MICCLLR: Multiple-Instance Learning using Class Conditional Log Likelihood Ratio

Yasser EL-Manzalawy[1,2] and Vasant Honavar[1]

[1]Department of Computer Science
Iowa State University
Ames, IA 50011-1040, USA
[2] Systems and Computer Engineering
Al-Azhar University
Cairo, Egypt
{yasser,honavar}@cs.iastate.edu

Abstract. *Multiple-instance learning (MIL) is a generalization of the supervised learning problem where each training observation is a labeled bag of unlabeled instances. Several supervised learning algorithms have been successfully adapted for the multiple-instance learning settings. We explore the adaptation of the Naive Bayes (NB) classifier and the utilization of its sufficient statistics for developing novel multiple-instance learning methods. Specifically, we introduce MICCLLR (multiple-instance class conditional log likelihood ratio), a method for mapping each bag of instances as a single meta-instance using class conditional log likelihood ratio statistics such that any supervised base classifier can be applied to the meta-data. The results of our experiments with MICCLLR using different base classifiers suggest that no single base classifier consistently outperforms other base classifiers on all data sets. We show that a substantial improvement in performance is obtained using an ensemble of MICCLLR classifiers trained using different base learners. We also show that an extra gain in classification accuracy is obtained by applying AdaBoost.M1 to weak MICCLLR classifiers. Overall, our results suggest that the predictive performance of the three proposed variants of MICCLLR are competitive to some of the state-of-the-art MIL methods.*

Key words: multiple-instance learning, image retrieval, drug activity prediction, ensemble of multiple-instance learning classifiers, boosted multiple-instance learning

1 Introduction

Dietterich et al. [1] introduced the multiple-instance learning (MIL) problem motivated by his work on classifying aromatic molecules according to whether or not they are "musky". In this classification task, each molecule can adopt multiple shapes as a consequence of rotation of some internal bonds. Dietterich et al. [1] suggested representing each molecule by multiple conformations (instances) representing possible shapes or conformations that the molecule can assume. The multiple conformations yield a multiset (bag) of instances (where each instance corresponds to a conformation) and the task of the classifier is to assign a class label to such a bag. Dietterich's proposed solution to the MIL problem is based on the standard multiple-instance assumption, that all the instances in a bag, in order for it be labeled negative, must contain no positively labeled instance, and a positive bag must have at least one positive instance. The resulting classification task finds application in drug discovery [1], identifying Thioredoxin-fold proteins [2], content-based image retrieval (CBIR) [3–5], and computer aided diagnosis (CAD) [6].

Several approaches to MIL have been investigated in the literature including a MIL variant of the backpropagation algorithm [7], variants of the k-nearest neighbor (k-NN) algorithm [8], the Diverse Density (DD) method [9] and EM-DD [10] which improves on DD by using Expectation Maximization (EM), DD-SVM [11] which trains a support vector machine (SVM) classifier in a feature space constructed from a mapping defined by the local maximizers and minimizers of the DD function, and MI logistic regression (MILR) [12]. Most of these methods search for a single instance contributing the positive bag label. Alternatively, a number of MIL methods [13–15] have a generalized view of the MIL problem where all the instances in a bag are assumed to participate in determining the bag label.

Two basic approaches for solving the MIL problem have been proposed in the literature: i) adapting supervised learning algorithms for MIL settings. Zhou [16] showed that standard single-instance supervised algorithms can be adapted for MI learning by shifting their focuses from discrimination on the instances to the discrimination on the bags. Many MIL methods can be viewed as single-instance learning methods adapted for the MIL settings. For example, MI-SVM [17], Citation-kNN [8], DD [9], and RBF-MIP [18]; ii) adapting MIL representation for supervised algorithms. The basic idea is to convert each bag of instances into a single feature vector such that supervised classifiers can be trained to discriminate between positive and negative bags by discriminating between their corresponding feature vectors. Several techniques for mapping bags into single feature vectors are discussed in the next section.

