Speeding Up and Boosting Diverse Density Learning

James R. Foulds and Eibe Frank

Department of Computer Science, University of Waikato, New Zealand
{jf47,eibe}@cs.waikato.ac.nz

Abstract. In multi-instance learning, each example is described by a *bag* of instances instead of a single feature vector. In this paper, we revisit the idea of performing multi-instance classification based on a point-and-scaling concept by searching for the point in instance space with the highest diverse density. This is a computationally expensive process, and we describe several heuristics designed to improve runtime. Our results show that simple variants of existing algorithms can be used to find diverse density maxima more efficiently. We also show how significant increases in accuracy can be obtained by applying a boosting algorithm with a modified version of the diverse density algorithm as the weak learner.

1 Introduction

Multi-instance (MI) learning [7] is a variation of traditional supervised learning with applications in areas such as drug activity prediction [7], content-based image retrieval [26], stock market prediction [13] and text categorization [1].

In standard supervised learning, each example is represented by a single feature vector. In MI learning, examples are collections of feature vectors, called *bags*. The feature vectors within the bags are known as *instances*. Each instance is a vector of (typically real-valued) attribute values. Each bag has a class label, but labels are not given for the instances. The task is to learn a model from a set of training bags to predict the class labels of unseen future bags.

The original motivating application for MI learning was the *musk* drug activity prediction problem [7]. The task is to predict whether a given molecule will bind to a target "binding site" on another molecule, and hence emit a "musky" odor. A molecule may take on several different conformations (shapes) by rotating its internal bonds. If a single conformation can bind to the target binding site, the molecule is considered to be active. This is a difficult learning problem because it is not always clear which conformation is responsible for activity. Dietterich *et al.* represented each molecule as a bag containing the different conformations that the molecule can adopt.

Much of the work on MI learning, including all early work and notably including [7] and [14], makes a specific assumption regarding the relationship between the instances within a bag and its class label. We will follow [21], and refer to this assumption as the *standard MI assumption*. It states that each instance has a hidden class label $c \in \Omega = \{+, -\}$. Here, '$+$' is the *positive* class, and '$-$' is the *negative* class. The set of class labels for bags is also Ω. Under this assumption, a bag is positive if and only if at least one of its instances is positive. (i.e. belong to the '$+$' class). Thus, the bag-level class label is determined by the disjunction of the instance-level class labels.

A number of algorithms for MI learning can be found in the literature. Dietterich *et al.* [7] presented the first algorithms for MI learning, which use axis-parallel hyper-rectangles (APR) to solve the *musk* problem. The algorithms build a single APR that identifies the "positive" region of instance space. At classification time, any bag that contains an instance within the APR is labeled as positive, as per the standard MI assumption.

A different method for tackling MI learning problems is to transform the data so that unmodified single-instance learner can be applied, e.g. by computing summary statistics as in the relational learning system RELAGGS [12], or by labelling each instance with its bag's label and combining predictions [16].

A common approach to MI learning is to "upgrade" standard supervised algorithms to handle the MI scenario by modifying their internals. Such algorithms include k-nearest neighbours [20], support vector machines [1, 11], decision trees and rules [6], logistic regression [24] and boosting [2, 24, 19]. In [3], a different approach is used: a *standard* boosting algorithm is applied in conjunction with an MI base learner—a special-purpose one that induces hyper-balls or hyper-rectangles. We pursue the same basic approach in this paper, and show that boosting can be applied successfully in conjunction with a modified version of an existing, established MI algorithm, the diverse density method proposed by Maron [14].

The diverse density method is a statistical approach to MI learning under the standard assumption. However, the original learning techniques within the framework are computationally expensive. In this paper we first investigate heuristics that are designed to improve computational efficiency. We show that runtime can be improved without loss of accuracy. We then show how to adapt the basic algorithm so that it can be boosted to increase predictive performance, yielding multi-instance classifiers that are competitive with the state-of-the-art.

The remainder of the paper is structured as follows: Sections 2 and 3 detail the diverse density framework and its associated algorithms respectively. Section 4 describes the heuristic variants of these algorithms we consider and Section 5 has experimental results. Section 6 shows how to adapt the basic method to perform boosting and presents the improvements in accuracy obtained. We conclude in Section 7.

