In Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004), pages 550-555.

A Faster Algorithm for Generalized Multiple-Instance Learning

Qingping Tao   Stephen D. Scott
Department of Computer Science
University of Nebraska - Lincoln
115 Ferguson Hall
Lincoln, NE 68588–0115
{qtao, sscott }@cse.unl.edu

Abstract

In our prior work, we introduced a generalization of the multiple-instance learning (MIL) model in which a bag's label is not based on a single instance's proximity to a single target point. Rather, a bag is positive if and only if it contains a *collection* of instances, each near one of a *set* of target points. This generalized model is much more expressive than the conventional multiple-instance model, and our first algorithm in this model had significantly lower generalization error on several applications when compared to algorithms in the conventional MIL model. However, our learning algorithm for this model required significant time and memory to run. Here we present and empirically evaluate a new algorithm, testing it on data from drug discovery and content-based image retrieval. Our experimental results show that it has the same generalization ability as our previous algorithm, but requires much less computation time and memory.

Introduction

In multiple-instance learning (MIL), each example is a multiset (called a *bag*) of instances (points) rather than a single instance. In the conventional MIL model, each bag's (boolean or real) label is entirely determined by a single point in the bag, e.g. for boolean labels, the bag's label is a disjunction of the points' boolean labels, each of which is typically determined by the point's proximity to a single target point. This is not sufficient for some application areas. Thus in our prior work (Scott, Zhang, & Brown 2003), we introduced the generalized MIL (GMIL) model where the target concept is a *set* of points and the label for a bag is determined by a more general (non-disjunctive) function over the attributes. We also gave an algorithm that learns geometric concepts in the GMIL model and applied it to several application areas including content-based image retrieval, drug discovery, robot vision, and protein sequence analysis, comparing our results to those of DD (Maron & Lozano-Pérez 1998) and EMDD (Zhang & Goldman 2001), which are very successful algorithms in the conventional MIL model. Our algorithm's classification error rates were consistently competitive with those of DD and EMDD, and in several cases significantly better. This corroborated our belief that the GMIL model is better suited for some applications than the conventional MIL model.

However, the basic learning algorithm we gave for the GMIL model is inherently inefficient, requiring hours of computation time and hundreds of megabytes of memory to train. Here we present a much faster and more memory-efficient algorithm that can handle the requirements of real pattern recognition systems. Our experimental results show that it has the same generalization ability as the basic algorithm, but needs much less computation time and memory.

The rest of this paper is organized as follows. In the following section, we discuss related work in conventional MIL. Then we describe our GMIL model and our original algorithm for learning in GMIL. After that, we describe our new, faster GMIL algorithm. In the experimental section, we evaluate our algorithm on two application areas (discovery of antagonist drugs and content-based image retrieval) and compare it to the old GMIL algorithm. In the last section we give our conclusions and future work.

The Conventional MIL Model

The multi-instance learning model was introduced by Dietterich, Lathrop, & Lozano-Perez (1997). In their model, each bag is classified as positive if and only if at least one of its elements is labeled as positive by the target concept. Their work was motivated by the problem of predicting whether a molecule would bind at a particular site. They argued empirically that axis-parallel boxes are good hypotheses for this and other similar learning problems. This MIL model has been extensively studied (Auer 1997; Andrews, Tsochantaridis, & Hofmann 2002), along with extensions for real-valued labels (Dooly *et al.* 2002; Ray & Page 2001).

In most MIL work, the label of a bag depends only on the label of a single point, and the label of each point typically is assumed to depend on a single target point. Exceptions include the algorithms DD (Maron & Lozano-Pérez 1998) and EMDD (Zhang & Goldman 2001) in which a target concept can be a disjunction over multiple points. However, such disjunctions have to be severely restricted for the sake of computational efficiency. In other work, De Raedt (1998) generalizes MIL in the context of inductive logic programming and defines an interesting framework connecting many forms of learning, allowing relations between instances. However, the transformations he gives between the models have exponential time and space complexity. Also, such generalizations must be explicitly tuned.

