Kernels for Generalized Multiple-Instance Learning

Qingping Tao, Stephen Scott, N. V. Vinodchandran, Thomas Takeo Osugi, Brandon Mueller

Preliminary versions of this work appeared in Tao et al. [1], [2].

Qingping Tao is with GC Image, LLC, P.O. Box 57403, Lincoln, NE 68505-7403.

Stephen Scott and N. V. Vinodchandran are with the Dept. of Computer Science, 256 Avery Hall, University of Nebraska, Lincoln, NE 68588-0115, {sscott,vinod}@cse.unl.edu.

Thomas Takeo Osugi is with JET in Suehiro-cho 2-ku, Oumu-Cho, Monbetsu-gun, Hokkaido 098-1702, Japan.

Brandon Mueller is with The Gallup Organization, 1001 Gallup Drive, Omaha, NE 68102.

Abstract

The multiple-instance learning (MIL) model has been successful in numerous application areas. Recently, a generalization of this model and an algorithm for it were introduced, showing significant advantages over the conventional MIL model on certain application areas. Unfortunately, that algorithm is not scalable to high dimensions. We adapt that algorithm to one using a support vector machine with our new kernel $k_\wedge$. This reduces the time complexity from exponential in the dimension to polynomial. Computing our new kernel is equivalent to counting the number of boxes in a discrete, bounded space that contain at least one point from each of two multisets. We show that this problem is #P-complete, but then give a fully polynomial randomized approximation scheme (FPRAS) for it. We then extend $k_\wedge$ by enriching its representation into a new kernel $k_{\min}$, and also consider a normalized version of $k_\wedge$ that we call $k_{\wedge/\vee}$ (which may or may not not be a kernel, but whose approximation yielded positive semidefinite Gram matrices in practice). We then empirically evaluate all three measures on data from content-based image retrieval, biological sequence analysis, and the Musk data sets. We found that our kernels performed well on all data sets relative to algorithms in the conventional MIL model.

Index Terms

kernels, support vector machines, generalized multiple-instance learning, content-based image retrieval, biological sequence analysis, fully polynomial randomized approximation schemes

I. Introduction

Dietterich et al. [3] introduced the multiple-instance learning (MIL) model motivated by the problem of predicting whether a molecule would bind at a particular site. Since shape of a molecule largely determines binding affinity, they represented each molecule by a high-dimensional vector that describes its shape, and labeled molecules that bind at a site as positive examples and those that do not bind as negative. Then their algorithm learned an axis-parallel box that distinguishes the positives from the negatives. The motivation for the MIL model is the fact that a single molecule can have multiple conformations (shapes), and only one conformation need bind at the site for the molecule to be considered positive. Thus when an example is negative, all conformations in it are negative, but if an example is positive, then at least one conformation of the set is positive, and the learner does not know which one(s). Since its introduction, the MIL model has been applied to content-based image retrieval [4], [5], [6], [7], [8], where each instance in a multi-instance example (*bag*) represents a feature of an image, and it is not known

which feature corresponds to the content the user wants to retrieve. As with binding prediction, the MIL model used for content-based image retrieval typically assumes that the label of a bag is a disjunction of the labels of the instances in the bag. I.e. a bag is labeled positive if and only if at least one of its instances is labeled positive by the target function (typically assumed to be a single point or a single axis-parallel box).

Recently, Scott et al. [9] generalized the MIL model, allowing an example's label to be represented in a much more general fashion than as a simple disjunction. Specifically, in Scott et al.'s generalization, the target concept can stipulate that in order to be positive, an example must have points near each of a *set* C of points *and* not near each of a set $\bar{C}$ of points. Note that bag labels in the conventional (disjunctive) MIL model are exactly determined solely by the labels of the individual instances of each bag. In contrast, this information is insufficient to determine a bag's label in Scott et al.'s generalization. This is because their model bases its bag labels on how many target points are "hit" or "missed."

Scott et al. then adapted an algorithm of Goldman et al. [10] to learn concepts in this new model. They empirically evaluated this algorithm (referred to here as GMIL-1) on problems from robot vision, content-based image retrieval, binding affinity, and biological sequence analysis. In all experiments, GMIL-1 was competitive with algorithms from the conventional MIL model. Further, on problems requiring the labeling function to be more general than a disjunction, GMIL-1 showed a significant advantage in generalization performance.

GMIL-1 works by first explicitly enumerating all axis-parallel boxes in the space $\{0, \ldots, s\}^d$, where d is the number of dimensions and $s+1$ is the number of discrete values in each dimension. Then it assigns boolean attributes to these boxes, and gives these attributes to Littlestone's algorithm Winnow [11], which learns a linear threshold unit. The time complexity of this algorithm is exponential in d, which obviously limits the applicability of GMIL-1. While there has been progress in developing heuristics to significantly speed up this algorithm in practice [12], the algorithm is still limited in its scalability. For example, the popular multi-instance benchmark data set Musk has over 160 dimensions, yielding over 10^{600} features for Winnow to process.

A popular means to implement exponentially large feature mappings is to use a support vector machine with a kernel that implicitly performs the mapping. We show that a kernel $k_\wedge$ exists that exactly corresponds to the feature mapping used by GMIL-1. To compute the kernel, one takes two bags of points P and Q and counts the number of boxes defined on $\{0, \ldots, s\}^d$ that contain

at least one point from P and at least one point from Q. We first show that this problem is #P-complete, and then present a fully polynomial randomized approximation scheme (FPRAS) for it. Since the values of $k_\wedge$ can be quite large, we also consider a normalized version of $k_\wedge$, which we call $k_{\wedge/\vee}(P,Q) = k_\wedge(P,Q)/k_\vee(P,Q)$. I.e., we divide the number of boxes containing a point from bags P and Q (given by $k_\wedge$) by the number containing a point from P or Q (given by $k_\vee$). The intuition behind this is to reduce inflated counts caused by large bags. As with $k_\wedge$, $k_{\wedge/\vee}$ is #P-complete to compute and has a FPRAS. It is unknown if $k_{\wedge/\vee}$ is a true kernel, though most of its approximate Gram matrices we computed were positive semidefinite.

A potential issue with our methods is that even when the kernel is positive semidefinite, our approximations may not be. We show this in our experimental results, where, depending on the quality of the approximation, the number of negative eigenvalues of the resulting Gram matrix varied from 0 to nearly 26% of the total number of eigenvalues. However, this posed no trouble for the SVM we used in our experiments; in fact, the SVM with our kernel approximations frequently performed competitively (and often outperformed) other methods on the same data. Indeed, there are other examples of successful use in SVMs of similarity measures that are not positive semidefinite, such as the sigmoid (hyperbolic tangent) function [13]. Further, in his detailed geometric study of SVMs with kernels that are not positive semidefinite, Haasdonk [14] showed that such SVMs can still be very effective learning algorithms. He starts by defining a pseudo-Euclidean (pE) space and formulating a (non-convex) SVM-like optimization problem whose solution(s) bisect the line segment(s) connecting the two closest points from the convex hulls of the positive and negative training examples in the pE space. He then goes on to relate solutions to this problem to that discovered by a conventional SVM. This gives further evidence that SVMs can be successful with kernels that are not positive semidefinite, especially if the negative eigenvalues of the Gram matrix are relatively small in number and in magnitude, which is the case for our approximate kernels.

Scott et al.'s GMIL model could be further generalized along the lines of Weidmann et al. [15]. Another of our contributions is a new remapping that generalizes Weidmann et al.'s "count-based" GMIL model and a kernel $k_{\min}$ that corresponds to that mapping. We then show that, as with $k_\wedge$, $k_{\min}$ is #P-complete to compute, so we give a FPRAS for it. In evaluating $k_{\min}$ on the same data sets as $k_\wedge$ and $k_{\wedge/\vee}$, we found that $k_{\min}$ can generalize better than $k_\wedge$ for a learning task in content-based image retrieval, but there is little room for improvement in the other learning

tasks we tested.

Our approach to learning in this generalized multiple-instance setting is to use our specialized kernels with standard SVMs that are designed for single-instance learning. Andrews et al. [16] use an alternative approach in the conventional MIL model, which is to reformulate an SVM optimization problem to directly operate on multiple-instance data. They do this by leveraging the definition of conventional MIL: all instances in a negatively-labeled bag are themselves negative and at least one instance of each positively-labeled bag is itself positive. Andrews et al. thus assign each individual instance a tentative label, and then use these tentative labels in a conventional SVM optimization problem with the additional (integer) constraints that at least one instance per positive bag is positive.

The approach of Andrews et al. has the advantage of allowing for any single-instance kernel (e.g. Gaussian) to be used. However, it is not clear if an SVM can be reformulated in a similar fashion for Scott et al.'s or Weidmann et al.'s generalized MIL models. This is because their models have multiple target points, and the target concept is defined by how many times each target point is "hit." Thus even if one exactly knew the individual label of each instance (i.e. whether it was "near" a target point), one could not infer the bag's label from merely that information. This makes it difficult to capture the necessary information in the form of additional constraints in an optimization problem.

The rest of this paper is organized as follows. In the next section we introduce some notation. In Section III we describe the conventional MIL model and then present Scott et al.'s generalization of it, as well as their algorithm GMIL-1. Then in Section IV we present our kernel-based reformulation of GMIL-1. We show that computing this kernel is equivalent to counting the number of boxes that contain at least one point from both sets P and Q, a problem that we formally define in Section V as #BOXAnd. We then give $k_{\wedge/\vee}$, our normalized version of $k_{\wedge}$, in Section VI. In Section VII we present $k_{\min}$. In all these sections, we show that computing the kernels is #P-complete, and give FPRASs for them. In Section VIII we describe experimental results of our new kernels on the applications of content-based image retrieval, protein sequence identification, and the Musk data sets. Our experiments measure the time required to run our approximation algorithms, whether the computed Gram matrices are positive semidefinite, and the generalization performance of an SVM using our methods. Finally, we conclude in Section IX.

II. Notation and Definitions

Let $\mathcal{X}$ denote $\{0,\ldots,s\}^d$ (though our results trivially generalize to $\mathcal{X} = \prod_{i=1}^{d}\{0,\ldots,s_i\}$). Let $B_{\mathcal{X}}$ denote the set of all axis-parallel boxes[1] from $\mathcal{X}$. We uniquely identify any box $b \in B_{\mathcal{X}}$ as a pair (b_ℓ, b_u), where b_ℓ is the "lower left" corner and b_u is the "upper right" corner. A point p is in box b if and only if $b_\ell \leq p \leq b_u$, where the inequality must hold for each dimension of b_ℓ, p, and b_u. There are $s+1$ possible values for each corner, so the number of intervals in each dimension is $\binom{s+1}{2} + s + 1$ since we allow degenerate (empty) intervals. Thus $|B_{\mathcal{X}}| = \left(\binom{s+1}{2} + s + 1\right)^d = \binom{s+2}{2}^d$.

Prior work in this generalization [10], [9], [12] used $\mathcal{X} = \{0,\ldots,s\}^d$ rather than $\mathcal{X} = \mathbb{R}^d$ since enumeration of the boxes was needed. In our work, our kernels count the number of boxes from $B_{\mathcal{X}}$ that contain points from the given bags. Thus we also use $\mathcal{X} = \{0,\ldots,s\}^d$ to ensure that $|B_{\mathcal{X}}|$ is finite. This does not pose a problem since in the data from our (and in prior) experimental work, it is straightforward to discretize and bound the instance space as follows. For each dimension $i \in \{1,\ldots,d\}$, scale up each real value by a fixed power of 10 to a desired precision[2] and then round them to integers. Finally, translate each axis to $\{0,\ldots,s\}$. (One should also represent the values $-\infty$ and $+\infty$ to accommodate points that lie outside the bounding box of the training data.)

