MILES: Multiple-Instance Learning via Embedded Instance Selection

Yixin Chen, Member, IEEE, Jinbo Bi, Member, IEEE, and James Z. Wang, Senior Member, IEEE

Abstract—Multiple-instance problems arise from the situations where training class labels are attached to sets of samples (named bags), instead of individual samples within each bag (called instances). Most previous multiple-instance learning (MIL) algorithms are developed based on the assumption that a bag is positive if and only if at least one of its instances is positive. Although the assumption works well in a drug activity prediction problem, it is rather restrictive for other applications, especially those in the computer vision area. We propose a learning method, MILES (Multiple-Instance Learning via Embedded instance Selection), which converts the multiple-instance learning problem to a standard supervised learning problem that does not impose the assumption relating instance labels to bag labels. MILES maps each bag into a feature space defined by the instances in the training bags via an instance similarity measure. This feature mapping often provides a large number of redundant or irrelevant features. Hence 1-norm SVM is applied to select important features as well as construct classifiers simultaneously. We have performed extensive experiments. In comparison with other methods, MILES demonstrates competitive classification accuracy, high computation efficiency, and robustness to labeling uncertainty.

Index Terms—Multiple-instance learning, feature subset selection, 1-norm support vector machine, image categorization, object recognition, drug activity prediction.

I. INTRODUCTION

Multiple-instance learning (MIL) was introduced by Dietterich et al. [19] in the context of drug activity prediction. It provides a framework to handle the scenarios where training class labels are naturally associated with sets of samples, instead of individual samples. In recent years, a variety of learning problems (e.g., drug activity prediction [19], [35], stock market prediction [36], data mining applications [46], image retrieval [56], [60], natural scene classification [37], text categorization [2], and image categorization [16]) have been tackled as multiple-instance problems.

A. Multiple-Instance Problems

A multiple-instance problem involves ambiguous training examples: a single example is represented by several feature vectors (instances), some of which may be responsible for the observed classification of the example; yet, the training label is only attached to the example instead of the instances. The multiple-instance problem was formally introduced in the context of drug activity prediction [19], while a similar type of learning scenario was first proposed in [13].

The goal of drug activity prediction is to foretell the potency of candidate drug molecules by analyzing a collection of previously synthesized molecules whose potencies are already known. The potency of a drug molecule is determined by the degree to which it binds to a site of medical interest on the target protein. The 3-dimensional structure of a drug molecule decides principally the binding strength: a drug molecule binds to the target protein if the shape of the molecule conforms closely to the structure of the binding site. Unfortunately, a molecule can adopt a wide range of shapes by rotating some of its internal bonds. Therefore, without the knowledge of the 3-dimensional structure of the target protein, which is in general not available, knowing that a previously-synthesized molecule binds to the target protein does not directly provide the binding information on the shapes of the molecule. For the multiple-instance problem in [19], training examples are molecules (named bags [35]). A bag contains several shapes, called instances, of the molecule. A label of binding or not binding is attached to each training bag instead of its instances.

The goal is to learn the concept of binding, hence to predict whether a new bag, i.e., a new drug molecule, binds to the target protein. Note that a bag is a multiset in that the order of the instances is ignored; yet, the multiplicity is explicitly significant, i.e., a bag may contain identical instances.

Far beyond the drug activity prediction problem, the multiple-instance problem emerges naturally in a variety of challenging learning problems in computer vision, including natural scene classification [37], content-based image retrieval [34], [56], [60], [15], image categorization [6], [32], [16], and object detection and recognition [40], [1], [21], [20], [45], [17], [42], [5]. Generally speaking, the goal of all these problems is to learn visual concepts from labeled images. The multiple-instance situation arises in these learning problems because of the choice of image representation. A single image in [37], [56], and [60] is represented by a collection of fixed-size blocks. In [16], an image is characterized by regions obtained from image segmentation. In [1], [21], [20], [45],
[17], [42], and [5] an image is described by a set of patches invariant to certain geometric transformations. These image patches are typically generated by salient region detectors [27], interest point detectors [38], and edge-based detectors [52]. In all of the previously mentioned learning problems, training labels are attached to images instead of the blocks, regions, or patches contained in the images. These learning problems match squarely with the bag-of-instances model: an image is a bag that contains instances corresponding to blocks, regions, or patches; instance labels are only indirectly accessible through the labels attached to bags.

B. Previous Work Related to Multiple-Instance Learning

One of the earliest algorithms for learning from multiple-instance examples was developed by Dietterich et al. [19] for drug activity prediction. The key assumption of their MIL formulation is that: a bag is positive if at least one of its instances is a positive example; otherwise, the bag is negative. Their algorithm, named the axis-parallel rectangle (APR) method [19], attempts to find an APR by expanding or shrinking a hyper-rectangle in the instance feature space to maximize the number of instances from different positive bags enclosed by the rectangle while minimizing the number of instances from negative bags within the rectangle. A bag is classified as positive if at least one of its instances falls within the APR; otherwise, the bag is classified as negative. A number of theoretical results have been obtained following the APR framework [4], [10], [33]. De Raedt showed the connection between MIL and inductive logic programming [18].

The basic idea of APR was extended in different ways that led to several MIL algorithms. Maron and Lozano-Pérez [35] proposed a general framework, named diverse density (DD), which has been tested on applications including stock market prediction [36], natural scene classification [37], and content-based image retrieval [56], [60]. Diverse density measures a co-occurrence of similar instances from different positive bags. The desired concept is learned by maximizing the DD function. Zhang and Goldman [61] combined the idea of expectation-maximization (EM) with DD, and developed an algorithm, EM-DD, to search for the most likely concept. Andrews et al. [2] formulated MIL as a mixed integer quadratic program. Integer variables are selector variables that select a positive instance from each positive bag. Their algorithm, which is called MI-SVM, has an outer loop and an inner loop. The outer loop sets the values of the selector variables. The inner loop then trains a standard support vector machine, SVM, in which the selected positive instances replace the positive bags. Later, Andrews and Hofmann [3] developed an algorithm based on a generalization of linear programming boosting. Ramon and De Raedt presented a neural networks framework for MIL [43]. A similar method has been derived in [59]. Wang and Zucker [53] presented two variants of the k-nearest neighbor algorithm (Bayesian-kNN and Citation-kNN) using Hausdorff distance. Zhou and Zhang [63] proposed ensembles of multi-instance learners, which achieved competitive test results on the benchmark data sets of drug activity prediction.

All the above MIL formulations explicitly or implicitly encode the assumption that a bag is positive if and only if at least one of its instances is positive. The assumption is valid in drug activity prediction. Typically, there is a single shape that allows binding, hence it is natural to relate the instance labels to the bag label through a disjunction (or a soft disjunction) function. However, for applications such as object recognition, a negative bag (e.g., a background image without the object of interest) may also contain instances that look similar to parts of the object. In other words, if a positive instance label indicates that the instance appears to be part of the object, a negative bag may contain positive instances as well. Several algorithms have been proposed to tackle this type of situation. Scott et al. [47] developed a method in which a bag is positive, if and only if, it contains a collection of instances, each near one of a set of target points. In [60], Zhang et al. observed that the average prediction of a collection of classifiers, constructed from the EM-DD algorithm with different starting points, is usually more accurate than the prediction given by any individual classifier in a content-based image retrieval experiment. Chen and Wang [16] proposed a MIL framework, DD-SVM, which, instead of taking the average prediction of classifiers built from the EM-DD, trains an SVM in a feature space constructed from a mapping defined by the local maximizers and minimizers of the DD function. Through the feature mapping, DD-SVM essentially converts MIL to a standard supervised learning problem.

There is an abundance of prior work that uses standard supervised learning techniques to solve multiple-instance problems. Zucker and Chevaleyre [65] applied decision trees and decision rules to a generalized multiple-instance problem. Gärtner et al. [22] designed a kernel for multiple-instance data. Hence, SVMs can be learned directly from the training bags. Xu and Frank [58] proposed logistic regression and boosting approaches for MIL with the assumption that all instances contribute equally and independently to a bag's label. Weidmann et al. [55] introduced a two-level learning framework, in which a bag is transformed into a meta-instance that can be learned by a propositional method. In a recent work [44], Ray and Craven compared several MIL methods with their supervised counterparts. They observed that although some MIL methods consistently outperformed their supervised counterparts, in several domains, a supervised algorithm gave the best performance among all the MIL algorithms they tested.