Our Generalized MIL model

In the GMIL model, the target concept is a set of points $C = \{c_1, \ldots, c_k\}$, and the label for a bag $B = \{p_1, \ldots, p_n\}$ is positive if and only if there is a subset of r target points $C' = \{c_{i_1}, \ldots, c_{i_r}\} \subseteq C$ such that each $c_{i_j} \in C'$ is near some point in B, where "near" is defined as within a certain distance under some weighted norm. Here r is a threshold indicating the minimum number of target points that must each be "hit" by some point from B (the same point in B could hit multiple target points). In other words, if we define a boolean attribute a_i for each target point c_i that is 1 if there exists a point $p_j \in B$ near it and 0 otherwise, then the bag's label is some r-of-k threshold function over the attributes (so there are k relevant attributes and B's label is 1 iff at least r of these attributes are 1).

We then extended our GMIL model in the following way. If we let $\bar{C} = \{\bar{c}_1, \ldots, \bar{c}_{k'}\}$ be a set of "repulsion" points, then we can require a positive bag to *not* have any points near each point of $\bar{C}' = \{\bar{c}'_{i_1}, \ldots, \bar{c}'_{i_{r'}}\} \subseteq \bar{C}$ in addition to having a point near each point in C'. This means that to be positive, certain feature values have to be present in some points of bag B, but also certain feature values must be *absent*. This proved useful in some of our experiments.

Our Previous Algorithm for GMIL

Our previous algorithm (which we call GMIL-1) for learning geometric concepts in the GMIL model assumes that each bag is a multiset of at most n points from the finite, discretized space $\mathcal{X} = \{1, \ldots, s\}^d$. Our algorithm then enumerates all the possible $N = (s(s+1)/2)^d$ axis-parallel boxes in $\mathcal{X}$ and creates two attributes for each box b: a_b and $\bar{a}_b$. Given a bag $B \in \mathcal{X}^n$, the algorithm sets $a_b = 1$ if some point from B lies in b and $a_b = 0$ otherwise. Then $\bar{a}_b = 1 - a_b$. These $2N$ attributes are then given to Winnow[1] (Littlestone 1988), which learns a linear threshold unit.

If each bag's label is given by some r-of-k threshold function over the attributes (so there are k relevant attributes and B's label is 1 iff at least r of these attributes are 1), then applying well-known results allows us to conclude that Winnow's generalization error when learning such functions is bounded by a polynomial in k, r, d, and $\log s$. However, its time complexity is $\Omega(s^{2d})$ per trial, which is exponential in both $\log s$ (the number of bits needed to describe each instance) and d.

To overcome the exponential dependence on $\log s$, the algorithm partitions the set of N boxes into *groups* such that it is guaranteed that for each box b in a group G, all attributes a_b have the same weight in Winnow, as do all attributes $\bar{a}_b$. Thus the algorithm maintains only one representative box per group G and exactly computes Winnow's weighted sum (i.e. the quantity compared to the threshold when making predictions) by summing the products of each group's weight and its size:

$$\sum_{i=1}^{N} a_i \, w_{a_i} + \bar{a}_i \, w_{\bar{a}_i} = \sum_{G \in \Gamma} |G| \left(a_G \, w_{a_G} + \bar{a}_G \, w_{\bar{a}_G} \right),$$

where $|G|$ is the number of boxes in group G, a_G is the attribute for group G's representative box, and Γ is the set of all groups. The set of groups is built by using the union of points from all bags[2] to rectilinearly partition $\mathcal{X}$ (see Figure 1). The result is a set of *regions* such that any pair of boxes b_i and b_j are in the same group if both have their "lower left" corners in region W and their "upper right" corners in region Z. It is easy to see that when the groups are constructed this way, for each pair of boxes b and b' in the same group G, b and b' contain the same subset of points from all bags. Thus $a_b = a_{b'}$ and $\bar{a}_b = \bar{a}_{b'}$ for all bags, and all the Winnow weight updates are the same for attributes a_b and $a_{b'}$ as well as $\bar{a}_b$ and $\bar{a}_{b'}$. Using this construction, at most $O(m^{2d+1})$ groups are built, where m is the number of points used to partition the space. Thus the exponential dependence on $\log s$ is removed, but the exponential dependence on d remains.