For multisets (bags) $P, Q \subseteq \mathcal{X}$, let $B(P)$ denote the set of boxes in $B_{\mathcal{X}}$ that contain a point from P and $B(P \wedge Q)$ denote the set of boxes in $B_{\mathcal{X}}$ that contain a point from P and a point from Q. When P and Q contain single points then we will omit set notation. For example, $B(\{p\} \wedge \{q\})$ will be denoted $B(p \wedge q)$.

We will use vector notation to refer to points in $\mathcal{X}$ only when it is necessary (e.g. in Section V-A); otherwise we will just use lower case letters to refer to points in $\mathcal{X}$. The notion of approximation that we use is defined as follows.

Definition 1: Let f be a counting problem. Then a randomized algorithm $\mathcal{A}$ is an *FPRAS (Fully Polynomial Randomized Approximation Scheme)* if for any instance x and parameters $\epsilon, \delta > 0$,

$$\Pr\left[|\mathcal{A}(x) - f(x)| \leq \epsilon f(x)\right] \geq 1 - \delta$$

[1]This includes degenerate boxes, i.e. those with size 0 in one or more dimensions.

[2]In our experiments, we did not rescale the CBIR or Musk data; we scaled the Protein data by a factor of 10.

and A's running time is polynomial in $|x|$, $1/\epsilon$, and $1/\delta$. Further, we call $\mathcal{A}(x)$ an *ϵ-good approximation* of $f(x)$.

We make frequent reference to the complexity class #P and to #P-completeness.

Definition 2: We say a function $f : \{0,1\}^* \rightarrow \mathbb{N}$ is in the counting complexity class #P if there is a nondeterministic polynomial-time Turing machine M so that for any $x \in \{0,1\}^*$, $f(x)$ = the number of accepting computations of M on input x. We say f is #P-complete if (a) $f \in \#P$ and (b) for all $g \in \#P$ there is a deterministic polynomial-time oracle Turing machine that computes g using oracle queries to f.

From the definition, it follows that if a function is #P-complete, then it is also NP-hard. Thus it is unlikely that #P-complete problems have efficient algorithms. Typically, the counting version of an NP-complete problem is also #P-complete (for example, #SAT, which counts the number of satisfying assignments of a Boolean formula, is a standard #P-complete problem). But there are #P-complete problems whose decision version is efficiently solvable. In particular, we will use one such problem, #MDNF (given a monotone DNF formula, output the number of satisfying assignments), in our reductions. For more detail on #P-completeness and related topics, see Papadimitriou [17] or Du and Ko [18]. Finally, for simplicity in our time complexity analyses, we assume arithmetic operations take constant time.

III. Multiple-Instance Learning

In the original MIL model [3], each example P is a *bag* (multiset) of instances, and P is given a label of positive if and only if at least one of the instances in P is labeled positive (it is unknown which instance(s) in P are labeled positive). Typically, the label of a point $p \in P$ is determined by its proximity to a target point c. Since its introduction, the MIL model has been extensively studied [16], [19], [5], [4], [21], [22], [20], [23], [24], [16], [6], [7], [8], [25], [26], [27], [28], [29], along with extensions for real-valued labels [30], [31]. Primary applications include molecular binding affinity (related to drug discovery), content-based image retrieval, and text classification. In general, problems in the original MIL model have been approached in two distinct ways: (1) with the goal of inferring a classifier that can label individual instances within a bag; and (2) with the goal of inferring a classifier that operates only on entire bags. In the original MIL model, a solution for (1) implies a solution for (2). However, as we describe below, this may not be the case for more general MIL models, where a bag's label is not a

simple disjunction of instance labels, but instead a function over the instance labels that itself must be learned. Our work falls in category (2).

In most MIL work, it is assumed that a bag is labeled positive if and only if at least one of its instances is labeled positive by the target function, where the target function is typically assumed to be a single point or a single axis-parallel box. Exceptions include some work of Maron et al. [20], [4], and Ray and Craven [32], in which a target concept can be a disjunction over multiple points. Also (in a subset of their experiments), Maron et al. [20], [4] mapped each pair of instances to a new instance and added spatial information about the instance pair, which defined a pairwise-conjunctive type of learning model. However, they found that allowing more than 2 disjuncts in the target concept or taking more than 2 instances at a time proved computationally very difficult.

In other work, De Raedt [33] generalized MIL in the context of inductive logic programming and defined an interesting framework connecting many forms of learning. One of his generalizations allowed relations between instances. However, the transformations given by De Raedt between the models had exponential time and space complexity.

Scott et al. [9] generalized the MIL model such that rather than P's label being a disjunction of the labels of the instances in P, the label is represented by a threshold function. In contrast to the conventional MIL model, in their model the target concept is defined by two *sets* of points. Specifically, they defined their concepts by a set of k "attraction" points $C = \{c_1, \ldots, c_k\}$ and a set of k' "repulsion" points $\bar{C} = \{\bar{c}_1, \ldots, \bar{c}_{k'}\}$. Then the label for a bag $P = \{p_1, \ldots, p_n\}$ is positive if and only if there is a subset of r points $C' \subseteq C \cup \bar{C}$ such that each attraction point $c_i \in C'$ is near some point in P (where "near" is defined as within a certain distance under some weighted norm) and each repulsion point $\bar{c}_j \in C'$ is not near any point in P. Here r is a threshold indicating the minimum number of target points from $C \cup \bar{C}$ that must each be "hit" by some point from P (if from C) or "missed" by all points from P (if from $\bar{C}$).

In other words, if one defines a boolean attribute a_i for each attraction point $c_i \in C$ that is 1 if there exists a point $p \in P$ near it and 0 otherwise and another boolean attribute $\bar{a}_i$ for each repulsion point $\bar{c}_j \in \bar{C}$ that is 1 if there is no point from P near it, then P's label is an r-of-$(k + k')$ threshold function over the attributes (so there are $k + k'$ relevant attributes and P's label is 1 iff at least r of these attributes are 1).

The boosting-based algorithm of Auer and Ortner [28] builds an ensemble of weak hypotheses,

each of which is an axis-parallel box or a ball. In this way, their algorithm works in a generalized MIL model similar to that of Scott et al., in that hypotheses are linear combinations of features that indicate whether a box (or ball) contains a point from a bag. The main difference is that in Scott et al.'s (and our) algorithms, all possible boxes are considered whereas with Auer and Ortner, only a relatively small subset of boxes is used. Because of this, our algorithms tend to search for repulsion points (regions where no positive bags can have points) as well as attraction points, whereas Auer and Ortner's algorithm does not search for repulsion points.

Recently, Chen et al. [34] approached MIL by representing bags in a new feature space derived by the instances of the training bags. Each bag i was represented as N-dimensional vector (N is the number of instances from all bags), where the jth dimension of the vector is a measure of similarity between bag i and the jth instance of the training bags (assuming all instances are represented in an ordered list, independent of their bags). They then applied a 1-norm SVM method to select the relevant features. This implicitly captures a generalization of MIL.

Independently of Scott et al., Weidmann et al. [15] defined their own generalizations of the MIL model. The first (*presence-based* MIL) is the same as Scott et al.'s model with $r = k$ and no repulsion points. Their second (*threshold-based* MIL) generalizes presence-based MIL by requiring each $c_i \in C$ to be near at least t_i distinct points from P for P to be labeled positive, where t_i is a nonnegative integer that is part of the definition of the target concept. Their third model (*count-based* MIL) generalizes threshold-based by requiring the number of distinct points from P that are near c_i to be at least t_i and at most z_i. Count-based MIL can represent the idea of repulsion points by setting $z_i = 0$ for each repulsion point. Thus this model generalizes the one of Scott et al. when $r = k + k'$. However, the ability of Scott et al.'s model to represent r-of-$(k + k')$ threshold concepts for $r < k + k'$ expands its representational ability beyond the scope of the generalizations of Weidmann et al.

When they introduced their generalized MIL model, Scott et al. also gave an algorithm (GMIL-1) for it. GMIL-1 is adapted from an algorithm by Goldman et al. [10] (which itself is built on the "virtual threshold gates" technique of Maass and Warmuth [35]), and Scott et al. applied it to various application areas. In all tests, GMIL-1 was competitive with (and often superior to) the MIL algorithms Diverse Density [20] and EMDD [24]. GMIL-1's advantage was most clear when there was no way to represent a target concept in the original MIL model.

GMIL-1 can be summarized as follows. It operates in a d-dimensional, discretized instance

space[3] $\mathcal{X}$. GMIL-1 enumerates the set $B_{\mathcal{X}}$ of all possible boxes in $\mathcal{X}$ and creates an attribute a_b for each box $b \in B_{\mathcal{X}}$. Given a bag $P \in \mathcal{X}^n$, the algorithm sets $a_b = 1$ if some point from P lies in b and $a_b = 0$ otherwise. To capture the notion of repulsion points, they also defined complementary attributes $\bar{a}_b = 1 - a_b$. These $N = 2|B_{\mathcal{X}}|$ attributes are given to the algorithm Winnow [36], which learns a linear threshold unit. Winnow maintains a weight vector $\vec{w} \in \mathbb{R}_+^N$ (N-dimensional positive real space), initialized to all 1s. Upon receiving input $\vec{x}_i \in [0,1]^N$, Winnow makes its prediction $\hat{y}_i = +1$ if $\vec{w} \cdot \vec{x}_i \geq \theta$ and 0 otherwise ($\theta > 0$ is a threshold). Given the true label y_i, the weights are updated as follows: $\vec{w} = \vec{w}\,\alpha^{\vec{x}_i(y_i - \hat{y}_i)}$ for some $\alpha > 1$. Pseudocode[4] for GMIL-1 is in Table I.

Unfortunately, the time complexity of GMIL-1 is linear in N, which is exponential in d. Later, Tao and Scott [12] developed heuristics to build a smaller number of groups in Line 4 of Table I while compromising little accuracy. This new Winnow-based algorithm (GMIL-2) requires significantly less time and memory than GMIL-1 in practice with small (if any) increases in prediction error. However, it still has time complexity exponential in d, so it cannot be applied to e.g. Musk data, which has $d > 160$.

IV. Kernel-Based Reformulation of GMIL-1

The time complexities of GMIL-1 and GMIL-2 depend on $|\mathcal{G}|$, the number of groups that $B_{\mathcal{X}}$ is partitioned into. This is influenced by the number of clusters created in Line 3 of Table I. The larger the number of representative points added to S', the higher resolution that the algorithm has to differentiate between points in S. However, the number of groups grows with $|S'|^d$. Thus the only way to scale the algorithm to the high-dimensional Musk data is to make $|S'|$ so small that there are not enough groups to differentiate well between points in the input space, making learning impossible.

An alternative to GMIL-1's exponentially large feature mapping is to use a support vector machine [38], [39], [40] with a kernel that implicitly performs the mapping. We will show that computing such a kernel on two bags P and Q corresponds to counting the number of boxes that contain at least one point from each of P and Q. After we show that this problem is #P-complete, we develop an FPRAS for it.

[3]The discretization is performed as described in Section II.

[4]In the table we omit a post-processing heuristic since it is not relevant to our work.

TABLE I

THE ALGORITHM GMIL-1 FROM SCOTT ET AL. [9]. GOLDMAN ET AL. [10] DISCUSSES HOW TO PARTITION THE SET OF BOXES IN LINE 4, WHICH IS DONE TO SPEED UP THE ALGORITHM. $B_{\mathcal{X}}$ = THE SET OF ALL AXIS-PARALLEL BOXES IN $\mathcal{X}$.