In the area of computer vision, several standard supervised learning techniques have been adapted for object detection and recognition where the underlying learning problems are essentially the multiple-instance problem [1], [21], [17], [42], [5]. Fergus et al. [21] presented a generative approach to learning and recognizing an object class from images. Each image (bag) is represented as a collection of image patches (instances) produced by a scale invariant detector [26]. An object is modeled as a group of patches, which are assumed to be generated by a probabilistic model. The model is estimated by maximizing the likelihood function using the EM technique. Agarwal and Roth [1] described a discriminative approach for object detection. A vocabulary of object parts is constructed from a set of sample images. The multiple-instance representation of each image is then transformed to
a binary feature vector based on the vocabulary. Each binary feature indicates whether or not a part or relation occurs in the image. The classifier is learned using the Sparse Network of Winnows (SNoW) framework. Csurka et al. [17] proposed an object categorization method using the bag-of-keypoints model, which is identical to the bag-of-instances model. An image is represented by a collection of affine invariant patches. Vector quantization is applied to the descriptors of all image patches to generate a predetermined number of clusters. Each image is then transformed to an integer-valued feature vector indicating the number of patches assigned to each cluster. Classifiers are implemented as Naive Bayes and SVM. Opelt et al. [42] described a boosting approach to object detection where, at each iteration, the weak hypothesis finder selects one local image patch and one type of local descriptor. Bar-Hillel et al. [5] proposed a boosting technique that learns a generative object model in a discriminative manner.

C. An Overview of Our Approach

The approach we take to tackle the multiple-instance problem further extends ideas from the diverse density framework [35], [16] and the wrapper model in feature selection [29]. Our approach identifies instances that are relevant to the observed classification by embedding bags into an instance-based feature space and selecting the most important features. We define a similarity measure between a bag and an instance. The coordinates of a given bag in the feature space represent the bag’s similarities to various instances in the training set. The embedding produces a possibly high dimensional space when the number of instances in the training set is large. In addition, many features may be redundant or irrelevant because some of the instances might not be responsible for the observed classification of the bags, or might be similar to each other. It is essential and indispensable to select a subset of mapped features that is most relevant to the classification problem of interest. Although any feature selection approaches could be applied for this purpose, we choose a joint approach that constructs classifiers and selects important features simultaneously. Specifically, we use the 1-norm SVM method because of its excellent performance in many applications [8], [64]. Since each feature is defined by an instance, feature selection is essentially instance selection. Therefore, we name our approach, MILES, Multiple-Instance Learning via Embedded instance Selection.

The proposed approach has the following characteristics:

- Broad adaptability: It provides a learning framework that converts a multiple-instance problem to a supervised learning problem. It demonstrates highly competitive classification accuracy in our empirical studies using benchmark data sets from different application areas. Moreover, in comparison with the DD-SVM algorithm [16], the proposed approach is less sensitive to the class label uncertainties.
- Low complexity: It is efficient in computational complexity, therefore, can potentially be tailored to tasks that have stringent time or resource limits. According to our empirical studies, the learning process of the proposed approach is on average one order of magnitude faster than that of the algorithms described in [21] and [16].
- Prediction capability: In some multiple-instance problems, classification of instances is at least as important as the classification of bags. The proposed approach supports instance naming, i.e., predicting instance labels. This is in contrast to the methods described in [53], [16], [17], and [42].

D. Outline of the Paper

The remainder of the paper is organized as follows. Section II describes an instance-based embedding of bags that transforms every bag to a uniform representation. Section III is dedicated to the description of a joint feature selection and classification method using a 1-norm SVM. A concrete 1-norm SVM formulation is presented. Section IV provides an algorithmic view of the approach. In Section V, we explain the extensive experimental studies conducted and demonstrate the results. We conclude and discuss possible future work in Section VI.

II. INSTANCE-BASED EMBEDDING OF BAGS

We introduce the instance-based embedding in this section. Before discussing the mathematical definition of the mapping, we first give a brief review of the diverse density framework based on [36] and [35], which forms the conceptual basis of the instance-based feature mapping. We denote positive bags as \( B_i^+ \) and the \( j \)-th instance in that bag as \( x_{ij}^+ \). The bag \( B_i^+ \) consists of \( n_i^+ \) instances \( x_{ij}^+ \), \( j = 1, \cdots, n_i^+ \). Similarly, \( B_i^- \), \( x_{ij}^- \), and \( n_i^- \) represent a negative bag, the \( j \)-th instance in the bag, and the number of instances in the bag, respectively. When the label on the bag does not matter, it will be referred to as \( B_i \) with instances as \( x_{ij} \). All instances belong to the feature space \( \mathbb{X} \). The number of positive (negative) bags is denoted as \( \ell^+ \) (\( \ell^- \)). For the sake of convenience, when we line up all instances in all bags together, we reindex these instances as \( x^k, k = 1, \cdots, n \) where \( n = \sum_{i=1}^{\ell^+} n_i^+ + \sum_{i=1}^{\ell^-} n_i^- \).

A. Review of Diverse Density

The diverse density framework was derived in [36] and [35] based on the assumption that there exists a single target concept, which can be used to label individual instances correctly. If we denote a given concept class as \( \mathcal{C} \), the diverse density of a concept \( t \in \mathcal{C} \) is defined as the probability\(^1\) that the concept \( t \) is the target concept given the training bags [36]:

\[
DD(t) = \Pr(t|B_i^+, \cdots, B_i^+, B_i^-, \cdots, B_i^-) \tag{1}
\]

The target concept that is most likely to agree with the data is then determined by maximizing this probability. Applying Bayes’ rule to (1) and further assuming that all bags are

\(^1\)Here, \( \mathcal{C} \) is assumed to be a countable set. Otherwise, the diverse density should be interpreted as probability density instead of probability. To simplify the notations, we use \( \Pr(\cdot) \) to represent either probability or probability density.
conditionally independent given the true target concept, we can write (1) as

\[ DD(t) = \frac{\Pr(t) \prod_{i=1}^{t} \Pr(B_i^+|t) \prod_{i=1}^{t} \Pr(B_i^-|t)}{\Pr(B_1^+, \ldots, B_{t}^+, B_{t+1}^-, \ldots, B_{t^+}^-)} \]

\[ = \left[ \prod_{i=1}^{t} \Pr(B_i^+|t) \prod_{i=1}^{t} \Pr(B_i^-|t) \right]^{\frac{t}{2}} \prod_{i=1}^{t} \Pr(t|B_i^+) \prod_{i=1}^{t} \Pr(t|B_i^-) \right] . \]  

(2)

If we assume a uniform prior on \( t \), for given training bags, maximizing \( DD(t) \) is equivalent to maximizing the second factor in (2), i.e., \( \prod_{i=1}^{t} \Pr(t|B_i^+) \prod_{i=1}^{t} \Pr(t|B_i^-) \). Maron [36] propose several ways to estimate \( \Pr(t|B_i) \) for various concept classes. For example, if \( C \) is a single point concept class where every concept corresponds to a single point in \( \mathbb{X} \), the most-likely-cause estimator [36] is then defined as:

\[ \Pr(t|B_i^+) \propto \max_j \exp \left( -\frac{||x_{ij}^+ - t||^2}{\sigma^2} \right) \]  

(3)

\[ \Pr(t|B_i^-) \propto 1 - \max_j \exp \left( -\frac{||x_{ij}^- - t||^2}{\sigma^2} \right) \]  

(4)

where \( \sigma \) is a predefined scaling factor.

An optimization algorithm, such as a gradient descent approach [35] or EM-DD [61], is used to search for a target concept that achieves maximal DD. Neither gradient descent algorithm nor EM-DD can guarantee the global optimality and hence they may get stuck at local solutions. Typically, multiple runs with different starting search points are necessary. Therefore, the process of maximization is often very time-consuming.

**B. Instance-Based Feature Mapping**

The DD framework can be interpreted from a feature selection point of view as follows. For a given concept class \( C \), each concept \( t \in C \) is viewed as an attribute or a feature (which is denoted as \( h_t \)) for the bags; and the value of the feature for bag \( B_i \) is defined as

\[ h_t(B_i) = \Pr(t|B_i) . \]  

(5)

If \( C \) is a countable set, i.e., \( C = \{ t_1, t_2, \ldots, t_j, \ldots \} \), then

\[ [h_t_1(B_i), h_{t_2}(B_i), \ldots, h_{t_j}(B_i), \ldots]^T = [\Pr(t_1|B_i), \Pr(t_2|B_i), \ldots, \Pr(t_j|B_i), \ldots]^T \]

determines the values of all the attributes (or features) for bag \( B_i \). We denote as \( F_C \) the space defined by \( h_{t_1}, h_{t_2}, \ldots, h_{t_j}, \ldots \). As a result, each bag can be viewed as a point in \( F_C \); and \( [\Pr(t_j|B_i^+), \ldots, \Pr(t_j|B_i^+), \Pr(t_j|B_i^-), \ldots, \Pr(t_j|B_i^-)]^T \) realizes the feature \( h_{t_j} \) for all the bags.