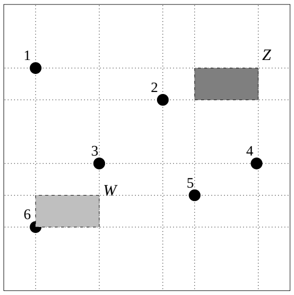


Figure 1: An example of how GMIL-1 constructs its groups.

We applied GMIL-1 on four different application areas: robot vision, content-based image retrieval, protein sequence identification, and drug discovery. Our experiments showed that GMIL-1 is competitive with (and often better than) DD (Maron & Lozano-Pérez 1998) and EMDD (Zhang & Goldman 2001), which are two of the better algorithms in the conventional MIL model (some of these previous results are summarized in the experimental section). But in these experiments GMIL-1 built around 4–7 million groups, requiring 500–700 Mb of memory and around 30 hours to train. However, we also found that after training was completed, most of the groups could be thrown out with no impact on prediction error. Indeed, after training we could always discard at least 80% (typically more than

[1] Winnow is very similar to the Perceptron algorithm but updates its weights multiplicatively rather than additively, which often yields much faster convergence, especially in cases like ours when most inputs are irrelevant.

[2] When applying GMIL-1 on some applications, we also used clustering on the training set to preprocess the data to reduce the number of points m used to build the groups. This was done with data sets where the dimension $d > 2$.

97%) of the groups with the lowest weight in Winnow and have a hypothesis that predicts nearly as well as the original. This suggests that only a relatively very small subset of groups is needed to learn these concepts well, which motivates our design of GMIL-2. In the following section, we will describe GMIL-2, which has the same generalization ability but runs much faster and uses less memory than GMIL-1.

The Algorithm GMIL-2

GMIL-2 is similar to GMIL-1 in that it groups boxes together, assigns boolean attributes to these groups, and gives these attributes to Winnow to learn an r-of-k threshold function over these attributes. The key difference from GMIL-1 is how GMIL-2 builds the groups. First, rather than using the union of points from all training bags, GMIL-2 uses a set Ψ of representative points that capture the distribution of the training bags. Second, rather than basing the construction of the groups on a rectilinear partitioning of $\mathcal{X}$, GMIL-2 directly considers all subsets of Ψ.

The Basic Algorithm

Recall that the reason we can group boxes together in GMIL-1 is that the boxes in each group contain the same set of points. For example, in Figure 1, the group defined by regions W and Z is a set of boxes that contains and only contains points 2, 3 and 5. Instead of representing this group as a box, we can represent it as a set $S = \{2, 3, 5\}$. So in this representation, any subset of $\{1, 2, 3, 4, 5, 6\}$ is a potential group. But not all of these subsets are valid groups, e.g. $\{4, 6\}$ since any box containing points 4 and 6 must also contain points 3 and 5. We can easily check if S is valid by finding the smallest box containing S and comparing S with the point set contained by the box (described below).

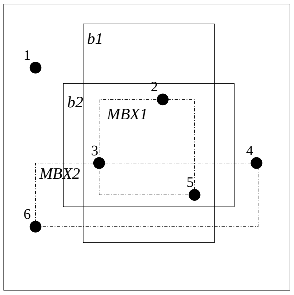


Figure 2: An example of how GMIL-2 constructs its groups.