1: Given training bags $P_1, \ldots, P_m$, threshold θ, and learning rate α
2: $S \leftarrow \bigcup_i P_i$
3: Cluster points in S, storing point representatives of clusters in S'
4: Partition $B_{\mathcal{X}}$ into groups $G_1, \ldots, G_\ell$ s.t. $\forall i$, each box $b \in G_i$ contains the same subset of points from S'
5: $\mathcal{G} \leftarrow \{G_1, \ldots, G_\ell\}$
6: $\forall\, i = 1, \ldots, \ell$, $b_i \leftarrow$ largest box in G_i
7: $\forall\, i = 1, \ldots, \ell$, initialize $w_i = 1$ and $\bar{w}_i = 1$
8: Define $a_i(P) = 1$ if b_i contains a point from bag P and 0 otherwise
9: Define $\bar{a}_i(P) = 1 - a_i(P)$
10: **for all** training rounds **do**
11: **for all** bags P_j **do**
12: **if** $\sum_i |G_i| w_i a_i(P_j) + \sum_i |G_i| \bar{w}_i \bar{a}_i(P_j) \geq \theta$ **then**
13: predict $\hat{y}_j = 1$
14: **else**
15: predict $\hat{y}_j = 0$
16: **end if**
17: **for all** $i = 1, \ldots, \ell$ **do**
18: $w_i \leftarrow w_i \alpha^{(y_j - \hat{y}_j) a_i(P_j)}$
19: $\bar{w}_i \leftarrow \bar{w}_i \alpha^{(y_j - \hat{y}_j) \bar{a}_i(P_j)}$
20: **end for**
21: **end for**
22: **end for**

First, recall the mapping of GMIL-1. GMIL-1 enumerates the set $B_{\mathcal{X}}$ of all possible boxes in $\mathcal{X}$ and creates an attribute a_b for each box $b \in B_{\mathcal{X}}$. Given a bag $P \in \mathcal{X}^n$, the algorithm sets $a_b = 1$ if some point from P lies in b and $a_b = 0$ otherwise. To capture the notion of repulsion points, they also defined complementary attributes[5] $\bar{a}_b = 1 - a_b$. This leads to the following

[5]This was done because Winnow in its standard form cannot represent negative weights. In our kernel formulation, we only use the N attributes a_b since SVMs can represent negative weights.

observation.

Observation 1: Consider two bags $P, Q \subseteq \mathcal{X}$ and a mapping $\vec{\phi}_\wedge(P) = (a_1, \ldots, a_N)$ where $a_i = 1$ if the corresponding box $b_i \in B_\mathcal{X}$ contains a point from P and 0 otherwise. Then when using an SVM rather than Winnow for learning, the remapping used by GMIL-1 corresponds to using the kernel

$$k_\wedge(P, Q) = \vec{\phi}_\wedge(P) \cdot \vec{\phi}_\wedge(Q) = |B(P \wedge Q)| \ ,$$

where $B(P \wedge Q)$ is the set of boxes that contain a point from P and contain a point from Q.

Proof: Since $\vec{\phi}_\wedge(P)$ and $\vec{\phi}_\wedge(Q)$ are binary vectors, their dot product is simply the number of 1s in corresponding positions. Since a bit from $\vec{\phi}_\wedge(P)$ is 1 if and only if the corresponding box contains a point from P, the value of $k_\wedge(P, Q)$ is obviously $|B(P \wedge Q)|$. Finally, $k_\wedge(P, Q)$ is a kernel by definition. ■

Of course, switching from Winnow to an SVM changes the regularizer used in learning: Winnow's multiplicative weight updates imply a relative entropy regularizer, whereas a support vector machine uses one based on the square of the 2-norm. Such a change in regularizer can have significant impact on the upper bounds on generalization error, especially when the target weight vector is sparse [41], [42], [43], [44], which is likely the case with our target weight vector (i.e. we expect only a handful of the set of possible boxes to be relevant). However, with the exception of Takimoto and Warmuth [45], it is not known how to efficiently run kernel-based algorithms with multiplicative weight updates. Further, in our case as with many others, the gain in efficiency far outweighs the change in error bound. As it turns out, our experimental results show an improvement in performance of the SVM using our kernels over our the Winnow-based algorithms, despite the change in error bound.

V. The Box Counting Problem #BOXAnd

From Observation 1, we now see that by switching from multiplicative weight updates (Winnow) to additive updates (SVM), one can efficiently scale GMIL-1 and GMIL-2 to handle high-dimensional data if we can efficiently compute the kernel $k_\wedge$. This kernel corresponds to the box counting problem that we call #BOXAnd, which we now define. The input to the problem is a triple $\langle \mathcal{X}, P, Q \rangle$. The problem #BOXAnd is to compute $|B(P \wedge Q)|$: the number of boxes in $B_\mathcal{X}$ that contain at least one point from each of P and Q. In this section we prove that #BOXAnd is #P-complete, and then we present a FPRAS for it.

A. *Hardness Result for* #BOXAnd

We prove that the counting problem #BoxAND is #P-complete.

Theorem 2: #BOXAnd is in #P.

Proof: We design a nondeterministic polynomial-time Turing machine, which on input $(\mathcal{X}, P, Q)$ (an instance of #BoxAND) has the number of accepting computations equal to the number of boxes in $B_{\mathcal{X}}$ that contain a point from P and a point from Q. Consider the nondeterministic machine M that on input $(\mathcal{X}, P, Q)$ first guesses a box $b \in B_{\mathcal{X}}$ and then accepts if and only if there is a point in P that is also in b, and a point in Q that is also in b. The machine M takes only linear time and the number of accepting paths of M is equal to the number of boxes that contain a point from P and a point from Q, which is $|B(P \wedge Q)|$. ■

Theorem 3: #BOXAnd is #P-complete.

Proof: We just established that #BOXAnd is in #P. We prove that #BoxAND #P-complete by reducing from the *monotone DNF counting problem* (#MDNF), shown to be #P-complete by Valiant [46]. An instance of #MDNF is a monotone boolean formula F (i.e. with no negated literals) in disjunctive normal form, and an algorithm for this problem is to output the number of satisfying assignments of F.

We need the following notation. Let F be a monotone DNF formula in n variables with m monotone terms $t_1, t_2, \ldots, t_m$. Let $S(F)$ denote the set of all satisfying assignments of F. Then $S(F) = \bigcup_i S(t_i)$. Each monotone term t can be identified with an n-bit binary vector $\vec{v_t}$ as follows: $\vec{v_t} = v_1 v_2 \ldots v_n$ where $v_i = 1$ if $x_i \in t$ and $v_i = 0$ if $x_i \notin t$. Then, since t is monotone, the set of satisfying assignments for t, $S(t) = \{\vec{a} \mid \vec{a} \geq \vec{v_t}\}$. (For two n-bit vectors $\vec{u} = (u_1, u_2, \ldots, u_n)$ and $\vec{v} = (v_1, v_2, \ldots, v_n)$, $\vec{u} \geq \vec{v}$ iff $u_i \geq v_i$ for all $1 \leq i \leq n$.)

We will reduce #MDNF to a special case of #BOXAnd where $\mathcal{X} = H_n = \{0,1\}^n$. The reduction f takes a formula $F = \bigvee_{1 \leq i \leq m} t_i$ and maps it to an instance $f(F) = \langle H_n, P, Q \rangle$ where $P = \{\vec{0}\}$ and $Q = \{\vec{v}_{t_1}, \vec{v}_{t_2}, \ldots, \vec{v}_{t_m}\}$.

We now argue that $|S(F)|$ equals the number of solutions to $\langle H_n, P, Q \rangle$ of #BOXAnd. Clearly, $B(\vec{0} \wedge \vec{v}) = \{(\vec{0}, \vec{u}) \mid \vec{u} \geq \vec{v}\}$. For any term t_i, $\vec{a} \in S(t_i) \Leftrightarrow \vec{a} \geq \vec{v}_{t_i} \Leftrightarrow (\vec{0}, \vec{a}) \in B(\vec{0} \wedge \vec{v}_{t_i})$. Thus the number of satisfying assignments of F = $|\bigcup_{1 \leq i \leq m} S(t_i)| = |\bigcup_{1 \leq i \leq m} B(\vec{0} \wedge \vec{v_{t_i}})| = |B(\{\vec{0}\} \wedge Q)|$ = the number of solutions to $\langle H_n, P, Q \rangle$. ■

B. An FPRAS for #BOXAnd

Our algorithm for estimating $|B(P \wedge Q)|$ is based on the general technique from Karp et al. [47] on the union of sets problem. In this problem, the goal is to take a description of m sets $B_1, \ldots, B_m$ and estimate the size of $B = \bigcup_{i=1}^{m} B_i$. Their algorithm is based on the idea of estimating the size of a set by sampling. They define two efficiently samplable sets U and G, $G \subseteq U$, with the guarantee that $|G| = |\bigcup_{i=1}^{m} B_i|$ and $|G|$ is at least a polynomial fraction of $|U|$. Once this is guaranteed, we can efficiently sample from U and compute the fraction of these samples that are in G, which gives a good estimate of $|G|$. The correctness of this approach follows from Chernoff bounds.

Now we give more details. In order to apply the technique of Karp et al., three criteria must be satisfied.

1) For all $i \in \{1, \ldots, m\}$, $|B_i|$ must be easily computed.
2) For all $i \in \{1, \ldots, m\}$, we must be able to sample uniformly elements from B_i.
3) Given any $s \in B$ and any $i \in \{1, \ldots, m\}$, we must be able to easily determine if $s \in B_i$.

If the above criteria are satisfied, Karp et al.'s algorithm proceeds as follows. First define $U = \{(s, i) \mid s \in B_i \text{ and } 1 \leq i \leq m\}$ (so $|U| = \sum_{i=1}^{m} |B_i|$). Define another set $G = \{(s, i) \mid i \text{ is the smallest index such that } s \in B_i\}$. Clearly $G \subseteq U$. Notice that by defining G in this manner we are avoiding double counting and hence $|G| = |B|$. Moreover, $|G| \geq |U|/n$. Karp et al.'s algorithm runs in trials. For each trial, first a set B_i is chosen at random with probability $|B_i|/|U|$. Then an element $s \in B_i$ is chosen uniformly at random. These two steps together uniformly sample a pair (s, i) from U. Finally, if $(s, i) \in G$, we increment a counter γ, otherwise do nothing. Our final estimate of $|B|$ is $|U|\gamma/S$, where S is the number of samples drawn. The following theorem bounds the error of this approximation.

Theorem 4: [47] If $S \geq 4(|U|/|G|)\ln(2/\delta)/\epsilon^2$, then

$$\Pr\left[(1-\epsilon)|B| \leq |U|\gamma/S \leq (1+\epsilon)|B|\right] \geq 1 - \delta \ .$$

We now apply Karp et al.'s result to #BOXAnd. Recall that for two points $p, q \in \mathcal{X}$, $B(p \wedge q)$ denotes the set of boxes that contain both p and q. Let $W = |B(P \wedge Q)|$. Then $W = |\bigcup_{p \in P, q \in Q} B(p \wedge q)|$. It is straightforward to compute $|B(p \wedge q)|$. Given points $p, q \in \mathcal{X}$, let $\ell = (\ell_1, \ldots, \ell_d)$ be the lower corner of the bounding box of p and q, i.e. $\ell_i = \min\{p_i, q_i\}$ for

all i. Similarly, define $u = (u_1, \ldots, u_d)$ as the upper corner. Then

$$|B(p \wedge q)| = \left(\prod_{1 \le i \le d} (\ell_i + 1) \right) \left(\prod_{1 \le i \le d} (s - u_i + 1) \right) \ .$$

Since we can exactly compute $|B(p \wedge q)|$ for all $(p, q) \in P \times Q$ and there are only n^2 such sets, we can easily choose a set $B(p \wedge q)$ with probability $|B(p \wedge q)| / \left(\sum_{p \in P, q \in Q} |B(p \wedge q)| \right)$. Further, since we can uniformly sample from $B(p \wedge q)$ by uniformly selecting lower and upper corners, we can uniformly sample from the set $U = \{(p, q, c) \mid p \in P, q \in Q, c \in B(p \wedge q)\}$.