Under the uniform prior assumption on concepts, it is not difficult to see that finding a concept to maximize the DD function in (2) is equivalent to selecting one feature, \( h_{t_j} \), to maximize the following measure

\[ f(h_{t_j}) = \prod_{i=1}^{t} h_{t_j}(B_i^+) \prod_{i=1}^{t} h_{t_j}(B_i^-) \]

\[ = \prod_{i=1}^{t} \Pr(t_j|B_i^+) \prod_{i=1}^{t} \Pr(t_j|B_i^-) . \]

From the perspective of feature selection, the DD framework appears to be rather restrictive because it always seeks for one and only one feature. Can we improve the performance by searching for multiple features? This is the basic motivation of our approach. In particular, our approach further extends the idea from the DD framework in constructing the features, which is described below. It then applies 1-norm SVM to build classifiers and select features simultaneously. The details of the joint feature selection and classification approach will be discussed in Section III.

The new feature mapping is derived based on (5). First, we need to specify a concept class. We choose to use the single point concept class. In addition, we assume that there may exist more than one target concept (the exact number needs to be determined by the feature selection process) and a target concept can be well approximated by an instance in the training bags. In other words, each instance in the training bags is a candidate for target concepts. Therefore, each instance corresponds to a concept, i.e.,

\[ C = \{ x^k : k = 1, \ldots, n \} \]  

(6)

where \( x^k \) are those reindexed instances as defined at the beginning of Section II. Furthermore, we assume that a target concept can be related to either positive bags or negative bags, whereas in the DD framework the target concept is defined for positive bags only. Under this symmetric assumption and our choice of concept class in (6), the most-likely-cause estimator in (3) and (4) can be written, independent of the bag label, as

\[ \Pr(x^k|B_i) \propto s(x^k, B_i) = \max_j \exp \left( -\frac{||x_{ij} - x^k||^2}{\sigma^2} \right) . \]

\( s(x^k, B_i) \) can also be interpreted as a measure of similarity between the concept \( x^k \) and the bag \( B_i \). The similarity between a concept and a bag is determined by the concept and the closest instance in the bag. A bag \( B_i \) is then embedded in \( F_C \) with coordinates \( m(B_i) \) defined as

\[ m(B_i) = [s(x^1, B_i), s(x^2, B_i), \ldots, s(x^n, B_i)]^T . \]  

(7)

For a given training set of \( \ell^+ \) positive bags and \( \ell^- \) negative bags, applying the mapping (7) yields the following matrix representation of all training bags in \( F_C \):

\[
\begin{bmatrix}
    m^+_1, \ldots, m^+_\ell^+ & m^-_1, \ldots, m^-_\ell^-
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    m(B^+_1), \ldots, m(B^+\ell^+), m(B^-_1), \ldots, m(B^-\ell^-)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    s(x^1, B^+_1) & \ldots & s(x^1, B^-_1) \\
    \vdots & \ddots & \vdots \\
    s(x^n, B^+_1) & \ldots & s(x^n, B^-_1)
\end{bmatrix}
\]
where each column represents a bag, and the $k$-th feature in $\mathbb{F}_C$ realizes the $k$-th row of the matrix, i.e.,

$$s(x^k, \cdot) = [s(x^k, B^+_1), \ldots, s(x^k, B^+_\ell), s(x^k, B^-_1), \ldots, s(x^k, B^-_\ell)].$$

Intuitively, if $x^k$ achieves high similarity to some positive bags and low similarity to some negative bags, the feature induced by $x^k$, $s(x^k, \cdot)$, provides some “useful” information in separating the positive and negative bags. Next, we present a simple example to illustrate the efficiency of the mapping (7).

C. An Example

We formulate a multiple-instance problem where each instance is generated by one of the following 2-dimensional probability distributions: $N_1 \sim \mathcal{N}([5, 5]^T, I)$, $N_2 \sim \mathcal{N}([-5, -5]^T, I)$, $N_3 \sim \mathcal{N}([-5, 5]^T, I)$, $N_4 \sim \mathcal{N}([5, -5]^T, I)$, and $N_5 \sim \mathcal{N}([0, 0]^T, I)$ where $\mathcal{N}([5, 5]^T, I)$ denotes the normal distribution with mean $[5, 5]^T$ and identity covariance matrix. Each bag comprises at most 8 instances. A bag is labeled positive if it contains instances from at least two different distributions among $N_1, N_2,$ and $N_3$. Otherwise, the bag is negative.

Using this model, we generated 20 positive bags and 20 negative bags with a total of 219 instances. Figure 1(a) depicts all the instances on a 2-dimensional plane. Note that instances from negative bags mingle with those from positive bags because a negative bag may include instances from any one, but only one, of the distributions $N_1, N_2,$ and $N_3$. After applying mapping (7), each bag was embedded in a 219-dimensional feature space ($\sigma$ was chosen to be 4.5). Intuitively, the mean vectors of $N_1, N_2$, and $N_3$ might be helpful in bag classification. So we manually selected three instances, $x^1 = [4.3, 5.2]^T$, $x^2 = [5.4, -3.9]^T$, and $x^k = [-6.0, 4.8]^T$, which were close to the mean vectors of $N_1, N_2$, and $N_3$. Figure 1(b) shows the embedding of all 40 bags in a subspace defined by the three instances. It is not difficult to observe that the bags can be separated by a plane in this 3-dimensional space, which corresponds to a simple bag classifier. This suggests that the mapping (7) can indeed induce useful features for bag classification, yet, automatic feature subset selection and the subsequent classifier design are the remaining questions. These questions are addressed in the next section.

III. Joint Feature Selection and Classification for Multiple-Instance Learning

Feature subset selection is a well-studied research problem in the areas of statistics, machine learning and pattern recognition [41], [29], [25], [30], [39], [12], [23], [28], [31], [50], [57]. Existing feature selection approaches generally fall in two categories: filter and wrapper [29], [57]. Some filter methods such as ranking through correlation coefficients or through Fisher scores, tend to select inter-correlated features and do not guarantee an acquisition of a good classifier. On the contrary, wrappers include the desired classifier as part of their performance evaluation. They tend to produce better generalization but may require an expensive computational cost. Our method is essentially a wrapper where the 1-norm SVM is used to construct classifiers and select important features simultaneously. The 1-norm SVM can be formulated as a linear program, which, in general, can be solved efficiently from an optimization point of view. The computational cost will not be an issue.

A. 1-Norm Support Vector Machines

Denote the class label variable by $y$, which takes values of +1 and −1. We consider the classification problem of finding a linear classifier

$$y = \text{sign} \left( w^T m + b \right)$$

in the feature space $\mathbb{F}_C$ to distinguish between positive bags and negative bags where $w$ and $b$ are model parameters, $m \in \mathbb{F}_C$ corresponds to a bag. The SVM approach constructs classifiers based on hyperplanes by minimizing a regularized training error, $\mathcal{E}_{\text{training}}$,

$$\lambda P[\cdot] + \mathcal{E}_{\text{training}}$$

where $P[\cdot]$ is a regularizer, $\lambda$ is called the regularization parameter, and $\mathcal{E}_{\text{training}}$ is commonly defined as a total of the loss that each bag introduces through a hinge loss function

$$\xi = \max \{1 - y(w^T m + b), 0\}.$$ 

When an optimal solution $w$ is obtained, the magnitude of its component $w_k$ indicates the significance of the effect of the $k$-th feature in $\mathbb{F}_C$ on the classifier. Those features corresponding to a non-zero $w_k$ are selected and used in the classifier.

The regularizer in standard SVMs is the squared 2-norm of the weight vector $\|w\|$, which formulates SVMs as quadratic programs (QP). Solving QPs is typically computationally more expensive than solving linear programs (LPs). There exist alternative LP formulations of SVMs [7], [49], [64] which involve regularization with a norm favoring sparsity, e.g., the 1-norm of $w$,

$$\|w\|_1 = \sum_k |w_k|.$$ 

The 1-norm penalty was adopted in other approaches similarly such as basis pursuit [14] and LASSO [51] for driving more components of $w$ to zero. Thus 1-norm SVM is also referred to as sparse SVM and has been applied to other practical problems such as drug discovery [8].