Suppose a set of points Ψ is given as representatives of a distribution of points in a d-dimensional feature space $\mathcal{X}$. Let S be a subset of Ψ. We call the smallest box that contains S the minimum bounding box (MBX) of S. A subset $S \subseteq \Psi$ is *valid* if there is no point $p \in \Psi \setminus S$ that is contained by the MBX of S, otherwise *invalid*. Each valid subset S represents a group G_S of boxes that contain and only contain S in Ψ. GMIL-2 enumerates all such groups. For each group G_S, we define two attributes: l_S and $\bar{l}_S$. Given a bag $B \in \mathcal{X}^n$, the algorithm sets $l_S = 1$ and $\bar{l}_S = 0$ if for some point p from B, the MBX of $\{p\} \cup S$ does not contain any point from $\Psi \setminus S$. Otherwise it sets $l_S = 0$ and $\bar{l}_S = 1$. These attributes are then given to Winnow.

Building the Group Set

We now describe our algorithm to build the set of groups $\Gamma = \{G_S \mid S \subseteq \Psi \text{ and } S \text{ is valid}\}$. Obviously, brute-force enumeration and testing of all $2^{|\Psi|}$ subsets of Ψ is impractical for even moderately sized Ψ. Thus our algorithm for building the set of groups exploits the geometric property that many of these $2^{|\Psi|}$ subsets will be invalid. Our algorithm is similar to breadth-first searching of the set of all possible groups. This BFS algorithm also leads to a heuristic for choosing Ψ in such a way that allows $|\Psi|$ to be quite large while keeping $|\Gamma|$ small, e.g. in our experiments, we used $|\Psi| \in [5, 105]$ but had $|\Gamma| < 10^5$.

Our BFS algorithm first puts into Γ the empty set and all singleton subsets $\{p\}$ for each $p \in \Psi$ because all such subsets are valid. After that, it identifies all valid subsets with size 2 and adds them to Γ, then valid subsets with size 3, and so on up to size $|\Psi|$. To find all valid subsets with size $k+1$, BFS looks at all size-k subsets. For each size-k subset S', BFS considers each point $p \in \Psi \setminus S'$. Each point p is added to S' to form $S'_p = S' \cup \{p\}$. Then BFS builds the MBX b_p for each S'_p and adds to Ψ the largest set $S''_p \subseteq \Psi$ contained by b_p. (Note that we might have $|S''_p| > k+1$. That set is still added to Ψ.)[3] Since any size-$(k+1)$ valid subset can be built by adding one or more points to a valid subset with size less than $k+1$, the algorithm is guaranteed to find all valid size-$(k+1)$ subsets. The time complexity of our BFS group-building algorithm is $O(|\Psi||\Gamma|)$.

Selecting Representatives

Obviously the more representatives that are used, the more accurately they represent the true distribution of the points in the bags. This tempts one to let Ψ be the union of the points in all the training bags. However this could make $|\Gamma|$ prohibitively large. Thus we need to reduce $|\Psi|$. One way is clustering all the points in the training bags and setting Ψ to the set of point representatives of these clusters.

Intuitively, the larger Ψ is, the better our resolution will be of the instance space and the better our algorithm will learn. However, because point representatives of clusters will typically lie in general position (i.e. not lie on the same low-dimensional hyperplanes), increasing $|\Psi|$ using cluster representatives only will dramatically increase $|\Gamma|$. To see this, note that if most subsets of the points in Figure 2 were collinear instead of in general position, the number of distinct valid groups would decrease significantly. This motivates our method of building Ψ: first we set Ψ to be the point representatives of a small number of clusters (e.g. 5). Then we add many random points (e.g. 100) to Ψ that are collinear with these point representatives.

Specifically, first we cluster all points in training bags into N clusters. Then we build a grid $\mathcal{G}$ such that the intervals on

[3] For example, in Figure 2, adding 2 into $\{3, 5\}$ yields a valid subset $S'_2 = S''_2 = \{2, 3, 5\}$. But adding 4 into $\{3, 6\}$ yields $S'_4 = \{3, 4, 6\}$ and $S''_4 = \{3, 4, 5, 6\}$. Then S''_4 is added to Ψ.