Note that $|U| = \sum_{p \in P, q \in Q} |B(p \wedge q)|$. Now consider all the pairs (p, q) such that $p \in P$ and $q \in Q$. We define a total order $\prec$ on these pairs by sorting first by p's index in P, and then by q's index in Q. I.e. given points $p_i, p_{i'} \in P$ and $q_j, q_{j'} \in Q$, we define $(p_i, q_j) \prec (p_{i'}, q_{j'})$ iff $i < i'$ or $i = i'$ and $j < j'$.

Consider another set $G = \{(p, q, c) \in U \mid$ there are no pairs $(p', q') \prec (p, q)$ s.t. $c \in B(p' \wedge q')\}$. Then $|G| = |\bigcup_{p \in P, q \in Q} B(p \wedge q)| = W$. We check whether $(p, q, c) \in G$ in $O(dn^2)$ time by checking c against each set $B(p \wedge q)$ for all $p \in P$ and $q \in Q$. Finally, we note

$$|U| = \sum_{p \in P, q \in Q} |B(p \wedge q)| \le n^2 \max_{p,q} |B(p \wedge q)| \le n^2 |G| \ . \tag{1}$$

Thus by drawing a sufficient number of samples (p, q, c) uniformly from U and incrementing γ when $(p, q, c) \in G$, we know that $\hat{W} = |U|\gamma/S$ is an ϵ-good approximation of W, as stated in the following theorem. Since the number of samples S, the time to draw each sample, and the time to check each sample for membership in G are all polynomial in n, d, $1/\epsilon$, and $1/\delta$, our algorithm for #BOXAnd is a FPRAS.

Theorem 5: If $S \ge 4n^2 \ln(2/\delta)/\epsilon^2$, then

$$\Pr\left[(1 - \epsilon)W \le \hat{W} = |U|\gamma/S \le (1 + \epsilon)W \right] \ge 1 - \delta \ .$$

Proof: Directly from application of Equation (1) to Theorem 4. ■

Our algorithm as presented has running time $O(n^4 d \ln(1/\delta)/\epsilon^2)$ since it takes $O(dn^2)$ steps to check each sample for membership in G. However, it is possible to check for membership in G in time $O(dn)$. Given a triple (p_i, q_j, c) sampled from U, first check all points $p_{i'} \in P$ that are contained in c. If $i' < i$ for some $p_{i'} \in c$, then $(p_{i'}, q_j) \prec (p_i, q_j)$ and $(p_i, q_j, c) \notin G$. If there does not exist such a $p_{i'}$, then check all points $q_{j'} \in Q$ that are contained in c. Again, if $j' < j$ for some $q_{j'} \in c$, then $(p_i, q_{j'}) \prec (p_i, q_j)$ and $(p_i, q_j, c) \notin G$. If no such $q_{j'}$ exists, then $(p_i, q_j, c) \in G$. This check requires time $O(dn)$, reducing the total running time to $O(n^3 d \ln(1/\delta)/\epsilon^2)$.

To further reduce time complexity, we adapt Karp et al.'s "self-adjusting coverage algorithm" as shown in Table II, which is a more efficient algorithm for the union of sets problem. The following theorem bounds the error of this algorithm.

Theorem 6: [47] If $S \geq 8(1+\epsilon)m\ln(2/\delta)/\epsilon^2$, then

$$\Pr\left[(1-\epsilon)\,|B| \leq \hat{Y} \leq (1+\epsilon)\,|B|\right] \geq 1-\delta \ .$$

TABLE II

SELF-ADJUSTING COVERAGE ALGORITHM FOR UNION OF SETS PROBLEM [47].

1: Given S
2: $gtime \leftarrow 0$
3: $N_S \leftarrow 0$
4: $U \leftarrow \{(s,i) \mid s \in B_i \text{ and } 1 \leq i \leq m\}$
5: **loop**
6: randomly choose $(s,i) \in U$ with probability $1/|U|$
7: **repeat**
8: $gtime \leftarrow gtime + 1$
9: **if** $gtime > S$ **then**
10: **go to FINISH**
11: **end if**
12: randomly choose $j \in \{1, \cdots, m\}$ with probability $1/m$
13: **until** $s \in B_j$
14: $N_S \leftarrow N_S + 1$
15: **end loop**
16: **FINISH:** $\hat{Y} \leftarrow S \cdot |U|/(m \cdot N_S)$

After making changes in Table II, we can get a self-adjusting coverage algorithm for #BOXAnd as shown in Table III. Since m, the number of all possible sets $B(p \wedge q)$, is at most n^2, we get the following theorem directly from Theorem 6.

Theorem 7: If $S \geq 8(1+\epsilon)n^2\ln(2/\delta)/\epsilon^2$, then

$$\Pr\left[(1-\epsilon)|B| \leq \hat{Y} \leq (1+\epsilon)|B|\right] \geq 1-\delta \ .$$

Our self-adjusting coverage algorithm only needs to check a box for membership in $B(p \wedge q)$ instead of G like in the previous algorithm (see Step 13 in Table III), which can be done in time $O(d)$. Thus we get a FPRAS for #BOXAnd with running time $O(n^2 d \ln(1/\delta)/\epsilon^2)$.

TABLE III

SELF-ADJUSTING COVERAGE ALGORITHM FOR #BOXAnd.

1: Given S and bags P and Q
2: $gtime \leftarrow 0$
3: $N_S \leftarrow 0$
4: $U \leftarrow \{(s,i) \mid s \in B_i = B(p \wedge q)$ and $1 \leq i \leq m$ and $(p,q) \in P \times Q\}$, where $m \leq n^2$
5: **loop**
6: randomly choose $(p,q,c) \in U$ with probability $1/|U|$
7: **repeat**
8: $gtime \leftarrow gtime + 1$
9: **if** $gtime > S$ **then**
10: **go to FINISH**
11: **end if**
12: randomly choose $p' \in P$ with probability $1/|P|$
13: randomly choose $q' \in Q$ with probability $1/|Q|$
14: **until** $c \in B(p' \wedge q')$
15: $N_S \leftarrow N_S + 1$
16: **end loop**
17: **FINISH:** $\hat{Y} \leftarrow S \cdot |U|/(m \cdot N_S)$

C. Discussion

According to Observation 1, $k_\wedge(P,Q)$ is a kernel since it is the dot product of two remapped vectors, but there is no guarantee that the Gram matrix computed by our approximation algorithm is positive semidefinite. However, it is reasonable to believe that if ϵ is small and the original Gram matrix has no zero eigenvalues, the approximated matrix would not adversely affect SVM optimization. This is corroborated by our experimental results, where we found that the negative eigenvalues of our Gram matrices were relatively small in number and in magnitude, and that the SVMs using our approximate kernels had very good generalization performance.

Another observation about our kernel is that its Gram matrix potentially can have large diagonal elements relative to the off-diagonal elements. For example, in our Musk experiments, the ratio of diagonal entries in the kernel matrix to the off-diagonal entries was often around 10^{50}. In practice, SVMs do not work well with diagonally dominant kernel matrices, since they look like scaled versions of the identity matrix (thus dividing each matrix value by a constant will not fix this problem). We tried a normalized variant of $k_\wedge$ that yields unit-length vectors in feature space: $k_\wedge(P,Q)/\sqrt{k_\wedge(P,P)\,k_\wedge(Q,Q)}$, but this also yielded poor results since again the kernel matrix resembled the identity matrix.

What was successful for us was applying the technique of Schölkopf et al. [48], who propose first using a nonlinear function to reduce the value of each matrix element, such as a sub-polynomial function $\varphi(x) = \text{sign}(x) \cdot |x|^\rho$ with $0 < \rho < 1$. To then get a positive definite kernel matrix, they use the empirical kernel map $\phi_n(x) = (k'(x,x_1), k'(x,x_2), \cdots, k'(x,x_n))$, where $k'(x,x_i) = \varphi(k(x,x_i))$. Finally they apply the kernel $k_{emp}(x,y) = \phi_n(x) \cdot \phi_n(y)$. In the empirical kernel, the set $\{x_1, \cdots, x_n\}$ can consist of all training and testing bags (referred to as *transduction*) or of only the training bags. We applied this method with $k_\wedge$ to address our diagonal dominance problem.

VI. A Normalized Version of $k_\wedge$

The kernel matrices of $k_\wedge$ usually have entries with very large values. For example, for the Musk data sets, each entry is larger than 10^{600}. These big entries can cause overflow and other numerical problems. Thus we now discuss ways to reduce the magnitude of $k_\wedge$ besides the self-normalizing version of $k_\wedge$ mentioned earlier. We present $k_{\wedge/\vee} = k_\wedge(P,Q)/k_\vee(P,Q)$, where $k_\vee(P,Q)$ is the number of boxes that contain a point from P or a point from Q. The intuition is that for a large value of $k_\wedge$, dividing by $k_\vee$ can reduce the impact of accidental matches due to many 1s in the remapped feature vectors caused by bags with many points.

To compute $k_\vee(P,Q)$, we first consider the more basic problem #BOX, defined as follows. An instance of the problem is a tuple $\langle \mathcal{X}, P\rangle$, where P is a set of n points from $\mathcal{X}$. An algorithm should output the number of boxes in $B_\mathcal{X}$ that contain a point from P. That is, an algorithm for #BOX on input $\langle \mathcal{X}, P\rangle$ should output $|B(P)|$.

Theorem 8: #BOX is #P-complete.

Proof: First, it is clear that as in the case of #BOXAnd (Section V-A), #BOX is in #P.

Recall that for sets P and Q, $B(P \wedge Q)$ denotes the set of boxes that contain a point from P and a point from Q, and $B(P)$ denotes the set of boxes that contain a point from P. From Theorem 3, we have that computing $|B(P \wedge Q)|$ is #P-complete. Since $B(P \wedge Q) = B(P) \cap B(Q)$, we have:

$$\begin{aligned} |B(P \wedge Q)| &= |B(P) \cap B(Q)| \\ &= |B(P)| + |B(Q)| - |B(P) \cup B(Q)| \\ &= |B(P)| + |B(Q)| - |B(P \cup Q)| \ . \end{aligned}$$

That is, #BOXAnd can be computed using three queries to #BOX. Since computing #BOXAnd is #P-complete, #BOX is #P-hard. Since #BOX is also in #P, it is #P-complete. ■

Since $k_\vee(P, Q) = |B(P \cup Q)|$, computing $k_\vee(P, Q)$ is #P-hard. We also notice that computing $k_\vee(P, Q)$ is a special case of #BOXAnd because $|B(P \cup Q)| = |B((P \cup Q) \cap (P \cup Q))|$. A direct, but less efficient, method for approximating $k_\vee(P, Q)$ is to run our algorithm from Section V-B to compute $k_\wedge(P \cup Q, P \cup Q)$ by drawing $4n^2 \ln(2/\delta)/\epsilon^2$ samples.