Another characteristic of the MIL problem is that training sets in many MIL problems are very imbalanced between classes, e.g., the number of negative bags can be much larger than the number of positive bags. To tackle this imbalanced issue and make classifiers biased towards the minor class, a simple strategy we used is to penalize differently on errors produced respectively by positive bags and by negative bags. Hence the 1-norm SVM is formulated as follows:

$$\min_{w, b, \xi, \eta} \lambda \sum_{k=1}^{n} |w_k| + C_1 \sum_{i=1}^{\ell^+} \xi_i + C_2 \sum_{j=1}^{\ell^-} \eta_j$$

s.t. \begin{align*}
(w^T m^+_i + b) + \xi_i & \geq 1, \quad i = 1, \ldots, \ell^+, \\
-w^T m^-_j + b + \eta_j & \geq 1, \quad j = 1, \ldots, \ell^-, \\
\xi_i, \eta_j & \geq 0, \quad i = 1, \ldots, \ell^+, j = 1, \ldots, \ell^-, 
\end{align*}
where $\xi$, $\eta$ are hinge losses. Choosing unequal values for parameters $C_1$ and $C_2$ will penalize differently on false negatives and false positives. Usually $C_1$ and $C_2$ are chosen so that the training error is determined by a convex combination of the training errors occurred on positive bags and on negative bags. In other words, let $C_1 = \mu$ and $C_2 = 1 - \mu$ where $0 < \mu < 1$.

To form a LP for the 1-norm SVM, we rewrite $w_k = u_k - v_k$ where $u_k, v_k \geq 0$. If either $u_k$ or $v_k$ has to equal to 0, we have $|w_k| = u_k + v_k$. The LP is then formulated in variables $u$, $v$, $b$, $\xi$ and $\eta$ as

$$
\text{min}_{u, v, b, \xi, \eta} \lambda \sum_{k=1}^{n} (u_k + v_k) + \mu \sum_{i=1}^{\ell^+} \xi_i + (1 - \mu) \sum_{j=1}^{\ell^-} \eta_j
$$

s.t.

$$
\begin{align*}
& [(u - v)^T m_i^+ + b] + \xi_i \geq 1, i = 1, \ldots, \ell^+, \\
& -[(u - v)^T m_j^- + b] + \eta_j \geq 1, j = 1, \ldots, \ell^-, \\
& u_k, v_k \geq 0, k = 1, \ldots, n, \\
& \xi_i, \eta_j \geq 0, i = 1, \ldots, n, \\
& \ell^+, \ell^-,
\end{align*}
$$

(9)

Solving LP (9) yields solutions equivalent to those obtained by the 1-norm SVM (8) because any optimal solution to (9) has at least one of the two variables $u_k$ and $v_k$ equal to 0 for all $k = 1, \ldots, n$. Otherwise, assume $u_k \geq v_k > 0$ without loss of generality, and we can find a better solution by setting $u_k = u_k - v_k$ and $v_k = 0$, which contradicts the optimality of $(u, v)$.

Let $w^* = u^* - v^*$ and $b^*$ be the optimal solution of (9). The magnitude of $w^*_k$ determines the influence of the $k$-th feature on the classifier. The set of selected features is given as $\{s(x_k, \cdot) : k \in I\}$ where

$$
I = \{k : |w^*_k| > 0\}
$$

is the index set for nonzero entries in $w^*$. The classification of bag $B_i$ is computed as

$$
y = \text{sign} \left( \sum_{k \in I} w_k^*(x_k, B_i) + b^* \right). 
$$

(10)

### B. Instance Classification

In some multiple-instance problems, classification of instances is at least as important as the classification of bags. For one example, an object detection algorithm needs not only to identify whether or not an image contains a certain object, but also locate the object (or part of the object) from the image if it contains the object. Under the multiple-instance formulation, this requires the classification of the bags (containing an object versus not containing any such object) as well as the instances in a bag that correspond to the object. The classifier (10) predicts a label for a bag. Next, we introduce a way to classify instances based on a bag classifier.

The basic idea is to classify instances according to their contributions to the classification of the bag. Instances in a bag can be grouped into three classes: positive class, negative class, and void class. An instance in bag $B_i$ is assigned to the positive class (negative class) if its contribution to $\sum_{k \in I} w_k^* s(x_k, B_i)$ is greater than or equal to (or less than) a threshold. An instance is assigned to the void class if it makes no contribution to the classification of the bag.

Given a bag $B_i$ with instances $x_{ij}, j = 1, \ldots, n_i$, we define an index set $\mathcal{U}$ as

$$
\mathcal{U} = \left\{ j^* : j^* = \arg \max_j \exp \left( -\frac{\|x_{ij} - x_k^*\|^2}{\sigma^2} \right), k \in I \right\}.
$$

It is not difficult to verify that the evaluation of (10) only needs the knowledge of the instances $x_{ij^*}, j^* \in \mathcal{U}$. In this sense, $\mathcal{U}$ defines a minimal set of instances responsible for the classification of $B_i$. Hence, removing an instance $x_{ij^*}, j^* \notin \mathcal{U}$ from the bag will not affect the value of $\sum_{k \in I} w_k^* s(x_k, B_i)$ in (10), and $\{1, \ldots, n_i\}$ minus the set $\mathcal{U}$ specifies the instances
in the void class. Since there can be more than one instance in the bag $B_i$ that maximizes $\exp\left(-\frac{\|x_{ij} - x^k\|^2}{\sigma^2}\right)$ for a given $x^k, k \in I$, we denote the number of maximizers for $x^k$ by $m_k$. Also, an instance $x_{ij}, j^* \in U$ can be a maximizer for different $x^k$’s, $k \in I$. Hence, for each $j^* \in U$, we define

$$\mathcal{I}_{j^*} = \left\{ k : k \in I, j^* = \arg\max_j \exp\left(-\frac{\|x_{ij} - x^k\|^2}{\sigma^2}\right) \right\}.$$ 

All the features $s(x^k, B_i), k \in \mathcal{I}_{j^*}$, can be computed from $x_{ij}$, i.e., $s(x^k, B_i) = s(x^k, \{x_{ij}\})$ for $k \in \mathcal{I}_{j^*}$.

It is straightforward to show that $\mathcal{I} = \bigcup_{j^* \in \mathcal{I}} \mathcal{I}_{j^*}$, and in general, $\mathcal{I}_{j_1} \cap \mathcal{I}_{j_2} \neq \emptyset$ for arbitrary $j_1 \neq j_2 \in \mathcal{U}$. In fact, the number of appearances of $k$ in $\mathcal{I}_{j_1}, \ldots, \mathcal{I}_{j_m}$ is $m_k$. We then rewrite the bag classifier (10) in terms of the instances indexed by $U$:

$$y = \text{sign}\left( \sum_{j^* \in \mathcal{U}} g(x_{ij^*}) + b^* \right)$$

where

$$g(x_{ij^*}) = \sum_{k \in \mathcal{I}_{j^*}} \frac{w_k^* s(x^k, x_{ij^*})}{m_k} \tag{11}$$

determines the contribution of $x_{ij^*}$ to the classification of the bag $B_i$. The instances can be classified according to (11): if $g(x_{ij^*}) > \tau$, $x_{ij^*}$ belongs to the positive class; otherwise, $x_{ij^*}$ belongs to the negative class. The choice of $\tau$ is application specific, and is an interesting research problem for its own sake. In our experiments, the parameter $\tau$ is chosen to be bag dependent as $-\frac{b^*}{|\mathcal{U}|}$.

Next, we present one simple example to illustrate the major steps in the above instance classification process.

**Example:** We assume that 5 features are selected by 1-norm SVM, and denote these features as $s(x^1, \cdot), \ldots, s(x^5, \cdot)$. Therefore, $\mathcal{I} = \{1, 2, 3, 4, 5\}$. A bag $B_i$, containing 4 instances ($x_{i1}, x_{i2}, x_{i3}, x_{i4}$), is classified as positive. The key information needed in computing $s(x^k, B_i)$ is the nearest neighbors of $x^k$ in the instances of $B_i$. Suppose that

- $x_{i2}, x_{i3}$ are the nearest neighbors of $x^1$;
- $x_{i1}$ is the nearest neighbor of $x^2$;
- $x_{i2}$ is the nearest neighbor of $x^3$;
- $x_{i1}$ is the nearest neighbor of $x^4$;
- $x_{i3}$ is the nearest neighbor of $x^5$. \tag{12}

Instances $x_{i1}, x_{i2}$ and $x_{i3}$ are the nearest neighbors of at least one of $x^k$’s, so $\mathcal{U} = \{1, 2, 3, 4\}$. Since $x_{i4}$ is not the nearest neighbor for any of $x^k$’s, $x_{i4}$ is assigned to the void class. Because $x^1$ has two nearest neighbors, it yields $m_1 = 2$. Similarly, we have $m_2 = m_3 = m_4 = m_5 = 1$. From (12), we find that $x_{i1}$ is the nearest neighbor of $x^2$ and $x^4$. So $x_{i1}$ determines the values of features $s(x^2, \cdot)$ and $s(x^4, \cdot)$. Therefore, $\mathcal{T}_1 = \{2, 4\}$. Similarly, we derive $\mathcal{T}_2 = \{1\}$ and $\mathcal{T}_3 = \{1, 3, 5\}$. In other words, the instance $x_{i1}$ contributes to the classification of $B_i$ via features $s(x^2, \cdot)$ and $s(x^4)$; the instance $x_{i2}$ contributes to the classification via $s(x^1, \cdot)$; and the instance $x_{i3}$ contributes to the classification via $s(x^3, \cdot)$, $s(x^5, \cdot)$, and $s(x^4, \cdot)$. The labels of instances $x_{i1}, x_{i2}$, and $x_{i3}$ can be predicted using (11).