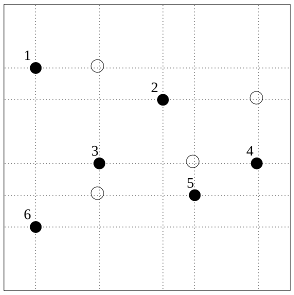


Figure 3: An example of how GMIL-2 adds additional points.

each dimension are defined by the projections of N clusters on this dimension. Let $\Omega_{\mathcal{G}}$ be the set of all crossing points on $\mathcal{G}$. We choose the N clusters and K additional random points from $\Omega_{\mathcal{G}}$ as our representatives. For example, in Figure 3, we add to the $N = 6$ cluster representatives (black points)$K = 4$ additional (white) points from $\Omega_{\mathcal{G}}$.

Contrasting GMIL-1 with GMIL-2

Obviously the only difference in the running of both learning algorithms is the makeup of the group set Γ. Here we present some more detailed comparisons between the group sets under various special cases of the representative sets.

First consider when Ψ for GMIL-2 is the union of all points in all training bags. Let S be a valid subset of Ψ and let Γ_1^S be the union of all groups for GMIL-1 whose boxes contain exactly S. Then the GMIL-2 group represented by S is exactly equal to Γ_1^S, $l_S = a_G$ and $\bar{l}_S = \bar{a}_G$ for all training bags, and the two algorithms are equivalent. The only difference is that the set of groups (i.e. the set of attributes given to Winnow) is smaller for GMIL-2.

Now consider the case when GMIL-1 builds its grid on a set of representative points rather than the union of all points in all bags (see Footnote 2). Now since the points in Figure 2 are only representatives, it is possible that there is e.g. a point x from the training bags that lies in box $b1$ but not in $b2$. From its use of the grid to build its group set, GMIL-1's groups can recognize this fact, but GMIL-2's subset-based construction of Γ prevents this. Thus GMIL-2 operates in a space with lower resolution than GMIL-1, and the only way it can match GMIL-1's resolution is to set $\Psi = \Omega_{\mathcal{G}}$. If this is the case, then the two algorithms are again equivalent.

Thus for special cases, the two algorithms build equivalent group sets. If only a subset of all training points or grid points is used as representative points, GMIL-2 can be treated as an approximation to GMIL-1. The time complexity of GMIL-2 is polynomial in d but in the worst case exponential in $|\Psi|$. Thus we can now scale up our algorithm to high dimensions without much of a time complexity increase. However, we obviously cannot use all points from the training bags, which means we learn on a reduced resolution. This may cause a significant increase in prediction error for large d (we are currently exploring this). But whether or not GMIL-2 is suitable for high-dimensional data, our experiments on smaller dimensional data confirm that GMIL-2 is significantly faster than GMIL-1. Further, it turns out that the low resolution does not hurt in practice.

Experimental Results

We now describe our experiments comparing GMIL-2 with GMIL-1 in terms of generalization error and computation costs. Our results show that GMIL-2 is much more efficient than GMIL-1 with no significant change in prediction error. We also compare these results with prediction errors of DD and EMDD on the same data sets.

The Application Areas

Binding Affinity/Antagonist Drugs Dietterich, Lathrop, & Lozano-Perez (1997) introduced the conventional MIL model motivated by predicting whether a conformation of a particular molecule would bind to a single site in another molecule. But an open problem is how to predict "antagonist drugs", whose jobs are to bind at multiple sites in a single molecule by fitting in several of them simultaneously.

We used a generalization of the synthetic data of Dooly *et al.* (2002) to reflect the notion of antagonist drugs. We used their modified data generator to build ten 5-dimensional data sets with 4 sub-targets[4]. For each data set, we first randomly generated 4 sub-targets and then used the target to label a training set and test set, each of size 200.