We now describe a more efficient way to approximate $Y = |B(P \cup Q)|$. Note that $|B(P \cup Q)| = |\bigcup_{p \in P \cup Q} B(p)|$ and the number of all possible sets $B(p)$ is at most $2n$, where $B(p)$ denotes the set of boxes that contain the point p. It is easy to verify that the three criteria for Karp et al.'s algorithm (see Section 4.4.2) are satisfied in this case: (1) for any p, $|B(p)|$ can be computed easily; (2) we can efficiently sample from $B(p)$; and (3) given $c \in B_{\mathcal{X}}$ and $p \in P$, we can efficiently check whether $c \in B(p)$. Therefore by drawing $8n \ln(2/\delta)/\epsilon^2$ samples, we will get an ϵ-good approximation $\hat{Y}$ of Y. Table IV gives pseudocode for this algorithm.

It is simple to see that if both $\hat{k}_\wedge$ and $\hat{k}_\vee$ are within a factor of ϵ of their true values, then

$$\left(\frac{1-\epsilon}{1+\epsilon}\right) k_\wedge / k_\vee \leq \hat{k}_\wedge / \hat{k}_\vee \leq \left(\frac{1+\epsilon}{1-\epsilon}\right) k_\wedge / k_\vee \ . \tag{2}$$

We have not proven that $k_{\wedge/\vee}$ is a kernel. However, our experimental results show that for all of our data sets, there is an ϵ such that our approximation of $k_{\wedge/\vee}$ consistently yields Gram matrices that are positive semidefinite. Further, we found that SVMs using $k_{\wedge/\vee}$ typically have generalization performance competitive to that of $k_\wedge$.

VII. A Count-Based Kernel for GMIL

The GMIL model used by Scott et al. and us could be further generalized along the lines of Weidmann et al. [15], as described in Section III. We now introduce a new remapping that

TABLE IV

SELF-ADJUSTING COVERAGE ALGORITHM FOR #BOX.

Given S and bags P and Q
$gtime \leftarrow 0$
$N_S \leftarrow 0$
$U \leftarrow \{(s,i) \mid s \in B_i = B(p)$ and $1 \leq i \leq m$ and $p \in P \cup Q\}$, where $m \leq n^2$
loop
  randomly choose $(p,c) \in U$ with probability $1/|U|$
  repeat
    $gtime \leftarrow gtime + 1$
    if $gtime > S$ **then**
      go to FINISH
    end if
    randomly choose $p' \in P \cup Q$ with probability $1/|P \cup Q|$
  until $c \in B(p')$
  $N_S \leftarrow N_S + 1$
end loop
FINISH: $\hat{Y} \leftarrow S \cdot |U|/(m \cdot N_S)$

generalizes Weidmann et al.'s "count-based" MIL model and a kernel $k_{\min}$ that corresponds to that mapping. We then show that, as with $k_\wedge$ and $k_{\wedge/\vee}$, $k_{\min}$ is #P-complete to compute, so we give a FPRAS for it. We found that $k_{\min}$ can generalize better than $k_\wedge$ for a learning task in content-based image retrieval, but there is little room for improvement in the other learning tasks we tested.

A. *Extending $k_\wedge$ to $k_{\min}$*

We now extend $k_\wedge$ to work in a model that generalizes count-based MIL of Weidmann et al. [15]. Recall that their count-based MIL model stipulates that a bag P is positive if and only if each concept point $c_i \in C$ is near at least t_i, and at most z_i, distinct points from P.

We define a remapping and a kernel to capture the notion of count-based MIL, but using r-of-$(k + k')$ threshold concepts. Recall the old mapping of $k_\wedge$ (Section 4.3), where $\vec{\phi}_\wedge(P)$ is a vector of $|B_\mathcal{X}|$ bits, and for each box $b \in B_\mathcal{X}$, attribute $a_b = 1$ if box b contains a point from bag

P and 0 otherwise. In our new mapping $\vec{\phi}_{\min}(P)$, each box $b \in B_{\mathcal{X}}$ has n bits associated with it, and $a_{bi} = 1$ if box b contains at least i points from P and 0 otherwise. (Thus if b contains exactly j points from P, we have $a_{bi} = 1$ for $i \leq j$ and $a_{bi} = 0$ for $i > j$.) To see how this captures count-based MIL, imagine that there is exactly one target box b, and all positive bags have at least t and at most $z - 1$ points in b. A weight vector capturing this target concept has $w_{bt} = +1$, $w_{bz} = -1$, all other weights 0, and a bias term of $-1/2$. If there are instead k such target boxes $b_1, \ldots, b_k$, then a weight vector capturing an r-of-k threshold function over such count-based attributes would have $w_{b_j t_j} = 1$ and $w_{b_j z_j} = -1$ for each $1 \leq j \leq k$. If bag P has n_j points inside box b_j, then box b_j's weights' contribution to the dot product with $\vec{\phi}_{\min}(P)$ will be 1 if $t_j \leq n_j < z_j$ and 0 otherwise. Thus setting the bias term to $r - 1/2$ will induce a positive prediction on bag P if and only if P's points successfully "hit" at least r of the k boxes, i.e. it represents multiple target boxes in an r-of-k threshold function. This strictly generalizes Weidmann et al.'s model.

Let $P_b \subseteq P$ be the set of points from P that are contained in box b. Then the dot product $\vec{\phi}_{\min}(P) \cdot \vec{\phi}_{\min}(Q)$ is equivalent to the kernel $k_{\min}(P, Q)$ that we define as:

$$k_{\min}(P, Q) \;=\; \sum_{b \in B_{\mathcal{X}}} \min(|P_b|, |Q_b|) = \sum_{b \in B(P \wedge Q)} \min(|P_b|, |Q_b|) \;. \tag{3}$$

B. A Hardness Result for $k_{\min}$

Consider the counting problem #BOXMin, which we define as follows: Given a triple $\langle \mathcal{X}, P, Q \rangle$, compute $k_{\min}(P, Q)$. We will use another related problem for showing the hardness of #BOXMin, which we now define. The problem #BOXAnd defined in Section V is, given input the triple $\langle \mathcal{X}, P, Q \rangle$, compute $k_{\wedge}(P, Q) = |B(P \wedge Q)|$. In our proof showing that #BOXAnd is #P-complete, we actually showed that a restricted version where $|P| = 1$ is #P-complete (see the proof of Theorem 3). We call this problem #RestrictedBOXAnd.

Theorem 9: #RestrictedBOXAnd is #P-complete.

Theorem 10: #BOXMin is #P-complete.

Proof: #BOXMin is in #P: Given a triple $\langle \mathcal{X}, P, Q \rangle$, a nondeterministic machine first guesses a $b \in \mathcal{X}$ and then computes $\min(|P_b|, |Q_b|)$. If the minimum is 0, it rejects. Otherwise it branches into $\min(|P_b|, |Q_b|)$ paths and accepts. It is clear that the number of accepting paths = $k_{\min}(P, Q)$.

We now show that in fact computing $k_{\min}(P, Q)$ where P contains only one point is #P-complete by reducing #RestrictedBOXAnd to the restricted version of #BOXMin. The reduction is the identity map: an instance $\langle \mathcal{X}, \{p\}, Q\rangle$ of #RestrictedBOXAnd is mapped to the instance $\langle \mathcal{X}, \{p\}, Q\rangle$ of $k_{\min}(P, Q)$. Then we get

$$\begin{aligned} k_{\min}(\{p\}, Q) &= \sum_{b \in B_{\mathcal{X}}} \min(|P_b|, |Q_b|) \\ &= \sum_{b \in B(p \wedge Q)} \min(|P_b|, |Q_b|) + \sum_{b \notin B(p \wedge Q)} \min(|P_b|, |Q_b|) \\ &= \sum_{b \in B(p \wedge Q)} 1 = |B(p \wedge Q)| = k_{\wedge}(\{p\}, Q) \ . \end{aligned}$$

The third equality is due to the following. For all $b \in B(p \wedge Q)$, $p \in b$ and $|Q \cap b| \geq 1$. Hence the minimum is exactly 1. For all $b \notin B(p \wedge Q)$, $p \notin b$ or $Q \cap b = \phi$. Hence the minimum is 0. Therefore computing $k_{\min}(\{p\}, Q)$ is the same as computing $|B(\{p\} \wedge Q)|$, which is #P-complete.

■

C. Approximating $k_{\min}$

One way to approximate $k_{\min}$ is to approximate (3) via a simple change to our algorithm for $k_{\wedge}$. When a sampled triple $(p, q, b) \in G$, we increment γ by $\min(|P_b|, |Q_b|)$ instead of by 1. Unfortunately, the best sample size bound we can get for this technique (via Lemma 11 below) is $S = n^6 \ln(2/\delta)/(2\epsilon^2)$, yielding a time complexity of $\Theta(n^7 d \log(1/\delta)/\epsilon^2)$. To obtain a more efficient way to approximate $k_{\min}$, we rewrite $k_{\min}$ as follows:

$$\begin{aligned} k_{\min}(P, Q) &= \sum_{b \in B_{\mathcal{X}}} \min(|P_b|, |Q_b|) = \sum_{b \in B(P \wedge Q)} \min(|P_b|, |Q_b|) \\ &= \sum_{b \in B(P \wedge Q)} \frac{|P_b||Q_b|}{\max(|P_b|, |Q_b|)} = \sum_{b \in B(P \wedge Q)} \sum_{p \in P_b, q \in Q_b} \frac{1}{\max(|P_b|, |Q_b|)} \\ &= \sum_{b \in B(P \wedge Q)} \sum_{p \in P, q \in Q} \frac{I(p \in P_b)\, I(q \in Q_b)}{\max(|P_b|, |Q_b|)} \\ &= \sum_{p \in P, q \in Q} \sum_{b \in B(p \wedge q)} \frac{1}{\max(|P_b|, |Q_b|)} \ , \end{aligned} \tag{4}$$

where $I(\cdot) = 1$ if its argument is true and 0 otherwise.

Now we approximate (4). We fix each (p, q) pair and approximate that term of the summation by uniformly sampling boxes from $B(p \wedge q)$ and taking the average of $1/\max(|P_b|, |Q_b|)$ for

each box b in the sample. Multiplying this average by $|B(p \wedge q)|$ gives us an approximation of that term of the sum (see Table V). To bound the sample size required to estimate these quantities, we will employ the Hoeffding bound.

Lemma 11: (Hoeffding) Let X_i be independent random variables all with mean μ such that for all i, $a \leq X_i \leq b$. Then for any $\lambda > 0$, $\Pr\left[\left|\frac{1}{S}\sum_{i=1}^{S} X_i - \mu\right| \geq \lambda\right] \leq 2e^{-2\lambda^2 S/(b-a)^2}$.
Since we are interested in ϵ-good approximations, we will use $\lambda = \epsilon\mu$.