Naive Bayes (NB) has proven effective in many practical applications, including text classification, medical diagnosis, and systems performance management [19–21]. In this work, we showed that NB classifier can be adapted for MIL setting. However, this adaptation imposes strong and unrealistic independence assumptions (instances within a bag are independent given the bag label and instance attributes are independent given the label of the instance). Alternatively, we propose MICCLLR, a generalized MIL algorithm that uses the class conditional log likelihood ratio statistics to map each bag into a single meta-instance. MICCLLR allows for any supervised learning algorithm to be the base classifier for classifying the meta-instance data. Our results evaluating MICCLLR using different base classifiers suggest that no single base classifier consistently outperforms other base classifiers on all data sets. Consequently, we show that a substantial improvement in performance is obtained using an ensemble of MIC-

CLLR classifiers trained using different base learners. Additional gain in classification accuracy is obtained by applying AdaBoost.M1 [22] to weak MICCLLR learners. Overall, our results suggest that the predictive performance of the three proposed variants of MICCLLR are competitive to some of the state-of-the-art MIL methods on five widely used MIL data sets for drug activity prediction [1] and image retrieval [17] domains.

The rest of this paper is organized as follows: Section 2 summarizes the formulations of the MIL problem and overviews a number of related MIL methods that follow the same approach of adapting MIL representation for single-instance learning algorithms. Section 3 introduces our method. Section 4 gives our experimental results. Section 5 concludes with a brief summary and discussion.

2 Preliminaries

2.1 Multiple-instance learning problem.

In the standard (single-instance) supervised classifier learning scenario, each instance (input to the classifier) is typically represented by an ordered tuple of attribute values. The instance space $I = D_1 \times D_2 \times ... \times D_n$ where D_i is the domain of the i^{th} attribute. The output of the classifier is a class label drawn from a set C of mutually exclusive classes. A training example is a labeled instance in the form $\langle X_i, c(X_i)\rangle$ where $X_i \in I$ and $c : I \rightarrow C$ is unknown function that assigns to an instance X_i its corresponding class label $c(X_i)$. For simplicity we consider only the binary classification problem in which $C = \{-1, 1\}$. Given a collection of training examples, $E = \{\langle X_1, c(X_1)\rangle, \langle X_2, c(X_2)\rangle, ..., \langle X_n, c(X_n)\rangle\}$, the goal of the (single-instance) learner is to learn a function c^* that approximates c as well as possible.

In the multiple-instance supervised classifier learning scenario, the goal is to train a (multiple-instance) classifier to label a *bag* of instances. Under standard MIL assumption, a bag is labeled negative if and only if all of its instances are negatively labeled and a bag is labeled positive if at least one of its instances is labeled positive. More precisely, Let B_i denotes the i^{th} bag in a set of bags B. Let $X_{ij} \in I$ denotes the j^{th} instance in the bag B_i and X_{ijk} be the value of the k^{th} feature in the instance X_{ij}. The set of MI training examples, E_{MI}, is a collection of ordered pairs $\langle B_i, f(B_i)\rangle$ where f is unknown function that assigns to each bag B_i a class label $f(B_i) \in \{-1, 1\}$. Under the standard MIL assumption [1], $f(B_i) = -1$ iff $\forall_{X_{ij} \in B_i} c(X_{ij}) = -1$; and $f(B_i) = 1$ iff $\exists_{X_{ij} \in B_i}$, such that $c(X_{ij}) = 1$. Given E_{MI}, a collection of MI training examples, the goal of the multiple-instance learner is to learn a good approximation function of f. It should be noted that the function f is defined in terms of a function $c : I \rightarrow C$. However, learning c from the MI training data is challenging since we have labels only associated with bags and we do not have labels for each instance.

A generalization of the MIL problem has been considered by Weidmann et al. [13] and Tao et al. [14]. In this setting, all the bag instances contribute the label assigned to the bag and negative bags may contain some positive instances.

Instead of a single concept, the generalized MIL problem considers a set of underlying concepts and requires a positive bag to have a certain number of instances in each of them.

2.2 Adapting MI representation for single-instance learning algorithms.

A number of existing methods for solving the MIL are based on the idea of adapting the MI representation for single-instance learning algorithms. Gartner et al. [23] mapped each bag into a single meta-instance using an aggregation function (e.g. mean, median, minimum, maximum, etc.) applied to each instance attribute. The resulting labeled meta-instances data set is then used to train an SVM classifier. Weidmann et al. [13] proposed a two-level classifier (TLC) trained from the data at two different levels of abstraction. The first classifier is trained from the MI data at the instance level by assigning the label of each bag to its instances and assigning a weight to each instance such that bags of different size will end up with the same weight. Then, the trained classifier is used to map MI data into a set of meta-instances and a second level supervised classifier is trained. In their experiments, Weidmann et al, [13] used a pruned decision tree and a Logit-boosted decision stumps (DS)[24] with 10 boosting iterations as the first and second level classifier, respectively.

Chen et al. [11] mapped each bag into a meta-instance in a feature space defined by a set of instance prototypes. An instance prototype is an instance that is close as possible to at least one instance in each positive bag and as far as possible from instances in negative bags. The algorithm, named DD-SVM, proceeds in two steps. First a collection of instance prototypes are learned such that each prototype is a local maximizer of the DD function. Second, each bag is mapped into a feature vector where the i^{th} feature is defined by the minimum distance between the i^{th} prototype and each instance in the bag. Finally, a standard SVM classifier is trained in the new feature space.

Recently, Chen et al. [15] proposed multiple-instance learning via embedded instance selection (MILES) which can be viewed as a variant of DD-SVM where the new feature space is defined by the set of all training instances instead of the set of prototypes used with DD-SVM. This feature mapping often provides a large number of irrelevant features. Therefore, 1-norm SVM is applied to select important features and construct classifiers simultaneously.

Zhou and Zhang [25] proposed constructive clustering-based ensemble (CCE) method where all the training instances are clustered into d groups. Then, each bag is mapped into a d binary vector, where the value of the i^{th} feature is set to one if the concerned bag has instances falling into the i^{th} group and zero otherwise. The above procedure is repeated for different values of d. For each value of d, a meta-instance representation of each bag is generated and an SVM classifier is trained. All the classifiers are then combined in an ensemble for prediction.

3 The Algorithm

We motivate our method by first introducing MI Naive Bayes (MINB), an adaptation of Naive Bayes (NB) classifier to MIL settings. The NB classification rule is defined by Eq. 1, where $Pr(c_j)$ is the a priori probability of class c_j and $Pr(a_k|c_j)$ is the probability that the k^{th} attribute of the instance X takes the value a_k given the class c_j.

$$c(X) = arg \max_{c_j \in C} Pr(c_j) \prod_{k=1}^{n} Pr(a_k|c_j) \tag{1}$$

These probabilities, which completely specify a NB classifier, can be estimated from the training data using standard probability estimation methods based on relative frequencies of the corresponding classes and attribute value and class label cooccurrences observed in the data [20]. These relative frequencies summarize all the information relevant for constructing a NB classifier from a training data set, and hence constitute sufficient statistics for NB Classifier.

When the class labels are binary, that is, $C = \{-1, 1\}$, the NB classifier can be viewed as a linear discriminant by considering the logarithm of posterior odds as defined by Equations 2 and 3.

$$\phi(X) = ln\frac{Pr(c=1)}{Pr(c=-1)} + ln\frac{Pr(a_1|c=1)}{Pr(a_1|c=-1)} + \ldots + ln\frac{Pr(a_n|c=1)}{Pr(a_n|c=-1)} \tag{2}$$

$$c(X) = \begin{cases} 1 \, , \, \phi(X) > 0 \\ -1 \, , otherwise \end{cases} \tag{3}$$

Similarly, given unlabeled bag B_i with m_i instances, MINB assigns a label to B_i as follows:

$$\begin{aligned} c(B_i) &= arg \max_{c_j \in C} Pr(c_j|B_i) \\ &= arg \max_{c_j \in C} Pr(B_i|c_j)Pr(c_j) \\ &= arg \max_{c_j \in C} Pr(X_{i1}, X_{i2}, \ldots, X_{im_i}|c_j)Pr(c_j) \\ &= arg \max_{c_j \in C} Pr(c_j) \textstyle\prod_{l=1}^{m_i} Pr(X_{il}|c_j) \end{aligned} \tag{4}$$

The prior probabilities of labels, $Pr(c_j)$, can be easily estimated by counting the number of negative and positive bags. Recalling that instances within a bag are not labeled, estimating $Pr(X_{il}|c_j)$ is not trivial. In order to approximate $Pr(X_{il}|c_j)$, we first need to assign a label to each instance. Then, assuming independence between attributes given the instance class dramatically simplifies the computation of $Pr(X_{il}|c_j)$. That is, $Pr(X_{il}|c_j) = \prod_k Pr(X_{ilk}|c_j)$. Following the approach in [13], we construct a single instance training data set from the set of all instances contained in all bags, labeled with their bag's class label. Instances in a bag B_i are assigned a weight equal $\frac{1}{|B_i|} \cdot \frac{N}{M}$, where $N = \sum_i^m |B_i|$ and M denotes the number of bags in the training data set.

Based on these assumptions, the MINB classification rule can be defined as in Eq. 5, where $X_{ilk} = a_k$ denotes the value of the k^{th} attribute in the l^{th} instance in bag B_i with m_i instances where each instance is represented by an ordered tuple of n attribute values.

$$c(B_i) = arg \max_{c_j \in C} Pr(c_j) \prod_{l=1}^{m_i} \prod_{k=1}^{n} Pr(X_{ilk}|c_j) \tag{5}$$

Alternatively, we can rewrite the MINB classifier as a linear discriminant:

$$\phi(B_i) = ln \frac{Pr(c=1)}{Pr(c=-1)} + ln \frac{Pr(X_{11}|c=1)}{Pr(X_{i1}|c=-1)} + \ldots + ln \frac{Pr(X_{im_i}|c=1)}{Pr(X_{im_i}|c=-1)} \tag{6}$$

$$c(B_i) = \begin{cases} 1 \, , \, \phi(B_i) > 0 \\ -1 \, , otherwise \end{cases} \tag{7}$$

Unfortunately, MINB has strong independence assumptions and its observed cross-validation performance on Musk data sets [1] is not competitive with the performance of the state-of-the art MIL methods (See Table 1). Instead of adapting NB for MIL setting, we propose to use NB to map the MI representation into a single meta-instance representation such that any standard supervised classification algorithm is applicable.

We now proceed to describe, MICCLLR, a MIL algorithm that uses class conditional log likelihood ratio (CCLLR) statistics estimated from the MI training data to map each bag into a single meta-instance. The pseudo code for MICCLLR is described in Algorithm 1. The input to the algorithm is a set of binary labeled bags E_{MI} and a base learner h. First, MICCLLR assigns the label of each bag to its instances and associate a weight with each instance to compensate for the fact that different bags may be of different sizes (i.e, different number of instances). Second, MICCLLR estimates the probability of each possible value for each attribute given the instance label. Under Naive Bayes assumption, the posterior probability of each attribute is independent of other attributes given the instance label. Therefore, the posterior probability of each attribute can be easily estimated from the training data using standard probability methods based on relative frequencies of each attribute value and class label occurrences observed in the labeled training instances [20]. Third, the algorithm uses the collected statistics to map each bag into a single meta-instance. Let $B_i = \{X_{i1}, \ldots, X_{im_i}\}$ be a bag of m_i instances. Each instance is represented by an ordered tuple of n attribute values. We define a function s that maps B_i into a single meta-instance of n real value attributes as; $s(B_i) = \{s_1, s_2, \ldots, s_n\}$ where each meta-instance attribute is computed using Eq. 8.

$$s_q = \frac{1}{m_i} ln \sum_{l=1}^{m_i} \frac{Pr(X_{ilq} = a_q|c=1)}{Pr(X_{ilq} = a_q)|c=-1)} \tag{8}$$

Once bags in a multiple-instance data set have been transformed into meta-instances, the base learner h is trained on the transformed data set of labeled meta-instances. During the classification phase, each bag to be classified is first transformed into a meta-instance in a similar fashion before being fed to the base classifier h.

Algorithm 1 Training MICCLLR

```
1: Input : $E_{MI} = \{\langle B_1, y_1\rangle, \dots \langle B_m, y_m\rangle\}$ set of training bags and $h$ base learner
2: Use $E_{MI}$ to construct the collection of all instances $E_{AV}$ by labeling each instance
   with its bag's class label and assign to instances in a bag $B_i$ a weight equal to
   $\frac{1}{|B_i|} \cdot \frac{N}{M}$ , where $N = \sum_i |B_i|$ and $M$ denotes the number of bags in the training
   data set.
3: Estimate the posterior probabilities of each attribute, $Pr(a_q|c_j)$, from $E_{AV}$.
4: Convert each bag in $E_{MI}$ to a single meta-instance $\{s_1, s_2, ..., s_n\}$ using Eq. 8.
5: Train the base learner $h$ using the meta-instance data.
```

4 Experiments and Results

In our experiments, we implemented MINB and MICCLLR using Java and WEKA API [26]. The rest of classification algorithms considered in our experiments were used as implemented in WEKA. The default parameters for all WEKA classifiers were used unless otherwise specified. As a measure of the predictive performance of the MIL algorithms, we used the classification accuracy obtained by averaging the results of 10 different runs of 10-fold cross-validation tests. We conducted our experiments using five widely used MIL data sets from drug activity prediction [1] and content-based image retrieval (CBIR) [17] application domains.

Recently, Demšar [27] has suggested that non-parametric tests should be preferred over parametric tests for comparing machine learning algorithms because the non-parametric tests, unlike parametric tests, do not assume normal distribution of the samples (e.g., the data sets). Demšar suggested a three-step procedure for performing multiple hypothesis comparisons using non-parametric tests. Unfortunately, this procedure can not be applied to our experimental results because it requires the number of data sets to be greater than 10 and the number of methods to be greater than 5 [27]. However, as noted by Demšar [27], the average ranks by themselves provide a reasonably fair comparison of classifiers. Hence, the classifiers being compared are ranked on the basis of their observed performance on each data set. Then, the average rank of each classifier on all data sets is used to compare the overall performance of different MIL methods.

4.1 Comparison of base learners for MICCLLR

As mentioned above, MICCLLR uses class conditional log likelihood ratio statistics collected from the training data for mapping each bag of instances into a single-meta instance such any supervised base classifier becomes applicable. In our experiments, we evaluated MICLLR using a representative set of base classifiers. Specifically, we used Logistic Regression (LR) [28], C4.5 [29], Alternating Decision Trees (ADTree) [30], and 2-norm SVM (SMO) [31] classifiers SVML, SVMP, and SVMR evaluated using three kernels (linear, puk [32], and radial-bias function (RBF) kernel) (respectively) as base classifiers for MICCLLR. Table 1 compares the classification accuracy of MINB and six MICCLLR classifiers evaluated using different base learners on Musk and CBIR data sets. Interestingly, MICCLLR classifiers with SVML, SVMP, ADTree, and J48 as base learners have better average ranks than MINB. The results also suggest that MICCLLR performance seems to be sensitive to the choice of the base classifier. However, none of the base classifiers produces a MICCLLR classifier with consistently superior performance on the five data sets. An ensemble of the five reported MICCLLR classifiers developed using WEKA implementation of majority voting, Vote classifier, outperforms any individual classifier on three out of five data sets. The ensemble of MICCLLR classifiers has the best average rank (1.6) followed by MICCLLR classifier using SVML and SVMP with average ranks 3.2 and 3.6, respectively. The predictive performance of the ensemble of MICCLLR classifiers, `MICCLLR_Vote`, could be further improved by: i) adding more MICCLLR classifiers utilizing other base learners to the ensemble; ii) using more sophisticated methods for constructing the ensemble (e.g., stacking [33]).

Table 1. Comparisons of prediction accuracy of MINB with five MICCLLR classifiers evaluated using different base learners on Musk and CBIR data sets. Last row is the performance of an ensemble of the five MICCLLR classifiers constructed using WEKA's Vote method. For each data set, the rank of each classifier is shown in parentheses. Last column is the average performance and rank for each method over the five data sets.

Method	musk1	musk2	elephant	fox	tiger	avg.
MINB	77.06(8)	77.90(6)	81.70(1.5)	56.80(3)	72.00(6)	72.75(4.9)
LR	80.86(7)	85.28(2)	74.35(6.5)	55.8(5)	67.45(8)	72.75(5.7)
J48	87.71(3)	72.05(8)	74.35(6.5)	60.25(1)	73.2(5)	73.51(4.7)
ADTree	84.89(5)	75.35(7)	75.05(5)	59.25(2)	76.85(3)	74.28(4.4)
SVML	86.02(4)	82.20(3)	80.65(3)	52.10(7)	79.10(1)	76.01(3.6)
SVMP	88.39(2)	81.55(4)	75.90(4)	55.00(6)	76.50(4)	75.47(4)
SVMP	82.40(6)	80.53(5)	70.10(8)	50.25(8)	70.25(7)	70.71(6.8)
Vote	91.64(1)	86.12(1)	81.70(1.5)	56.15(4)	78.50(2)	78.82(1.9)

4.2 Boosting weak MICCLLR classifiers

Several methods for adapting boosting algorithms for the MIL settings have been proposed in the literature [34–37]. Xu and Frank [36] have noted that supervised learning boosting algorithms (e.g., AdaBoost.M1 [22]) can be applied directly to weak MIL learners. However, they did not compare their proposed MI boosting method with this basic approach. Here, we explored the utility of directly applying AdaBoost.M1 to MICCLLR and MIWrapper [38] weak learners. We compared the performance of the two MIL boosting algorithms implemented in WEKA, MIBoost [36] and MIOpimalBall [35], with AdaBoosted MICCLLR and MIWrapper classifiers. For MIOptimalBall, the weak learner constructs a ball such that at least one instance from each positive bag is included in the ball and all negative instances lie outside the ball. For MIBoost, we used decision stumps (DS) [24] and C4.5 as the week learners with 25 iterations. For AdaBoost.M1, MICCLLR and MIWrapper weak learners were obtained using DS and C4.5 as the base classifiers and the number of boosting iterations was set to 25.

Table 2 shows that boosted MIWrapper and boosted MICCLLR with C4.5 have the best average ranks of 2 and 2.6, respectively. The results show that boosted MIWrapper and boosted MICCLLR classifiers are competitive with (if not outperforming) the two MIL boosting methods, MIOptimalBall and MIBoost. Interestingly, we observed that MIBoost, boosted MIWrapper, and boosted MICCLLR with C4.5 as the weak learner generally outperform their counterpart classifiers with DS as the weak learner. Among the classifiers using C4.5 as the weak learner, boosted MIWrapper has the best average rank while boosted MICCLLR has the best average rank if we limit our comparison to methods with DS as the weak learner.

Table 2. Comparisons of classification accuracy of two MIBoosting algorithms (MIOptimalBall and MIBoost) with boosted MIWrapper and boosted MICCLLR. For MIOptimallBall the weak learner is a ball while for other methods C4.5 and DS were used as weak learners.

Data	MIOptimalBall	MIBoost		boosted MIWrapper		boosted MICCLLR	
		C4.5	DS	C4.5	DS	C4.5	DS
musk1	70.37(7)	84.07(3)	76.98(6)	85.53(2)	78.13(5)	88.57(1)	82.49(4)
musk2	80.80(2)	80.55(3)	74.68(6)	81.61(1)	72.53(7)	79.18(4)	78.03(5)
elephant	72.00(7)	82.05(3)	81.65(4)	86.15(1)	80.75(5)	82.8(2)	78.5(6)
fox	54.60(7)	64.85(1)	62.95(2)	62.8(3)	62.65(4)	62.15(5)	58.8(6)
tiger	65.15(7)	78.95(6)	80.25(4)	81.2(3)	79.25(5)	82.5(1)	81.6(2)
Avg.	68.58(6)	78.09(3.2)	75.3(4.4)	79.46(2)	74.66(5.2)	79.04(2.6)	75.88(4.6)

4.3 Comparison of MICCLLR to other MIL methods

The classification accuracy of the best performing three MICCLLR classifiers, MICCLLR with SVM base learner trained using linear kernel (`MICCLLR_SVML`), ensemble of MICCLLR classifiers (`MICCLLR_Vote`), and `boosted MICCLLR_C4.5` was compared to existing MIL with reported performance on the five data sets considered in this study (See Table 3). The average ranks for the three MICCLLR classifiers are `boosted MICCLLR_C4.5` (4), `MICCLLR_Vote` (5.4), and

`MICCLLR_SVML` (8.4). Hence, `boosted MICCLLR_C4.5` improves the predictive performance over the majority vote ensemble of MICCLLR classifiers and the single MICCLLR classifier with SVM as the base learner. The results suggest that `boosted MICCLLR_C4.5` is also competitive in performance with the state-of-the-art MIL methods on Musk and CBIR data sets. The best performing three methods, as measured by the average rank of the classifier, are CH-FD [6], RW-SVM [39], and `boosted MICCLLR_C4.5` with average ranks 3.4, 3.6, and 4, respectively.

Table 3. Comparison of the classification accuracy of three MICCLLR classifiers with different MIL methods on Musk and CBIR data sets.

Method	musk1	musk2	elephant	fox	tiger	avg.
EM-DD [17]	84.50(9)	84.90(6)	78.30(10)	56.10(9)	72.10(10)	75.18(8.8)
mi-SVM [17]	87.40(7)	83.60(8)	82.20(4)	58.20(6)	78.9(8)	78.06(6.6)
MI-SVM [17]	77.90(10)	84.30(7)	81.40(7)	59.40(4)	84.00(1)	77.40(5.8)
MICA [40]	88.40(5)	90.50(1)	80.50(9)	58.70(5)	82.60(2)	80.14(4.4)
CH-FD [6]	88.80(3)	85.70(5)	82.40(3)	60.40(2)	82.20(4)	79.90(3.4)
I-DD [41]	90.80(2)	86.40(3)	81.50(6)	57.30(7)	80.70(5)	79.34(4.6)
RW-SVM [39]	87.60(6)	87.10(2)	83.30(1)	60.00(3)	79.50(6)	79.50(3.6)
`MICCLLR_SVML`	86.02(8)	82.20(9)	80.65(8)	52.10(10)	79.10(7)	76.01(8.4)
`MICCLLR_Vote`	91.64(1)	86.12(4)	81.70(5)	56.15(8)	78.50(9)	78.82(5.4)
boosted `MICCLLR_C4.5`	88.57(4)	79.18(10)	82.80(2)	62.15(1)	82.50(3)	79.04(4.0)

5 Conclusions

We introduced MINB, an adaptation of Naive Bayes for the MIL settings. We showed that the proposed MINB algorithm imposes strong and unrealistic independence assumptions (instances within a bag are independent given the bag label and instance attributes are independent given the label of the instance). We empirically showed that class conditional log likelihood ratio statistics estimated from the training data provide useful single feature representation of bags that allows the applicability of standard supervised learning methods (base learners) for predicting labels of MIL bags given their single feature vector representation as an input. The performance of our proposed method, MICCLLR, has been evaluated using different base learners. Moreover, we empirically showed that further improvements in MICCLLR performance is obtained using ensemble of MICCLLR classifiers utilizing different base learners. Finally, we demonstrated that an additional gain in classification accuracy is obtained when AdaBoost.M1 is applied directly to weak MIL learner derived from MIWrapper and MICCLLR using C4.5 as the base learner. Our results suggest that integrating AdaBoost.M1 with MIWrapper and MICCLLR weak learners is a promising approach for developing MIL methods with improved prediction performance.

References

1. Dietterich, T., Lathrop, R., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. 89, 31–71 (1997)
2. Wang, C., Scott, S., Zhang, J., Tao, Q., Fomenko, D., Gladyshev, V.: A Study in Modeling Low-Conservation Protein Superfamilies. Technical Report TR-UNL-CSE-2004-3, Dept. of Computer Science, University of Nebraska (2004)
3. Maron, O., Ratan, A.: Multiple-instance learning for natural scene classification. In: Proceedings of the Fifteenth International Conference on Machine Learning table of contents, pp. 341–349. (1998)
4. Zhang, Q., Goldman, S., Yu, W., Fritts, J.: Content-based image retrieval using multiple-instance learning. Proceedings of the Nineteenth International Conference on Machine Learning pp. 682–689. (2002)
5. Bi, J., Chen, Y., Wang, J.: A sparse support vector machine approach to region-based image categorization. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 1121–1128. (2005)
6. Fung, G., Dundar, M., Krishnapuram, B., Rao, R.: Multiple instance learning for computer aided diagnosis. In: Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference, pp. 425-432. MIT Press (2007)
7. Ramon, J., De Raedt, L.: Multi instance neural networks. Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning (2000)
8. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: a lazy learning approach. In: Proceedings 17th International Conference on Machine Learning. pp. 1119–1125. (2000)
9. Maron, O., Lozano-Perez, T.: A framework for multiple-instance learning. Adv. Neural. Inf. Process. Syst. 10, 570–576 (1998)
10. Zhang, Q., Goldman, S.A.: Em-dd: An improved multiple-instance learning technique. Neural. Inf. Process. Syst. 14, (2001)
11. Chen, Y., Wang, J.: Image categorization by learning and reasoning with regions. J. Mach. Learn. Res. 5, 913–939 (2004)
12. Ray, S., Craven, M.: Supervised versus multiple instance learning: An empirical comparison. In: Proceedings of the Twentieth-Second International Conference on Machine Learning. pp. 697–704 (2005)
13. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Proceedings of the European Conference on Machine Learning, pp. 468–479, Springer (2003)
14. Tao, Q., Scott, S., Vinodchandran, N., Osugi, T.: SVM-based generalized multiple-instance learning via approximate box counting. In: Proceedings of the twenty-first international conference on Machine learning, ACM New York, NY, USA (2004)
15. Chen, Y., Bi, J., Wang, J.: MILES: multiple-instance learning via embedded instance selection. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1931–1947 (2006)
16. Zhou, Z.: Multi-instance learning from supervised view. Journal of Computer Science and Technology 21, 800–809 (2006)
17. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Adv. Neural. Inf. Process. Syst. 15, pp. 561–568, Cambridge, MA:MIT Press (2003)
18. Zhang, M., Zhou, Z.: Adapting RBF neural networks to multi-instance learning. Neural Process. Lett. 23, 1–26 (2006)
19. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine learning 29, 103–130 (1997)

20. Mitchell, T.: Machine Learning. McGraw Hill (1997)
21. Hellerstein, J., Jayram, T., Rish, I.: Recognizing End-User Transactions in Performance Management. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 596–602. AAAI Press/The MIT Press (2000)
22. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proceedings of 13th International Conference in Machine Learning. pp. 148–156. (1996)
23. Gartner, T., Flach, P., Kowalczyk, A., Smola, A.: Multi-instance kernels. Proceedings of the 19th International Conference on Machine Learning pp. 179–186. (2002)
24. Friedman, J., Hastie, T., Tibshirani, R.: Special invited paper. additive logistic regression: A statistical view of boosting. Annals of statistics 337–374 (2000)
25. Zhou, Z., Zhang, M.: Solving multi-instance problems with classifier ensemble based on constructive clustering. Knowl. Inf. Syst. 11, 155–170 (2007)
26. Witten, I., Frank, E.: Data mining: Practical machine learning tools and techniques. 2nd edn. Morgan Kaufmann (2005)
27. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
28. Le Cessie, S., Van Houwelingrn, J.: Ridge estimators in logistic regression. Appl. Stat. 41, 191–201 (1992)
29. Quinlan, J.: C4. 5: programs for machine learning. Morgan Kaufmann (1993)
30. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Proceedings of the Sixteenth International Conference on Machine Learning table of contents, pp. 124–133. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1999)
31. Platt, J.: Fast training of support vector machines using sequential minimal optimization. MIT Press (1998)
32. Ustün, B., Melssen, W., Buydens, L.: Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems 81, 29–40 (2006)
33. Wolpert, D.: Stacked generalization. Neural Networks 5, 241–259 (1992)
34. Andrews, S., Hofmann, T.: Multiple instance learning via disjunctive programming boosting. In: 2003 Conference on Advances in Neural Information Processing Systems, pp. 65–72. Bradford Book (2004)
35. Auer, P., Ortner, R.: A Boosting Approach to Multiple Instance Learning. In: 15th European Conference on Machine Learning, pp. 63–74. Springer (2004)
36. Xu, X., Frank, E.: Logistic regression and boosting for labeled bags of instances. In: 8th Pacific-Asia Conference, PAKDD, pp. 26–28. Springer (2004)
37. Viola, P., Platt, J., Zhang, C.: Multiple Instance Boosting for Object Detection. In: Advances in Neural Information Processing Systems. pp. 1417–1424. (2006)
38. Frank, E., Xu, X.: Applying propositional learning algorithms to multi-instance data. Technical report, Dept. of Computer Science, University of Waikato (2003)
39. Wang, D., Li, J., Zhang, B.: Multiple-instance learning via random walk. Lecture Notes in Computer Science 4212, 473 (2006)
40. Mangasarian, O., Wild, E.: Multiple instance classification via successive linear programming. Data Mining Institute Technical Report 05-02 (2005)
41. Han, F., Wang, D., Liao, X.: An Improved Multiple-Instance Learning Algorithm. In: 4th international symposium on Neural Networks, pp. 1104–1109. Springer-Verlag Berlin, Heidelberg (2007)