Content-Based Image Retrieval In content-based image retrieval (CBIR), the user presents examples of desired images, and the task is to determine commonalities among the query images. Maron & Ratan (1998) explored the use of conventional MIL for CBIR. They filtered and subsampled the images and then extracted "blobs" (groups of m adjacent pixels), which were mapped to one point in a bag. Then they used the MIL algorithm diverse density (DD) (Maron & Lozano-Pérez 1998) to learn a hypothesis and find candidate images in the database. This work was extended by Zhang *et al.* (2002), who observed that in their experiments, it was likely that not one but several target points (in a disjunctive form) were responsible for labeling the examples. In addition, it is arguable that if the target concept were conjunctive (e.g. the desired images contain a field *and* a sky), then the standard MIL model will not work and a more expressive hypothesis is needed.

We experimented with two CBIR tasks[5]. One is Zhang et al.'s "sunset" task: to distinguish images containing sunsets from those not containing sunsets. Like Zhang et al., we built 30 random testing sets of 720 examples (120 positives and 600 negatives): 150 negatives each from the waterfall, mountain, field, and flower sets. Each of 30 training sets consisted of 50 positives and 50 negatives.

Another task is to test a conjunctive CBIR concept, where the goal is to distinguish images containing a field with no sky from those containing a field and sky or containing no

[4] Each sub-target corresponds to a site that a molecule must bind to in order to be considered a positive bag.

[5] Based on data from (Wang, Li, & Wiederhold 2001), the Corel Image Suite, and www.webshots.com.

field. We relabeled Zhang et al.'s field images from positive to negative those that contained the sky. Each training set had 6 bags of each of flower, mountain, sunset, and waterfall for negatives, and had around 30 fields, 6 of them negative and the rest positive. Each negative test set had 150 bags of each of flower, mountain, sunset, and waterfall. Also, each test set had 120 fields, around 50 serving as positives and the remainder as negatives.

Comparison of Prediction Error

For both GMIL-1 and GMIL-2, we k-means clustered all training sets and used the clusters' point representatives when building group sets (see Footnote 2). The drug discovery data was clustered into $m = 5$ clusters and $m = 6$ clusters for CBIR. For building groups for GMIL-2, we added to Ψ a randomly-chosen size-K subset of $\Omega_{\mathcal{G}}$, where K was varied to measure the effect of reduced resolution on generalization error. In all experiments, we trained both algorithms until they achieved zero training error or until they trained for 100 rounds.

In Figure 4, all plots show the same trend. When the number of additional points K is small, GMIL-2 has very high training error and testing error. But by increasing K, we see a dramatic decrease in the false negative error rates[6]. Once K exceeds some value (which depends on the data set), we see that GMIL-2's performance for both FP and FN error rates is statistically indistinguishable from those of GMIL-1. I.e. GMIL-2 learned a hypothesis that is statistically the same as GMIL-1's. In addition, the value of K required to do this is quite small (10–60), and much less than $|\Omega_{\mathcal{G}}|$ (as discussed earlier, $K = |\Omega_{\mathcal{G}}|$ would make the two algorithms exactly equivalent). For example, for the drug discovery data GMIL-2's prediction error was indistinguishable from GMIL-1's using $|\Psi| = 5 + K = 65$ representatives while GMIL-1 used 16807 grid points. Thus GMIL-2 can run on a relatively small number of representatives without a significant increase in prediction error[7].