TABLE V

APPROXIMATION ALGORITHM FOR #BOXMin.

```
1: Given S and bags P and Q
2: Ŷ ← 0
3: for all p ∈ P do
4:    for all q ∈ Q do
5:       sum ← 0
6:       for i = 0 to S do
7:          randomly choose b ∈ B(p ∧ q) with probability 1/|B(p ∧ q)|
8:          sum ← sum + 1/ max{P_b, Q_b}
9:       end for
10:      Ŷ ← Ŷ + |B(p ∧ q)| · sum/S
11:   end for
12: end for
```

Theorem 12: Let $\hat{k}_{\min}(P, Q)$ be our approximation of $k_{\min}(P, Q)$ via approximating each term of (4) individually as described above. Then after using $n^2(n-1)^2 \ln(2n^2/\delta)/(2\epsilon^2)$ total samples and $O(n^5 d \ln(n/\delta)/\epsilon^2)$ total time,

$$\Pr\left[(1-\epsilon)\, k_{\min}(P, Q) \leq \hat{k}_{\min}(P, Q) \leq (1+\epsilon)\, k_{\min}(P, Q)\right] \geq 1 - \delta \ .$$

Proof: First note that an ϵ-good approximation of each (p, q) term of the summation yields an ϵ-good approximation of $k_{\min}(P, Q)$. Thus we focus on a single (p, q) pair. Given $b \in B(p \wedge q)$, let $X(b) = 1/\max(|P_b|, |Q_b|)$. Then

$$\mu = \mathrm{E}[X] = \frac{1}{|B(p \wedge q)|} \sum_{b \in B(p \wedge q)} 1/\max(|P_b|, |Q_b|) \ .$$

Thus $X, \mu \in [1/n, 1]$. Lemma 11 says that our approximation (using a sample of size S) is not ϵ-good with probability at most

$$2e^{-2\epsilon^2\mu^2 S n^2/(n-1)^2} \leq 2e^{-2\epsilon^2 S/(n-1)^2} \ ,$$

since $\mu \geq 1/n$. Setting this to be at most δ/n^2 (so we can apply the union bound over all n^2 failure probabilities) and solving for S, we get $S \geq (n-1)^2 \ln(2n^2/\delta)/(2\epsilon^2)$ as sufficient for an ϵ-good approximation of each term. Repeat this n^2 times (once per (p, q) pair) to approximate (4). The time complexity is $O(n^5 d \ln(n/\delta)/\epsilon^2)$ since it takes time linear in n and d to compute each max. ■

As with $k_\wedge$, $k_{\min}$ is a kernel since it is the dot product of two vectors. Also as with $k_\wedge$, there is no guarantee that the kernel matrix computed by our approximation algorithm is positive semidefinite. However, we again found that empirically, the approximate kernel works well with $\epsilon = 0.1$, and yields a Gram matrix with only about 10% of its eigenvalues negative. Not surprisingly, we also found that $k_{\min}$'s kernel matrix can be diagonally dominant, so we again used Schölkopf et al.'s method to address this.

VIII. Experimental Results

To evaluate our new kernels, we tested them with SVM^{light} [49] on the following learning tasks: content-based image retrieval, biological sequence analysis, and the Musk data. Since we know of no theoretical results assessing whether approximations to kernels are positive semidefinite, we empirically evaluated the effect of varying ϵ on the resultant Gram matrices. We also measured the effect of ϵ on the time to build the matrices and on generalization error.

In our preliminary experiments, we found that $\epsilon = 0.1$ and $\delta = 0.01$ worked well without requiring excessive amounts of time to run the approximation algorithms. (Specifically, varying ϵ from 0.2 down to 0.05 changed overall, false positive, and false negative errors by at most 0.005.) We also varied C, SVM^{light}'s soft margin parameter, and got by far the best results with $C = 10^{10}$, i.e. a hard margin. Thus, unless otherwise indicated, all reported results use those parameter values. Since our kernels require the data to lie in a discretized, bounded space, we discretized the space using the training data as described in Section II.

To compare generalization performance, we experimented with other MIL algorithms [20], [24], [28]. We also report other results from the literature when appropriate. To determine the

impact of changing from a relative entropy regularizer to the squared 2-norm regularizer of an SVM, we also report results from the Winnow-based GMIL-2 algorithm [12].

A. *Effect of Varying* ϵ

For each of the data sets Protein ($m = 193$ examples), CBIR ($m = 900$), and Musk 1 ($m = 92$), we approximated the $m \times m$ Gram matrix for each of $k_\wedge$, $k_{\wedge/\vee}$, and $k_{\min}$, using values of ϵ of 0.2, 0.1, and 0.05. This was repeated 10 times per kernel-ϵ pair.[6] Each of these 10 times, we measured the time required to compute the matrix on a Macintosh with a 2.4 GHz Intel processor, then divided that time by $\binom{m}{2} + m$ to get the average time per kernel computation. These averages were averaged over the 10 matrices. We then took each of the 10 matrices, computed their eigenvalues, and counted how many were negative, averaging these numbers across all 10 matrices. This is our measure of how close the approximate matrices are to being truly positive semidefinite. (In addition, the magnitude of the largest negative eigenvalue was always at least 1000 times smaller than that of the largest positive eigenvalue, and often even smaller still.)

Results are in Table VI. Not surprisingly, for Protein and CBIR, as ϵ decreases, the percentage of negative eigenvalues decreases, though that does not happen for Musk 1. Most surprising is that, even though we do not know for certain that $k_{\wedge/\vee}$ is a kernel, there is some value of ϵ for which all 10 of its approximate Gram matrices *are* positive semidefinite[7].

Regarding time complexity, we find that for moderate values of n (CBIR and Musk 1), the kernel computation is fairly fast. (E.g. for Musk 1, the entire 92×92, $\epsilon = 0.2$, $k_\wedge$ Gram matrix was approximated in about 6 minutes, and the 900×900 approximate $k_\wedge$ matrix for CBIR was computed in about 2 hours for $\epsilon = 0.2$. Also, note that an SVM optimizer likely would not need to compute the kernel on all pairs of training instances, so these stated Gram matrix computation times are loose upper bounds on total SVM run time.) For larger values of n (Protein and Musk 2), computation can be moved off-line and easily parallelized.

[6]Exceptions: for Protein, we did not run any approximation for $\epsilon = 0.05$ and we did not run $k_{\min}$ at all. This was due to the large value of n, which was as big as 189. We also did not evaluate Musk 2 in this test, since n can be as large as 1044. However, in experiments on generalization performance, we did test $k_{\min}$ on Protein and all three kernels on Musk 2.

[7]Note that in the table, ϵ refers to the value of the parameter given to the algorithm. Thus for $k_{\wedge/\vee}$, $\epsilon = 0.2$ implies that each of $\hat{k}_\wedge$ and $\hat{k}_\vee$ is within 0.2 of its true value, which means that based on Equation (2), $2k_{\wedge/\vee}/3 \leq \hat{k}_{\wedge/\vee} \leq 3k_{\wedge/\vee}/2$.

TABLE VI

AVERAGE TIME (IN SECONDS) PER KERNEL EVALUATION AND PERCENTAGE OF EIGENVALUES OF THE $m \times m$ GRAM MATRIX THAT WERE NEGATIVE (ALL GRAM MATRICES WERE OF FULL RANK). RUNS WERE REPEATED 10 TIMES. FOR PROTEIN, $n \in [35, 189]$, $d = 8$, AND THE NUMBER OF EXAMPLES IN THE DATA SET WAS $m = 193$. FOR CBIR, $n \in [2, 15]$, $d = 5$, AND $m = 900$. FOR MUSK 1, $n \in [2, 40]$, $d = 166$, AND $m = 92$. EACH TIME FOR $k_{\wedge/\vee}$ IS THE SUM OF THE TIME TO APPROXIMATE $k_\wedge$ AND THE TIME TO APPROXIMATE $k_\vee$. IN THE TABLE, ϵ REFERS TO THE VALUE OF THE PARAMETER GIVEN TO THE ALGORITHM. THUS THE QUALITY OF $k_{\wedge/\vee}$'S APPROXIMATION IS IN FACT LESS THAN ϵ (CF. EQUATION (2)).

	ϵ for $k_\wedge$			ϵ for $k_{\wedge/\vee}$			ϵ for $k_{\min}$		
Data Set	0.2	0.1	0.05	0.2	0.1	0.05	0.2	0.1	0.05
Protein	6.24	10.96	N/A	6.41	11.44	N/A	N/A	N/A	N/A
	0.1%	0.05%	N/A	0%	0%	N/A	N/A	N/A	N/A
CBIR	0.018	0.053	0.213	0.026	0.082	0.323	0.028	0.090	0.364
	25.9%	15.8%	5.64%	11.7%	1.9%	0%	19.76%	10.29%	1.78%
Musk 1	0.086	0.258	1.032	0.172	0.516	1.672	0.803	2.58	10.32
	9.57%	9.78%	9.90%	0%	0%	0%	9.3%	9.3%	9.4%

In our experiments of Table VII, the run times for EMDD, DD, and Boost were about one minute per run for the CBIR learning task, and about an hour per run for GMIL-2. For Protein, EMDD and DD required about an hour per run (Auer and Ortner did not report run times for Boost). Thus even for $\epsilon = 0.2$, an SVM using our kernels is slower than the other algorithms we tested if only one run is considered. However, Table VII shows significant improvement in generalization performance of our methods over these others on several learning tasks, especially in total and false negative error. Further, once our kernels' Gram matrices are computed, SVM optimization takes under one minute. Thus, since learning tasks in applications such as CBIR, protein classification, and drug binding affinity can be treated as a series of database queries, one could compute an approximate Gram matrix once for the entire database and then amortize this effort over multiple queries. After a few hundred queries, the per-query effort of computing the entire matrix is comparable to the run times of the other algorithms.

B. Content-Based Image Retrieval

Maron and Ratan [4] explored the use of conventional MIL for content-based image retrieval (CBIR) for images of natural scenes. The system is *query by example*, where the user presents examples of desired images, and the system's job is to determine commonalities among them. They filtered and subsampled their images and then extracted "blobs" (groups of m adjacent pixels), which they mapped to a $(3m)$-dimensional space (one attribute per RGB pixel value). Each blob was mapped to one point in a bag, and all bags derived from query images were labeled as positive. This was extended by Zhang et al. [5], who compared the use of the algorithm Diverse Density (DD) [20] to EMDD [24] on data sets preprocessed with numerous feature extraction methods, including RGB profiling of blobs and YCrCb (luminance-chrominance) representations coupled with wavelet coefficients [50] to represent texture.

Some of Zhang et al.'s best results came from their segmentation-based YCrCb (luminance-chrominance) bag representation with wavelet coefficients. Specifically, they divided each image into 4×4 blobs of pixels, and represented each blob with six features: Y, Cr, Cb, HL(Y), LH(Y), and HH(Y), where the latter three features came from applying Daubechies-4 wavelet transforms [50] on the luminance component. They then segmented the image with a k-means segmentation algorithm [51] and for each segment, averaged the 6 features, which relates each segment to a point in the bag that corresponds to the entire image. To improve efficiency for GMIL-2, Tao and Scott removed the luminance value from each feature vector, reducing the dimensionality of the feature space to 5.

We experimented with the two CBIR tasks[8] used by Scott et al. [9]. One is the "Sunset" task: to distinguish images containing sunsets from those not containing sunsets. Like Zhang et al. [5], Scott et al. 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 CBIR task Scott et al. experimented with was to test a conjunctive CBIR concept, where the goal was to distinguish images containing a field with no sky from those containing a field and sky or containing no field. Zhang et al.'s field images that contained the sky were relabeled from positive to negative. Each training set had 6 bags of each of Flower, Mountain,

[8]Based on data from Wang et al. [52], the Corel Image Suite, and www.webshots.com.

Sunset, and Waterfall for negatives, and had 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. Each test set had 120 Fields, around 50 serving as positives and the remainder as negatives.

In addition to the above two CBIR tasks, we added the learning task "Sunset 2," where the negative examples are derived from images with no sunset or with a sunset with the sun itself visible, and the positive examples are derived from images containing a sunset, but the sun itself is hidden. We took the 30 testing/training combinations from the sunset task and relabeled from positive to negative the sunset bags for which the sun was visible. This relabeled about 20–30 positive bags to negative in each training set, and about 50–60 in each testing set.