### IV. An Algorithmic View

We summarize the above discussion in pseudo code. The input is a set of labeled bags $\mathcal{D}$, parameters $\sigma^2$, $\lambda$, and $\mu$. The collection of instances from all the bags is denoted as $\mathcal{C} = \{x^k : k = 1, \ldots, n\}$. The following pseudo code learns a bag classifier defined by $w^*$ and $b^*$.

**Algorithm 4.1: Learning Bag Classifier**

1. FOR (every bag $B_i = \{x_{ij} : j = 1, \ldots, n_i\}$ in $\mathcal{D}$)
2. FOR (every instance $x^k$ in $\mathcal{C}$)
3. $d = \min_j \|x_{ij} - x^k\|
4. $i_k = \arg\min_j \|x_{ij} - x^k\|
5. $m_k = 0$ for every $k$ in $\mathcal{T}$
6. FOR (every $j^*$ in $\mathcal{U}$)
7. $I_{j^*} = \{k : k \in I, j^* = \arg\min_j \|x_{ij} - x^k\|\}$
8. $m_k = m_k + 1$ for every $k$ in $I_{j^*}$
9. END
10. END
11. OUTPUT (all $x_{ij}$ satisfying $g(x_{ij}) > -\frac{b^*}{|\mathcal{U}|}$)

These outputs $x_{ij}$ correspond to positive instances.

### V. Experimental Results

We present systematic evaluations of the proposed MIL framework, MILES, based on three publicly available benchmark data sets. In Section V-A, we compare MILES with other MIL methods using the benchmark data sets in MIL, MUSK data sets [19]. In Section V-B, we test the performance of MILES on a region-based image categorization problem using the same data set as in [16], and compare MILES with other techniques. In Section V-C, we apply MILES to an object class recognition problem [21]. MILES is compared to several techniques specifically designed for the recognition task in terms of its performance. Computational issues are discussed in Section V-D. A Matlab implementation of MILES is available at [http://www.cs.olemiss.edu/~ychen/MILES.html](http://www.cs.olemiss.edu/~ychen/MILES.html).

#### A. Drug Activity Prediction

1) **Experimental Setup:** The MUSK data sets, MUSK1 and MUSK2, are benchmark data sets for MIL. Both data sets are publicly available from the UCI Machine Learning Repository [9]. The data sets consist of descriptions of molecules.
Specifically, a bag represents a molecule. Instances in a bag represent low-energy shapes of the molecule. To capture the shape of molecules, a molecule is placed in a standard position and orientation and then a set of 162 rays emitting from the origin is constructed to sample the molecular surface approximately uniformly [19]. There are also four features that describe the position of an oxygen atom on the molecular surface. Therefore, each instance in the bags is represented by approximately 5.17 shapes (instances) per molecule. MUSK2 has 102 molecules, of which 39 are positive, with an average of \( \lambda \times 5 \times 10^5 \) features per molecule. MUSK1 has 166 molecules (bags), of which 47 are labeled positive, with an average of 5.17 shapes (instances) per molecule.

Three parameters \( \sigma^2 \) (in the most-likely-cause estimator), \( \lambda \), and \( \mu \) (in the LP) need to be specified for MILES. We fixed \( \mu = 0.5 \) to penalize equally on errors in the positive class and the negative class. The parameters \( \lambda \) and \( \sigma^2 \) were selected according to a twofold cross-validation on the training set. We chose \( \lambda \) from 0.1 to 0.6 with step size 0.01, and \( \sigma^2 \) from \( 10^5 \) to \( 9 \times 10^5 \) with step size \( 5 \times 10^4 \). We found that \( \lambda = 0.45 \) and \( \sigma^2 = 5 \times 10^5 \) gave the minimum twofold cross-validation error on MUSK1, and \( \lambda = 0.37 \) and \( \sigma^2 = 8 \times 10^5 \) gave the best twofold cross-validation performance on MUSK2. These values were fixed for the subsequent experiments. The linear program of 1-norm SVM was solved using CPLEX version 9.0 [24].

2) Classification Results: Table 1 shows the prediction accuracy. We observed variations on the average accuracy of tenfold cross-validation for different random runs. The prediction accuracy in 15 runs varied from 84.1% to 90.2% for MUSK1, and from 84.5% to 91.5% for MUSK2. Therefore, we reported the mean and 95% confidence interval of the results of 15 runs of tenfold cross-validation for MILES.

We also included the results using the leave-one-out test for comparison with certain algorithms.

Table 1 summarizes the performance of twelve MIL algorithms in the literature: APR [19], DD-SVM [16], DD-SVM and mi-SVM [2], MI-NN [43], Multinst [4], Citation-kNN and Bayesian-kNN [53], RELIC [46], and Bagging-APR [63]. The prediction accuracies for Bayesian-kNN and Citation-kNN were based on the leave-one-out test, while the performance of other methods was evaluated using tenfold cross-validation.

Table 1 shows that Bagging-APR achieves the best performance on both MUSK1 and MUSK2 data sets. Bagging-APR applied an ensemble algorithm, bagging [11], to a base learner that used the APR method. Therefore, its excellent performance should be credited to the ensemble scheme as well as the base learner. Since the bagging technique can potentially be applied to all algorithms listed in Table 1 to improve the prediction accuracy, it might be more meaningful to compare only the base learners. Among all 12 base multiple-instance learners, the APR method gave the best overall prediction accuracy on MUSK1 and MUSK2 data sets (the average accuracy on two data sets is 90.8%). It is observed that the proposed framework, MILES, is the second best in terms of the average prediction accuracy (90.1%, leave-one-out test) over the two data sets. The tenfold cross-validation results of MILES are also comparable with those of other techniques. Even though MILES is not designed specially for the drug

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MUSK1</th>
<th>MUSK2</th>
<th>Type of Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILES</td>
<td>86.3 : [84.9, 87.7]</td>
<td>87.7 : [86.3, 89.1]</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>APR [19]</td>
<td>92.4</td>
<td>89.2</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>Bagging-APR [63]</td>
<td>92.8</td>
<td>93.1</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>Bayesian-kNN [53]</td>
<td>90.2</td>
<td>82.4</td>
<td>leave-one-out test</td>
</tr>
<tr>
<td>Citation-kNN [53]</td>
<td>92.4</td>
<td>86.3</td>
<td>leave-one-out test</td>
</tr>
<tr>
<td>DD [35]</td>
<td>88.9</td>
<td>82.5</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>DD-SVM [16]</td>
<td>85.8</td>
<td>91.3</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>EM-DD [61]</td>
<td>84.8</td>
<td>84.9</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>mi-SVM [2]</td>
<td>87.4</td>
<td>83.6</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>MI-SVM [2]</td>
<td>77.9</td>
<td>84.3</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>MI-NN [43]</td>
<td>88.0</td>
<td>82.0</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>Multinst [4]</td>
<td>76.7 : [73.6, 79.8]</td>
<td>84.0 : [81.4, 86.6]</td>
<td>tenfold cross-validation</td>
</tr>
<tr>
<td>RELIC [46]</td>
<td>83.7</td>
<td>87.3</td>
<td>tenfold cross-validation</td>
</tr>
</tbody>
</table>

2The EM-DD results reported in [61] were obtained by selecting the optimal solution using the test data. The EM-DD result cited in this paper was provide by [2] using the correct algorithm.

3The best performance reported in [63] was achieved by combining EM-DD learners (prediction accuracy: 96.5% for MUSK1 and 97.0% for MUSK2). However, it seems that the EM-DD learner implemented in [63] used the test data to select the optimal solution. Therefore, the results were not included in Table 1.
activity prediction problem, its performance is comparable with that of the APR method. In fact, MILES generated the best performance on MUSK2 (93.1%, leave-one-out test).