Comparison of Computation Costs

The computation time of both GMIL-1 and GMIL-2 depends on the number of groups and how fast they converge. Their memory requirements also depend on the number of groups. In Table 1(a), we compare the number of groups built by GMIL-1 to the number built by GMIL-2 with $K = 100$. It is obvious that GMIL-2 built much smaller group sets than GMIL-1, which means GMIL-2 used much less memory and ran much faster than GMIL-1. We also compared the average number of trials required for each algorithm to converge to zero error on the training set. From Table 1(b), we see that GMIL-2 converged in about the same number or even fewer trials than GMIL-1. So generally speaking, they have the same rate of convergence. In all these experiments, GMIL-1 required 500-700 Mb of memory and ran for around 30 hours to train on a single data set. In contrast, GMIL-2 used less than 50 Mb of memory and finished training in less than one hour on average.

Table 1: Comparison of the computation costs of GMIL-1 and GMIL-2.

(a) The average number of groups generated.

Data Sets	GMIL-1	GMIL-2	Savings (%)
Drug discovery	4.08×10^6	6.86×10^4	98.3
CBIR: sunset	3.89×10^6	6.19×10^4	98.4
CBIR: conjunctive	5.75×10^6	8.46×10^4	98.5

(b) The average number of trials in which the algorithms converged.

Data Sets	GMIL-1	GMIL-2
Drug discovery	26.8	34.0
CBIR: sunset	41.7	29.0
CBIR: conjunctive	21.4	16.8

Comparison to DD and EMDD

We now compare generalization error of GMIL-1 and GMIL-2 (with $K = 100$) to those of EMDD and DD (that work in the conventional MIL model). EMDD and DD were run on the original, unclustered data, so they have full resolution, in contrast to the GMIL algorithms. Results are presented in Table 2. For drug discovery, the FP errors of the two GMIL algorithms are less than DD and EMDD, but their FN errors are higher. We suspect that some of the FN error comes from the data's low resolution due to clustering in forming the grid. In the sunset task of CBIR, all four algorithms have similar FP and FN error; none of them shows a significant advantage over the others. However, for the conjunctive task, GMIL-1 and GMIL-2 achieved the lowest FP and FN errors. This occurred despite the potential loss of information due to clustering. Thus we see that conjunctive CBIR concepts benefit from generalized MIL. Also we notice that GMIL-2 has a little bit higher prediction error but again shows almost same generalization ability as GMIL-1 when compared to DD and EMDD.

Table 2: Generalization error of GMIL-1, GMIL-2, DD, and EMDD.

FP error.

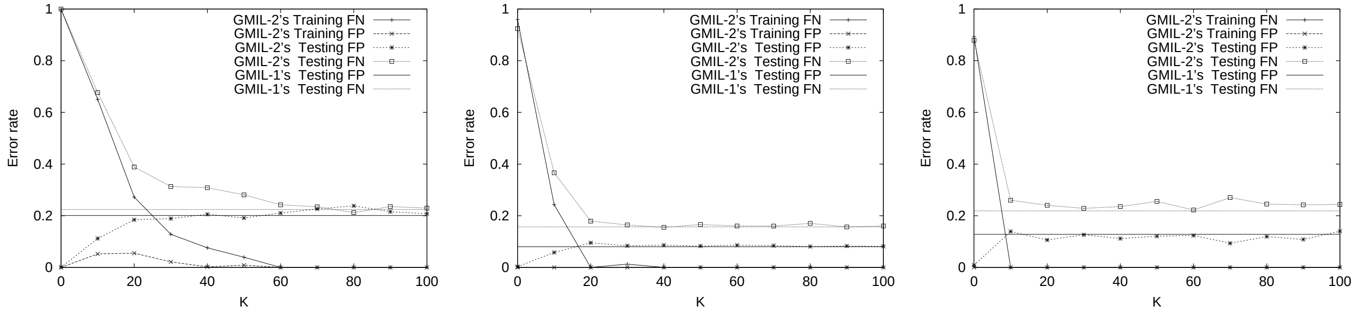
Data Sets	GMIL-2	GMIL-1	DD	EMDD
Drug discovery	0.208	0.201	0.274	0.263
CBIR: sunset	0.082	0.080	0.078	0.082
CBIR: conjunctive	0.140	0.128	0.173	0.213

FN error.