The top three rows of each table in Table VII summarize the prediction error of $k_\wedge$, $k_{\wedge/\vee}$, $k_{\min}$, and GMIL-2 [12]. We also give results for Diverse Density [20] and EMDD [24], which operate in the conventional MIL model. The Sunset task fits well into the conventional MIL model; hence error rates for EMDD and DD are only about 1% higher than ours. But since the Conjunctive and Sunset 2 tasks are more complex, we see that the GMIL model helps significantly over conventional MIL.

Also reported are results from our runs of the boosting-based algorithm of Auer and Ortner [28]. We used axis-parallel boxes as the weak hypotheses, so their ensembles are weighted combinations of axis-parallel-boxes, just like our hypotheses. The main difference between their ensembles and our hypotheses is that we use features from all the boxes in $B_{\mathcal{X}}$, whereas theirs use a very small subset[9]. Further, our algorithm looks for both "attraction" and "repulsion" boxes, i.e. for boxes that must be hit for a bag to be positive and for ones that must be missed. In contrast, Auer and Ortner's algorithm only seeks out boxes that must be hit. Based on the large gaps between our and Boost's false negative error rates, we infer that the Conjunctive and Sunset 2 tasks require some sort of repulsion points.

C. Identifying Trx-Fold Proteins

The low conservation of primary sequence in protein superfamilies such as Thioredoxin-fold (Trx-fold) makes conventional modeling methods difficult to use. Wang et al. [53] proposed using

[9]We ran their algorithm for 2, 5, 7, 10, and 20 boosting rounds. The results for 10 rounds are slightly better than the others, so they are the ones reported in the table.

TABLE VII

CLASSIFICATION ERRORS AND STANDARD DEVIATIONS FOR CBIR AND PROTEIN LEARNING TASKS. $k_{\min}$, $k_\wedge$, AND $k_{\wedge/\vee}$ WERE ALL APPROXIMATED WITH $\epsilon = 0.1$ AND $\delta = 0.01$. "BOOST" RESULTS FOR PROTEIN ARE FROM AUER AND ORTNER [28], "GMIL-2" RESULTS ON SUNSET AND CONJUNCTIVE ARE FROM TAO AND SCOTT [12], AND "GMIL-2" RESULTS ON PROTEIN ARE FROM WANG ET AL. [53]. ALL OTHER RESULTS ARE FROM OUR OWN EXPERIMENTS.

Total Error

TASK	$k_{\min}$	$k_\wedge$	$k_{\wedge/\vee}$	**GMIL-2**	**EMDD**	**DD**	**Boost**
Sunset	0.084 ± 0.015	0.086 ± 0.016	0.086 ± 0.012	0.096 ± 0.008	0.096 ± 0.019	0.099 ± 0.020	0.112 ± 0.021
Sunset 2	0.072 ± 0.007	0.075 ± 0.009	0.074 ± 0.008	N/A	0.094 ± 0.015	0.101 ± 0.014	0.123 ± 0.019
Conj.	0.086 ± 0.017	0.106 ± 0.018	0.097 ± 0.019	0.147 ± 0.028	0.215 ± 0.028	0.181 ± 0.024	0.203 ± 0.026
Protein	0.215 ± 0.045	0.208 ± 0.046	0.173 ± 0.041	0.250	0.365 ± 0.057	0.664 ± 0.113	0.036

False Positive Error

TASK	$k_{\min}$	$k_\wedge$	$k_{\wedge/\vee}$	**GMIL-2**	**EMDD**	**DD**	**Boost**
Sunset	0.085 ± 0.014	0.088 ± 0.016	0.087 ± 0.012	0.082 ± 0.011	0.082 ± 0.017	0.078 ± 0.016	0.113 ± 0.032
Sunset 2	0.076 ± 0.008	0.079 ± 0.009	0.078 ± 0.009	N/A	0.044 ± 0.020	0.056 ± 0.019	0.089 ± 0.026
Conj.	0.086 ± 0.017	0.107 ± 0.017	0.098 ± 0.019	0.140 ± 0.031	0.213 ± 0.035	0.173 ± 0.032	0.206 ± 0.034
Protein	0.215 ± 0.045	0.208 ± 0.047	0.172 ± 0.042	0.250	0.365 ± 0.057	0.668 ± 0.113	0.036

False Negative Error

TASK	$k_{\min}$	$k_\wedge$	$k_{\wedge/\vee}$	**GMIL-2**	**EMDD**	**DD**	**Boost**
Sunset	0.078 ± 0.019	0.077 ± 0.020	0.075 ± 0.015	0.157 ± 0.018	0.166 ± 0.048	0.168 ± 0.051	0.108 ± 0.040
Sunset 2	0.030 ± 0.009	0.034 ± 0.011	0.026 ± 0.009	N/A	0.471 ± 0.217	0.595 ± 0.203	0.475 ± 0.103
Conj.	0.076 ± 0.020	0.089 ± 0.022	0.099 ± 0.020	0.244 ± 0.049	0.244 ± 0.099	0.282 ± 0.101	0.160 ± 0.098
Protein	0.144 ± 0.402	0.169 ± 0.391	0.250 ± 0.446	0.250	0.360 ± 0.513	0.125 ± 0.643	0.000

multiple-instance learning as a tool for identification of new Trx-fold proteins. They mapped each protein's primary sequence to a bag as described below.

In the QFC algorithm [54], the physico-chemical properties of the amino acids in the molecules are characterized using various indices and standard measurements, such as GES hydropathy index [55], [56], solubility [57], polarity, pI, Kyte-Doolittle index [58], α helix index [59], and molecular weight. A protein sequence is described by a set of variables x_1 through x_n, and for each x_i, there is a value x_{ij} for the ith amino acid index (property) value at the jth position of the sequence. Thus x_{i1} through x_{im} constitutes a profile of the protein in terms of the ith

amino-acid property index.

Wang et al. mapped the Trx data to the multiple-instance learning model in the following way. First, they found the primary sequence motif in each (positive and negative) sequence and extracted a window of size 214 around it (30 residues upstream, 180 downstream). They then mapped all sequences to their profiles based on the 7 properties of Kim et al. [54], yielding 7-dimensional data, which they then smoothed with a Gaussian kernel.

Since each 7-tuple $x_i = (x_{i1}, \ldots, x_{i7})$ in each profile is tied to a particular residue r_{x_i} in the original sequence, they added an eighth coordinate x_{i8} to x_i that corresponds to r_{x_i}'s position in the sequence. They first aligned the sequences based on a conserved primary sequence motif, and then they set x_{i8} to be the position of r_{x_i} in the alignment.

Wang et al. used GMIL-2 in cross-validation tests: 20-fold CV on 20 positives and 8-fold CV on 160 negatives. In each round, they trained GMIL-2 on 19 positive proteins plus one of 8 sets of negative proteins, and tested on the held-out positive protein plus the remaining 7 sets of negative proteins. We performed the same tests with $k_\wedge$, $k_{\wedge/\vee}$, and $k_{\min}$, comparing to GMIL-2, EMDD, and DD. Also reported are results from experiments by Auer and Ortner's [28] boosting-based MIL algorithm. The results suggest that our algorithms' use of all boxes from $B_\mathcal{X}$ is causing overfitting. Restricting our kernels to count boxes from a subset of $B_\mathcal{X}$ may remedy this.

An issue that arises when applying multiple-instance learning to protein identification problems is determining the eighth coordinate x_{i8} for x_i. In the Trx data set, a simplistic method (based on a small motif that is known to appear in all Trx sequences) was used to first align all the sequences, yielding a coordinate system for the eighth dimension. In general, such multiple alignments are difficult or even impossible to achieve unless the sequences are already highly similar (but if this is the case, then learning approaches such as hidden Markov models are preferred). In contrast, *pairwise* alignments are much easier to generate, suggesting an alternative approach. Given two sequences S_P and S_Q, we could first pairwise-align them to get the coordinate system for computing $k(P, Q)$, where P and Q are the bags from S_P and S_Q, and k is one of our kernels. Thus rather than using a single, universal coordinate system for the eighth dimension, we could custom-build a coordinate system for each pair of sequences. In contrast, DD, EMDD and Boost all require a single, universal coordinate system, which may limit their applicability.

D. Musk Data Sets

We tested on the Musk data sets from the UCI repository[10], which represent different conformations of various molecules, labeled according to whether they exhibit a "musk-like" odor when smelled by a human expert. We performed 10-fold cross-validation experiments on the same 10 partitions used by Dietterich et al. [3].

The ratio of diagonal Gram matrix entries to off-diagonal entries was around 10^{50}, so we applied the method of Schölkopf et al. [48] (Section V-C), using the function $x^{1/50}$ to reduce the range of each entry in the Gram matrices. We then let SVM^{light} work with the original kernel matrices as well as transduction empirical kernels and non-transduction empirical kernels. Table VIII summarizes our results and those from the literature.

While we see that the empirical kernels based on $k_\wedge$ and $k_{\min}$ provided some of the best results on Musk, there is no improvement of $k_{\min}$ over $k_\wedge$. In fact, the results exactly match except for false positive error on Musk 1 for the transduction case (not shown), in which $k_\wedge$ is better. One possible explanation for this is that since $k_{\min}(P,Q)/k_\wedge(P,Q) \in [1,n]$ for all P,Q and the kernel is so diagonally dominant for such high-dimensional input data, the kernel values are too similar to each other to make a difference in training and testing. Thus in cases like Musk when there is diagonal dominance, there is probably little reason to choose $k_{\min}$ over $k_\wedge$.

We also found that for both the transduction and non-transduction cases, $k_{\wedge/\vee}$ had better performance on Musk 2 than $k_{\min}$ and $k_\wedge$, though this was not the case on Musk 1. One possible reason is that Musk 2 data is not filtered to remove highly similar conformations like Musk 1. The sizes of bags in Musk 2, which are hundreds of points, are much larger than those in Musk 1. The normalization helps to reduce the impact of accidental matches.

IX. Conclusions

Algorithms GMIL-1 [9] and its faster variant GMIL-2 [12] in Scott et al.'s generalization of the conventional MIL model have enjoyed success in applications that cannot be represented in the conventional MIL model. However, both algorithms are inherently inefficient, preventing scaling to higher-dimensional data. We adapted their Winnow-based algorithms to instead use a support vector machine using our new kernel $k_\wedge$. We then introduced a normalized version of

[10] http://www.ics.uci.edu/~mlearn/MLRepository.html

TABLE VIII

CLASSIFICATION ERRORS AND STANDARD DEVIATIONS ON THE MUSK DATA SETS. $k_{\min}$, $k_\wedge$, AND $k_{\wedge/\vee}$ WERE ALL APPROXIMATED WITH $\epsilon = 0.1$ AND $\delta = 0.01$. EMDD, MI-SVM, AND MI-SVM ARE FROM ANDREWS ET AL. [16], DD IS FROM MARON AND LOZANO-PÉREZ [20], TLC IS FROM WEIDMANN ET AL. [15], BOOST IS FROM AUER AND ORTNER [28], MILES IS FROM CHEN ET AL. [34], IAPR IS FROM DIETTERICH ET AL. [3], AND POLYNOMIAL MINIMAX KERNEL AND MI KERNEL ARE FROM GÄRTNER ET AL. [29].

ALGORITHM	**MUSK1**	**MUSK2**
$k_{\min}$	0.176 ± 0.080	0.227 ± 0.093
$k_{\min\ emp}$ non-transduction	0.120 ± 0.059	0.118 ± 0.070
$k_{\min\ emp}$ transduction	0.098 ± 0.059	0.097 ± 0.068
$k_\wedge$	0.176 ± 0.080	0.227 ± 0.093
$k_{\wedge\ emp}$ non-transduction	0.120 ± 0.059	0.118 ± 0.070
$k_{\wedge\ emp}$ transduction	0.088 ± 0.044	0.097 ± 0.068
$k_{\wedge/\vee}$	0.153 ± 0.047	0.275 ± 0.087
$k_{\wedge/\vee\ emp}$ non-transduction	0.109 ± 0.044	0.099 ± 0.077
$k_{\wedge/\vee\ emp}$ transduction	0.098 ± 0.049	0.078 ± 0.056
TLC	0.113 ± 0.016	0.169 ± 0.032
EMDD	0.152	0.151
DD	0.120	0.160
mi-SVM	0.126	0.164
MI-SVM	0.221	0.157
Boost	0.080	0.129
IAPR	0.076 ± 0.028	0.108 ± 0.031
MILES	0.137 ± 0.007	0.123 ± 0.007
Polynomial Minimax Kernel	0.084 ± 0.07	0.137 ± 0.012
MI Kernel	0.136 ± 0.011	0.120 ± 0.010

$k_\wedge$ called $k_{\wedge/\vee}$ that showed improvement over $k_\wedge$ on certain learning tasks and generally had Gram matrices that were positive semidefinite, or at least nearly so. Finally, we introduced a third kernel $k_{\min}$ that induces a remapping that generalizes the GMIL models of Scott et al. and Weidmann et al. [15]. We showed that each of our similarity measures is hard to compute in general, and then we presented a FPRAS for each.

Empirical results show improvements in generalization error for our methods over the Winnow-based GMIL-2. Also, $k_{\wedge/\vee}$ had improved performance over $k_\wedge$ on some tasks, and had a positive

semidefinite Gram matrix for some approximation level ϵ for each application.

It is trivial to parallelize the computation of our kernel to get an almost linear speedup. Further, since each learning task in applications such as CBIR and drug binding affinity can be treated as a database query, one could build the kernel matrix once for the entire database and reuse it for each query. This would amortize the cost of building the matrix over many queries.

ACKNOWLEDGMENT

The authors thank Tom Dietterich for his Musk partitionings, Qi Zhang, Sally Goldman, and James Wang for the CBIR data (indirectly from Corel and webshots.com), Qi Zhang for his EMDD/DD code, Ronald Ortner for his boosting-based MIL code, and the anonymous reviewers for their helpful comments. This research was funded in part by NSF grants CCR-0092761, CCF-0430991, and EPS-0091900. It was also supported in part by NIH Grant Number RR-P20 RR17675. This work was completed in part utilizing the Research Computing Facility of the University of Nebraska. Qingping Tao, Thomas Osugi, and Brandon Mueller did this work at the University of Nebraska.

REFERENCES

[1] Q. Tao, S. Scott, N. V. Vinodchandran, and T. Osugi, "SVM-based generalized multiple-instance learning via approximate box counting," in *Proc. of the 21st International Conf. on Machine Learning*, 2004, pp. 799–806.

[2] Q. Tao, S. Scott, N. V. Vinodchandran, T. Osugi, and B. Mueller, "An extended kernel for generalized multiple-instance learning," in *Proc. of the 16th IEEE Int. Conf. on Tools with Artificial Intelligence*, 2004, pp. 272–277.

[3] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multiple-instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1–2, pp. 31–71, 1997.

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

[5] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts, "Content-based image retrieval using multiple-instance learning," in *Proc. 19th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2002, pp. 682–689.

[6] Y. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions," *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 913–939, 2004.

[7] Z. Zhou, M. Zhang, and K. Chen, "A novel bag generator for image database retrieval with multi-instance learning techniques," in *Proc. of the 15th IEEE Int. Conf. on Tools with Artificial Intelligence*, 2003, pp. 565–569.

[8] C. Yang and T. Lozano-Pérez, "Image database retrieval with multiple-instance learning techniques," in *Proc. of the 16th International Conf. on Data Engineering*, 2000, pp. 233–243.

[9] S. Scott, J. Zhang, and J. Brown, "On generalized multiple-instance learning," *International Journal of Computational Intelligence and Applications*, vol. 5, no. 1, pp. 21–35, March 2005.

[10] S. A. Goldman, S. K. Kwek, and S. D. Scott, "Agnostic learning of geometric patterns," *Journal of Computer and System Sciences*, vol. 6, no. 1, pp. 123–151, February 2001.

[11] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol. 2, no. 4, pp. 285–318, 1988.

[12] Q. Tao and S. Scott, "A faster algorithm for generalized multiple-instance learning," in *Proc. of the 17th International Florida Artificial Intelligence Research Society Conf. (FLAIRS)*, 2004, pp. 550–555.

[13] B. Schölkopf, *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997.

[14] T. Haasdonk, "Feature space interpretation of SVMs with indefinite kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 482–492, 2005.

[15] N. Weidmann, E. Frank, and B. Pfahringer, "A two-level learning method for generalized multi-instance problems," in *Proc. of the European Conf. on Machine Learning*, 2003, pp. 468–479.

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

[17] C. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[18] D. Du and K. Ko, *Theory of Computational Complexity*. John Wiley and Sons, 2000.

[19] P. Auer, "On learning from multi-instance examples: Empirical evaluation of a theoretical approach," in *Proc. 14th International Conf. on Machine Learning*. Morgan Kaufmann, 1997, pp. 21–29.

[20] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in Neural Information Processing Systems 10*, 1998, pp. 570–576.

[21] P. M. Long and L. Tan, "PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples," *Machine Learning*, vol. 30, pp. 7–21, 1998.

[22] A. Blum and A. Kalai, "A note on learning from multiple-instance examples," *Machine Learning*, vol. 30, pp. 23–29, 1998.

[23] J. Wang and J.-D. Zucker, "Solving the multiple-instance problem: A lazy learning approach," in *Proc. of the Seventeenth International Conf. on Machine Learning*, 2000, pp. 1119–1125.

[24] Q. Zhang and S. A. Goldman, "EM-DD: An improved multiple-instance learning technique," in *Neural Information Processing Systems 14*, 2001, pp. 1073–1080.

[25] J. Ramon and L. de Raedt, "Multi instance neural networks," in *Proc. of the ICML-2000 Workshop on Attribute-Value and Relational Learning*, 2000.

[26] P. Auer, P. M. Long, and A. Srinivasan, "Approximating hyper-rectangles: Learning and pseudo-random sets," in *Proc. of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. ACM, 1997, pp. 314–323.

[27] H. Blockeel, D. Page, and A. Srinivasan, "Multi-instance tree learning," in *Proc. of the 22nd International Conf. on Machine Learning*, 2005, pp. 57–64.

[28] P. Auer and R. Ortner, "A boosting approach to multiple instance learning," in *Proc. of the 15th European Conf. on Machine Learning*, 2004, pp. 63–74.

[29] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proc. of the Nineteenth International Conf. on Machine Learning*, 2002, pp. 179–186.

[30] D. R. Dooly, Q. Zhang, S. A. Goldman, and R. A. Amar, "Multiple-instance learning of real-valued data," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 651–678, 2002.

[31] S. Ray and D. Page, "Multiple instance regression," in *Proc. of the Eighteenth International Conf. on Machine Learning*, 2001, pp. 425–432.

[32] S. Ray and M. Craven, "Supervised versus multiple-instance learning: An empirical comparison," in *Proc. of the 22nd International Conf. on Machine Learning*, 2005, pp. 697–704.

[33] L. De Raedt, "Attribute-value learning versus inductive logic programming: The missing links," in *Proc. 8th International Conf. on Inductive Logic Programming*. Springer Verlag, 1998, pp. 1–8.

[34] Y. Chen, J. Bi, and J. Z. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1931–1947, December 2006.

[35] W. Maass and M. K. Warmuth, "Efficient learning with virtual threshold gates," *Information and Computation*, vol. 141, no. 1, pp. 66–83, 1998.

[36] N. Littlestone, "Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow," in *Proc. of the Fourth Ann. Workshop on Comp. Learning Theory*. Morgan Kaufmann, 1991, pp. 147–156.

[37] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psych. Rev.*, vol. 65, pp. 386–407, 1958, (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[38] V. Vapnik, *Statistical Learning Theory*, ser. Adaptive and Learning Systems for Signal Processing, Communications, and Control. New York: John Wiley and Sons, 1998.

[39] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[40] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[41] M. Warmuth and S. V. N. Vishwanathan, "Leaving the span," in *Proc. of the 18th Ann. Conf. Learning Theory*, 2005.

[42] R. Khardon, D. Roth, and R. Servedio, "Efficiency versus convergence of boolean kernels for online learning algorithms," *Journal of Artificial Intelligence Research*, vol. 24, no. Sep, pp. 341–356, 2005.

[43] R. Khardon and R. Servedio, "Maximum margin algorithms with boolean kernels," *Journal of Machine Learning Research*, vol. 6, pp. 1405–1429, 2005.

[44] T. Zhang, "Regularized Winnow methods," in *Advances in Neural Information Processing Systems*, 2000, pp. 703–709.

[45] E. Takimoto and M. K. Warmuth, "Path kernels and multiplicative updates," *Journal of Machine Learning Research*, vol. 4, pp. 773–818, 2003.

[46] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM J. of Comp.*, vol. 8, pp. 410–421, 1979.

[47] R. Karp, M. Luby, and N. Madras, "Monte-Carlo approximation algorithms for enumeration problems," *Journal of Algorithms*, vol. 10, pp. 429–448, 1989.

[48] B. Schölkopf, J. Weston, E. Eskin, C. Leslie, and W. S. Noble, "A kernel approach for learning from almost orthogonal patterns," in *Proc. of the 13th European Conf. on Machine Learning*, 2002, pp. 511–528.

[49] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1999, ch. 11, pp. 169–184.

[50] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Comm. Pure and Appl. Math.*, vol. 41, pp. 909–996, 1988.

[51] J. A. Hartigan and M. A. Wong, "Algorithm AS136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[52] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001.

[53] C. Wang, S. Scott, J. Zhang, Q. Tao, D. E. Fomenko, and V. N. Gladyshev, "A study in modeling low-conservation protein superfamilies," Department of Computer Science, University of Nebraska, Tech. Rep. TR-UNL-CSE-2004-3, 2004.

[54] J. Kim, E. N. Moriyama, C. G. Warr, P. J. Clyne, and J. R. Carlson, "Identification of novel multi-transmembrane proteins from genomic databases using quasi-periodic structural properties," *Bioinformatics*, vol. 16, no. 9, pp. 767–775, 2000.

[55] D. M. Engelman, T. A. Steitz, and A. Goldman, "Identifying non-polar transbilayer helices in amino acid sequences of membrane proteins," *Annual Review of Biophysics and Biophysical Chemistry*, vol. 15, pp. 321–353, 1986.

[56] G. V. Heijne, "Membrane protein structure prediction: Hydrophobicity analysis and the positive-inside rule," *Journal of Molecular Biology*, vol. 225, pp. 487–494, 1992.

[57] T. Brown, *Molecular Biology Labfax*, 2nd ed. Academic Press, 1998.

[58] J. Kyte and R. F. Doolittle, "A simple method for displaying the hydropathic character of a protein," *Journal of Molecular Biology*, vol. 157, pp. 105–132, 1982.

[59] G. Deleage and B. Roux, "An algorithm for protein secondary structure prediction based on class prediction," *Protein Engineering*, vol. 1, pp. 289–294, 1987.