We tested the sensitivity of MILES with respect to the parameters $\lambda$ and $\sigma^2$. Our algorithm is rather stable with choices of parameter values as long as they are chosen from a reasonable range. Figure 2 shows the grey level encoded leave-one-out test accuracy under different values of $\lambda$ and $\sigma^2$. For MUSK1, $\lambda$ varies from 0.2 to 0.5 with step size 0.01; $\sigma^2$ varies from $3 \times 10^5$ to $7 \times 10^5$ with step size $5 \times 10^4$. The maximal and minimal leave-one-out test accuracies are respectively 89.1% and 81.5% (recall that the accuracy based on the parameters chosen from twofold cross-validation is 87.0%). For MUSK2, $\lambda$ varies from 0.1 to 0.6 with step size 0.02; $\sigma^2$ varies from $5 \times 10^5$ to $9 \times 10^5$ with step size $5 \times 10^4$. The maximal and minimal leave-one-out test accuracies are respectively 94.1% and 83.3% (recall that the accuracy based on the parameters chosen from twofold cross-validation is 93.1%).

B. Region-Based Image Categorization

Image categorization refers to the labeling of images into predefined categories. Depending on the imagery features used in the classification, image categorization algorithms can be divided into two groups: global approaches and component-based approaches. The global image classification approaches use features that characterize the global information of an image. Although the global features can usually be computed with little cost and are effective for certain classification tasks, some visual contents of images could only be locally defined. A number of component-based approaches have been proposed to exploit local and spatial properties of an image. Component-based image categorization has been formulated and tackled as a multiple-instance learning problem in the past works [37], [56], [60], [2], [16].

1) Experimental Setup: The image data set consists of 2,000 images taken from 20 CD-ROMs published by the COREL Corporation. Each COREL CD-ROM contains 100 images representing a distinct concept. Therefore, the data set has 20 thematically diverse image categories, each containing 100 images. Images are in JPEG format of size $384 \times 256$ or $256 \times 384$. The category names are listed in Table II along with the identifiers (IDs) for 20 categories. Since the classification problem is multi-class, we apply the simple one-against-the-

<table>
<thead>
<tr>
<th>Category ID</th>
<th>Category Name</th>
<th>Regions per Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>African people and villages</td>
<td>4.84</td>
</tr>
<tr>
<td>1</td>
<td>Beach</td>
<td>3.54</td>
</tr>
<tr>
<td>2</td>
<td>Historical building</td>
<td>3.1</td>
</tr>
<tr>
<td>3</td>
<td>Buses</td>
<td>7.59</td>
</tr>
<tr>
<td>4</td>
<td>Dinosaurs</td>
<td>2.00</td>
</tr>
<tr>
<td>5</td>
<td>Elephants</td>
<td>3.02</td>
</tr>
<tr>
<td>6</td>
<td>Flowers</td>
<td>4.46</td>
</tr>
<tr>
<td>7</td>
<td>Horses</td>
<td>3.89</td>
</tr>
<tr>
<td>8</td>
<td>Mountains and glaciers</td>
<td>3.38</td>
</tr>
<tr>
<td>9</td>
<td>Food</td>
<td>7.24</td>
</tr>
<tr>
<td>10</td>
<td>Dogs</td>
<td>3.80</td>
</tr>
<tr>
<td>11</td>
<td>Lizards</td>
<td>2.80</td>
</tr>
<tr>
<td>12</td>
<td>Fashion models</td>
<td>5.19</td>
</tr>
<tr>
<td>13</td>
<td>Sunset scenes</td>
<td>3.52</td>
</tr>
<tr>
<td>14</td>
<td>Cars</td>
<td>4.93</td>
</tr>
<tr>
<td>15</td>
<td>Waterfalls</td>
<td>2.56</td>
</tr>
<tr>
<td>16</td>
<td>Antique furniture</td>
<td>2.30</td>
</tr>
<tr>
<td>17</td>
<td>Battle ships</td>
<td>4.32</td>
</tr>
<tr>
<td>18</td>
<td>Skiing</td>
<td>3.34</td>
</tr>
<tr>
<td>19</td>
<td>Desserts</td>
<td>3.65</td>
</tr>
</tbody>
</table>
coefficients in the HL band are given as into four frequency bands: the LL (low low), LH (low high), high frequency bands. The coefficients in different frequency bands show variations in different directions, hence capture the texture properties. To calculate these moments, a Daubechies-4 root of the second order moment of wavelet coefficients in frequency bands of the wavelet transforms, i.e., the square root of energy in the high-frequency bands. The other three represent square root of energy in the low-frequency bands.

After one-level wavelet transform, a wavelet transform is applied to the L component of the image. This data set has been used in [16] to demonstrate much improved performance of a MIL algorithm, DD-SVM, in comparison with several other techniques. Since we will compare MILES with the DD-SVM approach, we adopt the same image segmentation algorithm as described in [16] and [54]. A brief summary about the imagery features is given as follows. To segment an image, the system first partitions the image into non-overlapping blocks of size $4 \times 4$ pixels. A feature vector is then extracted for each block. Each feature vector consists of 6 features. Three of them are the average color components in a block. The LUV color space is used, where L encodes luminance, and U and V encode color information (chrominance). The other three represent square root of energy in the high-frequency bands of the wavelet transforms, i.e., the square root of the second order moment of wavelet coefficients in high frequency bands. The coefficients in different frequency bands show variations in different directions, hence capture the texture properties. To calculate these moments, a Daubechies-4 wavelet transform is applied to the L component of the image. After one-level wavelet transform, a $4 \times 4$ block is decomposed into four frequency bands: the LL (low low), LH (low high), HL, and HH bands, each containing $2 \times 2$ coefficients. If the coefficients in the HL band are given as $c_{i,j}$, where $i, j = 1, 2$, then a feature is defined as $\left( \frac{1}{4} \sum_{i=1}^{2} \sum_{j=1}^{2} c_{i,j}^2 \right)^{\frac{1}{2}}$. The other two features are computed similarly from the LH and HH bands.

After segmentation, three extra features are computed for each region to describe shape properties. They are normalized inertia of order 1, 2, and 3. As a result, each region in any image is characterized by a 9-dimensional feature vector characterizing the color, texture, and shape properties of the region.

In our experiments, images within each category were randomly partitioned in half to form a training set and a test set. We repeated each experiment for 5 random splits, and reported the average of the results obtained over 5 different test sets. The parameter $\mu$ was set to be 0.5. Parameters $\sigma^2$ and $\lambda$ were selected according to a twofold cross-validation on the training set. We chose $\lambda$ from 0.1 to 0.6 with step size 0.05 and $\sigma^2$ from 5 to 15 with step size 1. We found that $\lambda = 0.5$ and $\sigma^2 = 11$ gave the minimum twofold cross-

![Fig. 3. Images randomly sampled from 20 categories and the corresponding segmentation results. Segmented regions are shown in their representative colors.](image-url)
validation error. We then fixed \( \lambda = 0.5 \) and \( \sigma^2 = 11 \) in all subsequent experiments.

2) Categorization Results: We first report the confusion matrix of the proposed method in Table III based on images in Category 0 to Category 9, i.e., 1,000 images. Each row lists the average percentages of images in a specific category classified to each of the 10 categories. The numbers on the diagonal show the classification accuracy for each category and off-diagonal entries indicate classification errors. A detailed examination of the confusion matrix shows that two of the largest diagonal entries indicate classification errors. A detailed examination of the confusion matrix shows that two of the largest diagonal entries indicate classification errors.

We compared the overall prediction accuracy of MILES with that of DD-SVM [16], MI-SVM [2], and a method proposed in [17] (we call it \( k \)-means-SVM to simplify the discussions). \( k \)-means-SVM constructed a region vocabulary by clustering regions using the \( k \)-means algorithm. Each image was then transformed to an integer-valued feature vector indicating the number of regions assigned to each cluster. SVMs were constructed using these integer-valued features. The size of the vocabulary and the parameters of \( k \)-means-SVM were chosen according to a twofold cross-validation, using all 2,000 images. The average classification accuracies over 5 random test sets and the corresponding 95\% confidence intervals are provided in Table IV. On both data sets, the performance of MILES is significantly better than that of MI-SVM and \( k \)-means-SVM. MILES outperforms DD-SVM, though the difference is not statistically significant as the 95\% confidence intervals for the two methods overlap.

3) Sensitivity to Labeling Noise: The results in Section V-B.2 demonstrate that the performance of MILES is highly comparable with that of DD-SVM. Next, we compare MILES with DD-SVM in terms of the sensitivity to the noise in labels. Sensitivity to labeling noise is an important performance measure for classifiers because in many practical applications, it is usually impossible to get a “clean” data set and the labeling processes are often subjective. In terms of binary classification, we define the labeling noise as the probability that an image is mislabeled. In this experiment, training sets with different levels of labeling noise were generated as follows. We first randomly picked \( d\% \) of positive images and \( d\% \) of negative images from a training set. Then, we modified the labels of the selected images by negating their labels, i.e., positive (negative) images were labeled as negative (positive) images. Finally, we put these images with new labels back to the training set. The new training set has \( d\% \) of images with negated labels (or “noisy” labels).

We compared the classification accuracy of MILES with that of DD-SVM for \( d = 0 \) to 30 (with step size 2) based on 200 images from Category 2 (Historical buildings) and Category 7 (Horses). The reason of selecting these two categories is that both DD-SVM and MILES produce almost perfect classification accuracies at \( d = 0 \), which makes them a good data set for comparing sensitivities at different levels of labeling noise. The training and test sets have equal size.

---

**Table III**
The confusion matrix of image categorization experiments (over 5 randomly generated test sets) using MILES. Each row lists the average percentage of images (test images) in one category classified to each of the 10 categories. Numbers on the diagonal show the classification accuracies (in %).

<table>
<thead>
<tr>
<th>Category</th>
<th>Cat. 0</th>
<th>Cat. 1</th>
<th>Cat. 2</th>
<th>Cat. 3</th>
<th>Cat. 4</th>
<th>Cat. 5</th>
<th>Cat. 6</th>
<th>Cat. 7</th>
<th>Cat. 8</th>
<th>Cat. 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat. 0</td>
<td>68.8</td>
<td>1.8</td>
<td>9.3</td>
<td>0.3</td>
<td>1.3</td>
<td>10.3</td>
<td>1.3</td>
<td>4.3</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Cat. 1</td>
<td>4.0</td>
<td>66.0</td>
<td>4.0</td>
<td>3.6</td>
<td>0.7</td>
<td>3.0</td>
<td>1.3</td>
<td>0.0</td>
<td>15.7</td>
<td>1.7</td>
</tr>
<tr>
<td>Cat. 2</td>
<td>4.6</td>
<td>2.0</td>
<td>75.7</td>
<td>4.0</td>
<td>0.7</td>
<td>6.3</td>
<td>0.7</td>
<td>0.3</td>
<td>3.7</td>
<td>2.0</td>
</tr>
<tr>
<td>Cat. 3</td>
<td>0.0</td>
<td>2.7</td>
<td>2.3</td>
<td>90.3</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.0</td>
<td>1.7</td>
</tr>
<tr>
<td>Cat. 4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>99.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 5</td>
<td>2.0</td>
<td>1.3</td>
<td>5.0</td>
<td>0.0</td>
<td>0.7</td>
<td>77.7</td>
<td>0.0</td>
<td>6.6</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>Cat. 6</td>
<td>3.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>Cat. 7</td>
<td>1.0</td>
<td>1.3</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>1.3</td>
<td>0.7</td>
<td>95.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 8</td>
<td>3.6</td>
<td>13.7</td>
<td>5.3</td>
<td>1.0</td>
<td>0.3</td>
<td>2.7</td>
<td>1.7</td>
<td>0.7</td>
<td>71.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 9</td>
<td>5.0</td>
<td>1.3</td>
<td>0.3</td>
<td>1.0</td>
<td>0.7</td>
<td>2.3</td>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>85.4</td>
</tr>
</tbody>
</table>

**Table IV**
Comparing the image categorization accuracy obtained using MILES with those of other methods. The numbers listed are the average classification accuracies (in %) over 5 random test sets and the corresponding 95\% confidence intervals. The 1000-image data set contains images from Category 0 to Category 9. The 2000-image data set contains images from all 20 categories. Training and test sets are of equal size.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>1000-Image Data Set</th>
<th>2000-Image Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILES</td>
<td>82.6 : [81.4, 83.7]</td>
<td>68.7 : [67.3, 70.1]</td>
</tr>
<tr>
<td>DD-SVM [16]</td>
<td>81.5 : [78.5, 84.5]</td>
<td>67.5 : [66.1, 68.9]</td>
</tr>
<tr>
<td>MI-SVM [2, 16]</td>
<td>74.7 : [74.1, 75.3]</td>
<td>54.6 : [53.1, 56.1]</td>
</tr>
<tr>
<td>( k )-means-SVM [17]</td>
<td>69.8 : [67.9, 71.7]</td>
<td>52.3 : [51.6, 52.9]</td>
</tr>
</tbody>
</table>
classification accuracy (over 5 randomly generated test sets) and the corresponding 95% confidence interval are presented in Figure 4. The average accuracy of DD-SVM drops quickly below 75% when \( d > 20 \), therefore, only the data associated with \( d \leq 20 \) are included in Figure 4.

Figure 4 shows that MILES and DD-SVM have similar classification accuracy when there is no labeling noise. As the noise level increases, the average classification accuracy of DD-SVM decreases rapidly, whereas the performance of MILES remains almost the same when \( d \) varies from 0 to 18. The lower bound of the 95% confidence interval for MILES is well above 90% even when 24% of training images have their labels negated. But for DD-SVM, the lower bound of the 95% confidence interval stays above 90% when the mislabeled images is less than 10%. This demonstrates the robustness of MILES against labeling noise. In contrast, the diverse density function in DD-SVM is very sensitive to instances in negative bags [16]. The diverse density value at a point is exponentially reduced if there is a single instance from a negative bag close to the point. Consequently, the local maximizers of the DD function are sensitive to labeling noise. MILES abandons the process of maximizing the DD function. The labeling noise can, in essence, be viewed as some “defective features” defined by the instances in the mislabeled bags. Because of the appropriate regularization in 1-norm SVM, the feature selection process tends to exclude those defective features. As a result, MILES is less sensitive to labeling noise.

C. Object Class Recognition

We carried out experiments on the Caltech data set used by Fergus et al. [21] and others [42], [5]. The goal is to learn and recognize object class models from unsegmented images. Each image is represented as a collection of “salient regions”. Objects are modeled as flexible constellations of the regions. The labels of the training images are given, i.e., it is known whether or not a training image contains a certain object from a particular class. However, the labels of the salient regions in each training image are unknown, so it is not clear which salient regions correspond to the object, and which are not. This learning situation can be viewed as a multiple-instance problem. We compared MILES with the Fergus method [21] as well as the approaches described in [42] and [5].

1) Experimental Setup: The image data set was downloaded from the Website of Robotics Research Group at University of Oxford\(^5\). It contains the following four object classes and background images: Airplanes (800 images), Cars (800 images), Faces (435 images), Motorbikes (800 images), and Background (900 general background and 1,370 road background).

We followed the descriptions in [21] for feature extraction. A brief summary of the process is given below. All images are converted to gray-scale. Salient regions are found using the method introduced by Kadir and Brady [26], which detects regions that are salient over both location and scale\(^6\). In our experiments, the scale varies between 5 and 50. Figure 5 shows some images randomly selected from five classes along with the detected salient regions. Each salient region is cropped from the image and rescaled to an image patch of size 11 \( \times \) 11 pixels so it corresponds to a 121-dimensional feature vector. Principal component analysis (PCA) is applied to all image patches for dimensionality reduction. Each patch is then represented by a vector of the coordinates within the first 15 principal components. The scale (radius of the salient region) and location of the salient region are also included. This leads to a total of 18 features for each image patch.

Images within each object class were partitioned in half to form a training set and a test set of equal size. Our partition is identical to that used in [21]. The background images were also divided equally into training and test sets. The parameter \( \mu \) was chosen to be 0.5. The other two parameters \( \sigma^2 \) and \( \lambda \) were selected according to a twofold cross-validation on the training set. We found that \( \lambda = 0.5 \) and \( \sigma^2 = 10^5 \) gave the minimum twofold cross-validation error for airplanes, faces, and motorbikes; while \( \lambda = 0.5 \) and \( \sigma^2 = 2 \times 10^5 \) worked best for cars.

2) Recognition Results: The recognition decision is binary, i.e., object present versus object absent. To be consistent with the experimental setting in [21], [42], and [5], images in the three object categories (Airplanes, Faces, and Motorbikes) were tested against the general background images, while the Cars images were tested against the road background. The performance is measured by the equal-error-rates point on the receiver-operating characteristic (ROC) curve, i.e., the point where the true positive rate is equal to the true negative rate. For example, a true positive rate of 90% at an equal-error-rates point means that 90% of the positive images (object present) are correctly classified, and 90% of the negative

\(^5\)These data sets are available at http://www.robots.ox.ac.uk/~vgg/data.html/.

\(^6\)We used the salient region detector implemented by Timor Kadir. We used Version 1.5 in the experiments. The software is available at http://www.robots.ox.ac.uk/~timork/salscale.html/.
Fig. 5. Images randomly selected from five classes: Airplanes (first column), Cars (second column), Faces (third column), Motorbikes (fourth column), and Background images (fifth column). Each circle in an image represents a salient region.