Data Sets	GMIL-2	GMIL-1	DD	EMDD
Drug discovery	0.230	0.224	0.108	0.109
CBIR: sunset	0.160	0.157	0.168	0.166
CBIR: conjunctive	0.244	0.219	0.282	0.244

[6] For small K, false positive error is 0 because the initial weights in Winnow cause it to always predict negative with using low-resolution inputs.

[7] We also compared GMIL-1 and GMIL-2 on robot vision data, where all training bags (with no clustering) were used to build GMIL-1's grid on 185344 points and GMIL-2 used $|\Psi| = 300$ points. Our results were similar in that GMIL-2's FP and FN rates were statistically indistinguishable from those of GMIL-1.



(a) Drug discovery.

(b) CBIR: sunset.

(c) CBIR: conjunctive.

Figure 4: Comparison of prediction error rates of GMIL-1 and GMIL-2 while varying K (the number of additional points used by GMIL-2).

Conclusions and Future Work

While the standard MIL model is powerful, there exist applications with natural target concepts that cannot be represented. This is what motivated our prior introduction of the GMIL model, along with the algorithm GMIL-1 to learn geometric concepts in it. Here we presented GMIL-2, a much faster algorithm than GMIL-1. GMIL-2 uses a compact group set based on a set of representative points. Our experimental results show that it has the same generalization ability as GMIL-1 and needs much less time and memory to train.

In future work, we will apply our new algorithm to protein sequence data to further compare it to GMIL-1. We will also look at applications with relatively high dimension, e.g. $d = 100$, to see if the reduced resolution has an adverse effect on prediction error when d is large. We will also explore kernel techniques (for use in a support vector machine) to quickly compute the weighted sum of the inputs, entirely eliminating the need to enumerate the groups and yielding a more scalable algorithm.

Acknowledgments

The authors thank Jun Zhang for her help with the implementation, Brian Pinette for the robot data, Qi Zhang, Sally Goldman, and James Wang for the CBIR data (indirectly from James Wang, Corel, and webshots.com), Qi Zhang for his EMDD/DD code and affinity data generator, and Josh Brown for his helpful discussions. This research was funded in part by NSF grants CCR-0092761, EPS-0091900, and a grant from the NU Foundation. It was also supported in part by NIH Grant Number RR-P20 RR17675 from the IDeA program of the National Center for Research Resources.

References

Andrews, S.; Tsochantaridis, I.; and Hofmann, T. 2002. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*.

Auer, P. 1997. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proc. 14th International Conference on Machine Learning*, 21–29. Morgan Kaufmann.

De Raedt, L. 1998. Attribute-value learning versus inductive logic programming: The missing links. In *Proc. 8th International Conference on Inductive Logic Programming*. Springer Verlag.

Dietterich, T. G.; Lathrop, R. H.; and Lozano-Perez, T. 1997. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence* 89(1–2):31–71.

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

Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2:285–318.

Maron, O., and Lozano-Pérez, T. 1998. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*.

Maron, O., and Ratan, A. L. 1998. Multiple-instance learning for natural scene classification. In *Proc. 15th International Conf. on Machine Learning*, 341–349. Morgan Kaufmann, San Francisco, CA.

Ray, S., and Page, D. 2001. Multiple instance regression. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 425–432.

Scott, S. D.; Zhang, J.; and Brown, J. 2003. On generalized multiple-instance learning. Technical Report UNL-CSE-2003-5, Dept. of Computer Science, University of Nebraska.

Wang, J. Z.; Li, J.; and Wiederhold, G. 2001. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(9):947–963.

Zhang, Q., and Goldman, S. A. 2001. EM-DD: An improved multiple-instance learning technique. In *Neural Information Processing Systems 14*.

Zhang, Q.; Goldman, S. A.; Yu, W.; and Fritts, J. E. 2002. Content-based image retrieval using multiple-instance learning. In *Proc. 19th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA.