (sec)	32.8	9.6	263.9	346.6	21.6	27.3	4.6	13.8	1.1	3.4
Max. modeling time (sec)	1337	17130	2047	5230	3620	53210	50381	6258	57.3	5187
Avg. modeling time (sec)	**283**	**2850**	**1157**	**1565**	**864**	**9088**	**8022**	**1109**	**21**	**1473**

distinguished it from other algorithms. First, it showed very good predictive accuracy on most data sets as measured by the average accuracy rank. In particular, it did well on text and preprocessed image data sets. Second, the differences between mi-DS and nine other algorithms were statistically significant for the six of them. Third, it performed quite well on data with missing values. Fourth, it is faster than most of the algorithms it was compared with. Importantly, the approach used in the mi-DS can be used as a generic framework for modifying other rule-based algorithms so that they can be used for solving MIL problems. This can be done in phase I of mi-DS as the rule generation process can be done by any rule learner, while the

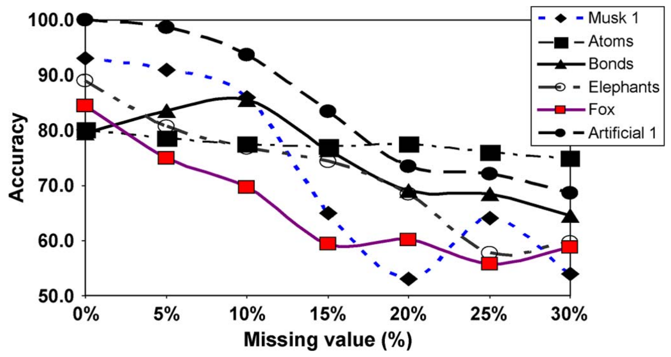

Fig. 6. Accuracy of mi-DS on data with missing values.

construction of a similarity matrix in phase II and the prediction procedure in phase III would remain unchanged.

References

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Proc. Neural Inf. Process. Syst.*, 2002, pp. 577–584.
[2] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*. Irvine, CA: School Inf. Comput. Sci., Univ. California, 2007.
[3] P. Auer and R. Ortner, "A boosting approach to multiple instance learning," in *Proc. 15th Eur. Conf. Mach. Learn.*, 2004, pp. 63–74.
[4] H. Blockeeel, D. Page, and A. Srinivasan, "Multi-instance tree learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, 2005, pp. 57–64.
[5] A. Blum and A. Kalai, "A note on learning from multi-instance examples," *Mach. Learn.*, vol. 30, no. 1, pp. 23–29, Jan. 1998.
[6] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[7] Y. Chen and J. Wang, "Image categorization by learning and reasoning with regions," *J. Mach. Learn. Res.*, no. 5, pp. 913–939, 2004.
[8] Y. Chen, J. Bi, and J. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 1931–1947, Dec. 2006.
[9] Y. Chevaleyre and J. D. Zucker, "Solving multi-instance and multi-part learning problems with decision trees and rule sets. Application to the mutagenesis problem," in *Lecture Notes in Artificial Intelligence*. Berlin: Springer-Verlag, 2001, pp. 204–214.
[10] Y. Chevaleyre and J. D. Zucker, "A framework for learning rules from multiple instance data," in *Proc. 12th Eur. Conf. Mach. Learn.*, 2001, pp. 49–60.
[11] J. Y. Chiang and S.-R. Cheng, "Multiple-instance content-based image retrieval employing isometric embedded similarity measure," *Pattern Recognit.*, vol. 42, no. 1, pp. 158–166, Jan. 2009.
[12] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[13] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multi-instance problem with axis-parallel rectangles," *Artif. Intell. J.*, vol. 89, no. 1/2, pp. 31–71, Jan. 1997.
[14] L. Dong, "A comparison of multi-instance learning algorithms," M.S. thesis, Univ. Waikato, Hamilton, New Zealand, 2006.
[15] G. Fung, M. Dundar, B. Krishnapuram, and R. B. Rao, "Multiple instance learning for computer aided diagnosis," in *Proc. NIPS*, 2007, pp. 425–432.
[16] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 179–186.
[17] P. Gehler and O. Chapelle, "Deterministic annealing for multiple-instance learning," in *Proc. AISTATS*, 2007, pp. 481–487.
[18] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Comput.*, vol. 13, no. 3, pp. 637–649, 2001.
[19] L. A. Kurgan, K. J. Cios, and S. Dick, "Highly scalable and robust rule learner: Performance evaluation and comparison," *IEEE Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 32–53, Feb. 2006.
[20] L. A. Kurgan and K. J. Cios, "CAIM discretization algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 2, pp. 145–153, Feb. 2004.
[21] C. Leistner, A. Saffari, and H. Bischof, "MIForests: Multiple-instance learning with randomized trees," in *Proc. 11th ECCV*, 2010, pp. 29–42.
[22] P. M. Long and L. Tan, "PAC-learning axis aligned rectangles with respect to product distributions from multiple-instance examples," in *Proc. Conf. Comput. Learn. Theory*, 1996, pp. 228–234.
[23] O. Maron and T. Lozano-Perez, "A framework for multiple-instance learning," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1998.
[24] O. L. Mangasarian and E. W. Wild, "Multiple instance classification via successive linear programming," *J. Optim. Theory Appl.*, vol. 137, no. 3, pp. 555–568, Jun. 2008.
[25] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[26] M. Mizianty, L. A. Kurgan, and M. Ogiela, "Discretization as the enabling technique for the naive Bayes and semi-naive Bayes based classification," *Knowl. Eng. Rev.*, vol. 25, no. 4, pp. 421–449, 2010.
[27] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," *J. Mach. Learn. Res.*, vol. 6, pp. 783–816, Dec. 2005.
[28] T. A. Pham and A. N. Jain, "Customizing scoring functions for docking," *J. Comput.-Aided Molecular Des.*, vol. 22, no. 5, pp. 269–286, 2008.
[29] J. Platt, "Machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1998.
[30] X. Qi and Y. Han, "Incorporating multiple SVMs for automatic image annotation," *Pattern Recognit.*, vol. 40, no. 2, pp. 728–741, Feb. 2007.
[31] A. Srinivasan, S. Muggleton, R. D. King, and M. J. E. Sternberg, "Mutagenesis: ILP experiments in a non-determinate biological domain," in *Proc. ILP*, 1994, pp. 217–232.
[32] J. Wang and J. Zucker, "Solving the multi-instance problem: A lazy learning approach," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 1119–1125.
[33] H. Y. Wang, Q. Yang, and H. Zha, "Adaptive p-posterior mixture-model kernels for multiple instance learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1136–1143.
[34] G. I. Webb and J. Agar, "Inducing diagnostic rules for glomerular disease with the DLG machine learning algorithm," *Artif. Intell. Med.*, vol. 4, no. 6, pp. 3–14, 1992.
[35] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA: Morgan Kaufmann, 2005.
[36] X. Xu and B. Li, "Multiple class multiple-instance learning for image categorization," *Lecture Notes in Computer Science*, vol. 4844, pp. 155–165, 2007.
[37] X. Xu and E. Frank, "Logistic regression and boosting for labeled bags of instances," in *Proc. Pacific Asia Conf. Knowl. Discov. Data Mining*, 2004, pp. 779–806.
[38] Y. Zhang, A. C. Surendran, J. C. Platt, and M. Narasimhan, "Learning from multi-topic web documents for contextual advertisement," in *Proc. ACM SIGKDD*, 2008, pp. 1051–1059.
[39] Q. Zhang and S. A. Goldman, "EM-DD: An improved multiple-instance learning technique," in *Proc. Neural Inf. Process. Syst.*, 2001, vol. 14, pp. 1073–1080.
[40] Z. H. Zhou and J. M. Xu, "On the relation between multi-instance learning and semi-supervised learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1167–1174.
[41] Z. H. Zhou, Y. Y. Sun, and Y. F. Li, "Multi-instance learning by treating instances as non-I.I.D. samples," in *Proc. ICML*, 2009, pp. 1249–1256.
[42] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm Support Vector Machines," in *Neural Information Processing*. Cambridge, MA: MIT Press, 2003, p. 16.

Dat T. Nguyen received the B.S. and M.S. degrees in computer science from the School of Science, Vietnam National University—Ho Chi Minh City, Ho Chi Minh City, Vietnam. He is currently working toward the Ph.D. degree in the Department of Computer Science, Virginia Commonwealth University, Richmond.

His research interests include machine learning, data mining, and visualization of high-dimensional data.

Cao D. Nguyen received the M.S. degree in computer science from Vietnam National University—Ho Chi Minh City, Ho Chi Minh City, Vietnam, and the Ph.D. degree in computer science and computational biology from the University of Colorado, Denver.

He is currently a Research Assistant Professor with The Western Australian Institute for Medical Research Centre for Diabetes Research, The University of Western Australia (UWA), Perth, Australia, where he is also with the Centre for Medical Research. Before joining UWA, he was a Postdoctoral Fellow with Virginia Commonwealth University, Richmond, where he worked on the National Institutes of Health-funded *Systems Biology for Studies of Cognition in Down Syndrome* project. His research interests are machine learning algorithms applied on genetic data of type-1 and type-2 diabetes, diabetic complications, and other complex diseases such as cancer and schizophrenia.

Rosalyn Hargraves received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Virginia, Charlottesville.

She is currently an Associate Professor of electrical and computer engineering with Virginia Commonwealth University, Richmond, where she is also the Associate Dean for Graduate Studies with the School of Engineering. Her research interests are biomedical signal and image processing, machine learning, K-20 Science Technology Engineering and Mathematics (STEM) education, and international development STEM activities.

Dr. Hargraves was a recipient of numerous fellowships, honors, and awards, including the Dominion Strong Men and Women Excellence in Leadership Award in 2008, the Richmond Joint Engineers Council Engineer of the Year in 2006, the American Association for the Advancement of Science Diplomacy Fellowship in 2003–2004, and the National Society of Black Engineers National Lumpkin Educator of the Year Award in 2001.

Lukasz A. Kurgan (M'02) received the M.S. degree (with honors) in automation and robotics from the AGH University of Science and Technology, Krakow, Poland, in 1999 and the Ph.D. degree in computer science from the University of Colorado, Boulder, in 2003.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include applications of machine learning in structural bioinformatics. He authored and coauthored several machine learning algorithms and methods for high-throughput prediction of protein and short ribonucleic acid structure and function. He published close to 80 peer-reviewed journal articles. He currently serves as an Editor of *PLoS ONE*, *BMC Bioinformatics*, *Neurocomputing*, *Journal of Biomedical Science and Engineering*, and *Protein and Peptide Letters*.

Dr. Kurgan has been a member of numerous conference committees in the area of bioinformatics, data mining, machine learning, and computational intelligence.

Krzysztof J. Cios (SM'90) received the M.S. and Ph.D. degrees from the AGH University of Science and Technology, Krakow, Poland, the D.Sc. degree from the Polish Academy of Sciences, Warsaw, Poland, and the M.B.A. degree from The University of Toledo, Toledo, OH.

He has worked at universities in Ohio and Colorado. He is currently a Professor with Virginia Commonwealth University, Richmond, where he is also the Chair of the Computer Science Department. His research interests are in the areas of machine learning, computational neuroscience, data mining, and bioinformatics. His research has been funded by the National Institutes of Health, the National Aeronautics and Space Administration, the National Science Foundation, the North Atlantic Treaty Organization, and the U.S. Air Force. He published three books and about 200 journal and conference papers. He serves on several journals' editorial boards.

Prof. Cios was a recipient of the Norbert Wiener Outstanding Paper Award, the Neurocomputing Best Paper Award, and the Fulbright Senior Scholar Award. He has been elected a Foreign Member of the Polish Academy of Arts and Sciences.