### TABLE V

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Airplanes</th>
<th>Cars (Rear)</th>
<th>Faces</th>
<th>Motorbikes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILES</td>
<td>98.0</td>
<td>94.5</td>
<td>99.5</td>
<td>96.7</td>
</tr>
<tr>
<td>Fergus et al. [21]</td>
<td>90.2</td>
<td>90.3</td>
<td>96.4</td>
<td>92.5</td>
</tr>
<tr>
<td>Opelt et al. [42]</td>
<td>88.9</td>
<td>90.1</td>
<td>93.5</td>
<td>92.2</td>
</tr>
<tr>
<td>Bar-Hillel et al. [5]</td>
<td>89.7</td>
<td><strong>97.7</strong></td>
<td>91.7</td>
<td>93.1</td>
</tr>
</tbody>
</table>

Comparing the object class recognition performance obtained using MILES with that of other methods. The numbers listed are the true positive rates (in %) at the equal-error-rates point on the ROC curve (i.e., true positive rate = true negative rate).

Images (background images) are also correctly classified. The error rate is therefore 10%. The ROC curve of MILES is obtained by varying the $b^*$ parameter in (10).

The comparison results are summarized in Table V. The boosting technique proposed by Bar-Hillel et al. [5] can take two different methods of selecting weak learners, hence, we included in Table V the best results for each category. MILES gives the best performance on three object classes: the error rates of MILES over Airplanes, Faces, and Motorbikes are around 20.4%, 13.9%, and 48%, respectively, of those of the second best (the Fergus method on Airplanes and Faces, the Bar-Hillel method on Motorbikes). The method proposed by Bar-Hillel et al. [5] gave the best results on Cars category. The overall recognition performance of MILES appears to be very competitive.

3) Selected Features and Instance Classification: Since the MILES classifier is determined entirely by the select features, these features describe the properties that are common to objects in the same class and separate objects from background clutters. According to (7), each selected feature is defined by a unique instance in the training bags. Next, we show some instances (or image patches) associated with the selected features. We divided the selected features into positive features and negative features according to the sign of their weights ($w_k^*$’s). In the experiments for all four object classes, it is observed that every positive feature is defined by an image patch from a positive image (i.e., containing a certain object) and every negative feature is defined by an image patch from a background image. It is interesting that MILES succeeded in doing this without explicit constraint. On the other hand, this also suggests that these data sets may contain much more information than is typically assumed in the context of MIL [48]. Unless this issue can be fully investigated, the above object class recognition results should be interpreted
cautiously.

Figure 6 shows 32 image patches related to positive features for each object class. Within each class, the image patches are arranged (from left to right, top to bottom) in descending order of the magnitude of the weight coefficients, which indicates the degree of importance for the selected features. Even though all image patches related to positive features were selected from positive images, some patches were not taken from a region corresponding to the object. We call these features false positive features. An image patch for a false positive feature is displayed with a bounding box in Figure 6. We also included in the figure the number of features before feature selection (i.e., the number of instances in all training bags), the number of negative features, the number of positive features, and the number of false positive features. As indicated by these numbers, the solution of MILES is very sparse. On average, more than 98% of the weights are zero. The results also seem to suggest that the feature selection process of MILES can, to some extent, capture the characteristics of an object class.

Next, we tested the instance classification algorithm proposed in Section III-B. Specifically, if an image is classified as containing a certain object, the contribution of each salient region within the image is evaluated using (11). A salient region is viewed as part of the object if its contribution is greater than \( \tau = \frac{\lambda}{b} \). Such salient regions are named positive regions. Figure 7 shows some randomly selected test images from the four object classes. For each block of images, images in the first row show all the detected salient regions, and the corresponding images in the second row show only the regions that are identified by MILES as parts of the object. The results suggest that MILES performs well on instance classification. The selected positive regions may provide generative clues in identifying the location of the object.

D. Computation Time

Compared with the methods based on the diverse density framework (DD-SVM [16]) and the Fergus method, MILES is significantly more efficient in terms of computational complexity. We summarized in Table VI the training time needed by MILES, DD-SVM, and the Fergus method on different data sets. The training of MILES, DD-SVM, and the Fergus method are performed on a 2.2GHz PC, a 700MHz PC, and a 2GHz PC [21], respectively. The time on MUSK1 and MUSK2 data sets is based on the total training time of tenfold cross-validation (the parameters were selected from twofold cross-validation with 51 values for \( \lambda \) and 17 values for \( \sigma^2 \)). The training time on the COREL data set is measured on a training set of 500 images (the parameters were selected from twofold...
Fig. 7. Sample test images that are classified as containing a targeted object. For each block of images, images in the first row show all the detected salient regions enclosed by circles, and the corresponding images in the second row show only the regions that are identified by MILES as parts of the object.

cross-validation with 11 values for $\lambda$ and 11 values for $\sigma^2$).

The longest training time among the four object classes of the Caltech data set is shown in Table VI (the parameters were chosen from twofold cross-validation with 6 values for $\lambda$ and 5 values for $\sigma^2$). Overall, the learning process of MILES is more than one order of magnitude faster than that of the other two methods.

VI. CONCLUSIONS AND FUTURE WORK

We have proposed a learning algorithm, MILES, which transforms a MIL problem to a supervised learning problem. A bag is embedded in a feature space defined by the instances in all the training bags where the coordinates of a bag represent its similarities to various instances. A feature selection tech-
nique using 1-norm SVM is applied to identify discriminative features (or instances) and construct bag classifiers at the same time. We tested MILES over the benchmark data sets taken from applications of drug activity prediction, image categorization, and object class recognition. In comparison with other methods, MILES demonstrates competitive classification accuracy on all three data sets. In addition, it is on average more than one order of magnitude faster in terms of learning speed than the Fergus method and a method based on the diverse density framework [16]. Moreover, it demonstrates good performance in instance classification and high robustness to the labeling noise.

MILES has several limitations:

- The performance of MILES depends on whether there are “useful” features among those defined by the instances in the training bags. Constraining the features to only those derived from the training bags reduces the search space, but may also hamper the performance on small data sets where it is possible that none of the instances is close to a “target point.” This may explain the observation that MILES performs better on leave-one-out test than on tenfold cross validation for the MUSK1 and MUSK2 data sets.
- In some applications, for example 3D object recognition [45], geometric constraints on the image patches are extremely useful in reducing the search space and improving the recognition accuracy. However, MILES is not designed to take advantage of this type of prior knowledge. The instance-based feature mapping cannot easily model the spatial relationships among the instances.
- The feature vectors generated by the mapping (7) are not sparse. Therefore, the LP requires the storage of a data matrix of size \((\ell^+ + \ell^-) \times n\) where \(\ell^+ + \ell^-\) is the number of training bags, and \(n\) is the number of instances in all the training bags. For some applications, \(n\) can be several orders of magnitude greater than \(\ell^+ + \ell^-\). Therefore, the storage requirement can be prohibitive. The needed storage space could be significantly reduced by an instance similarity measure that generates sparse features.

Continuations of this work could take several directions:

- The instance classification method can be improved. The current decision rule, which is based on a threshold \(-\frac{b^*}{\alpha^*}\), may identify many false positive instances, especially when the number of instances in a bag is large. The false positive rate may be reduced by optimizing the threshold.
- The mapping used in MILES can be viewed as feature aggregation for bags, i.e., take the maximum of each feature across instance in a bag. In this sense, MILES is closely related to the work by Gärtner et al. [22], in which several different aggregation operators were proposed to build a bag-level representation and subsequently a bag kernel. Therefore, it will be interesting to test different aggregation operators within the MILES framework.
- In some applications, the training data are given in a one-class setting. For example, the domain-based protein interaction inference problem typically only has positive training samples, namely, pairs of proteins that interact [62]. The interaction of two proteins is determined by the interaction of their domains. Therefore, if we view a pair of proteins as a bag and the domain-domain pairs as the instances in the bag, we have a multiple-instance representation. Multiple-instance learning with one-class training data is an interesting direction that we would like to pursue.

MILES can be integrated into a target tracking system for object detection. Evaluation of different feature selection techniques and investigation of the connection between the current approach and generative probabilistic methods are also interesting.

**ACKNOWLEDGMENTS**

Yixin Chen is supported by a Louisiana Board of Regents RCS program under Grant No. LBOR0077NRR00C, the US National Science Foundation EPSCoR program under Grant No. NSF/LEQSF(2005)-PFUND-39, The Research Institute for Children, and the University of New Orleans. James Z. Wang is supported by the US National Science Foundation under Grant Nos. IIS-0219272, IIS-0347148, and ANI-0202007, The Pennsylvania State University, and the PNC Foundation. The authors would like to thank the anonymous reviewers and the Associate Editor for their comments which have led to improvements of this paper. We would also like to thank Timor Kadir for providing the salient region detector and Rob Fergus for sharing the details on the object class recognition experiments.

**REFERENCES**