2 Diverse Density

Diverse density $DD(h) : h \in H \rightarrow \mathbb{R}^+$, as defined in [13, 14], is an objective function for determining the best hypothesis $\hat{h}$ in a certain class of probabilistic multi-instance classifiers H. The objective function is designed to model the intuition that under the standard MI assumption, the positive region of the instance space $\chi = \mathbb{R}^d$ is most likely to be close to instances from many different (i.e. "diverse") positive bags. In [13, 14], diverse density is assumed to be proportional to the posterior density for the model parameters, so under this interpretation maximizing diverse density corresponds to finding a maximum a-posteriori estimate $\hat{h}_{MAP}$.

As this posterior density is not defined in terms of a conditional likelihood in [13, 14], it is not immediately obvious that it is an appropriate objective function for classification learning. The "diverse"-ness property is only a heuristic motivation for $DD(h)$.

However, diverse density can instead be understood as a conditional likelihood function under a different interpretation of some terms[1] (see also [23]). This interpretation is what we use here because it is more in line with work in probabilistic machine learning, and it motivates diverse density learning as a principled maximum-likelihood estimate of a discriminative model. Additionally, under this interpretation, we do not have to resort to cross-validation on the training set in order to choose a decision boundary[2].

Consider the task of learning a discriminative model $Pr(+|B_i, h)$ for predicting the probability that a bag B_i is positive, given a hypothesis h. Let $Pr(+|B_{ij}, h)$ be the probability that instance j of bag B_i is positive. Assuming independence, $Pr(+|B_i, h) = 1 - \prod_j (1 - Pr(+|B_{ij}, h))$, the probability that at least one instance is positive, in line with the standard MI assumption. Further assuming that bags are iid, the conditional likelihood function for h given a dataset D is

$$\begin{aligned} L(h) &= Pr(Y|D, h) \\ &= \prod_i Pr(+|B_i^+, h) \prod_i Pr(-|B_i^-, h) \\ &= \prod_i \left(1 - \prod_j (1 - Pr(+|B_{ij}^+, h))\right) \prod_i \left(\prod_j (1 - Pr(+|B_{ij}^-, h))\right), \end{aligned}$$

where Y is the set of labels for the bags in D, and the B_i^+s and B_i^-s are the positive and negative training bags, respectively.

The model parameters to be learnt are $h = \{x_1, \ldots x_d, s_1, \ldots, s_d\}$, where $x \in \chi$ is the location of the "target point" identifying the positive region of the instance space, and s is the feature scaling vector. To complete the model, we still need to specify $Pr(+|B_{ij}, h)$. Maron and Lozano-Perez use a radial "Gaussian-like" function $Pr(+|B_{ij}, h) = \exp(-||B_{ij} - x||^2)$, with $||B_{ij} - x||^2 = \sum_k s_k^2 (B_{ijk} - x_k)^2$. In other words, it is assumed that the probability that instance j of bag B_i is positive drops exponentially with distance from point x, with the scaling of each dimension k in feature space determined by s_k.

Using this form for $Pr(+|B_{ij}, h)$, the conditional likelihood function is identical to Maron and Lozano Perez' diverse density function $DD(h)$ under what they call the "noisy-or model" (so named because the $Pr(+|B_i^+, h)$ term corresponds to a probabilistic version of a logical "or"). Hence, maximizing the (log) likelihood of this model is equivalent to maximizing diverse density. Maron also formulated the "most-likely-cause" model for diverse density learning. With this model, the likelihood needs to be modified so that $Pr(+|B_i, h) = \max_j Pr(+|B_{ij})$. The $\max$ operator can be viewed as an approximation to a logical "or", so the most-likely-cause model is also consistent with the standard MI assumption.

[1] Specifically, Maron and Lozano Perez' "$Pr(h|B)$" and "$Pr(h|B_{ij})$" are interpreted as $Pr(+|B, h)$ and $Pr(+|B_{ij}, h)$ respectively.

[2] This fact was also exploited implicitly in EM-DD [25].

3 Existing Diverse Density Algorithms

In this section we discuss existing MI learning algorithms that are based on the diverse density framework. In the next section we introduce new variants of these methods that are based on simple heuristics designed to reduce training time while maintaining classification accuracy on new data.

The original maxDD diverse density algorithm attempts to find the target concept by maximizing diverse density (i.e. conditional likelihood) over the instance space, using gradient ascent with multiple starting points. Because the feature scaling vector is also unknown, it is optimized simultaneously (and initialized with all values set to 1.0). Restarts are performed at every instance from each positive bag, as the target point is necessarily close to some of those instances.

Maron [13] also proposed an alternative DD maximization algorithm, Pointwise Diverse Density (PWDD), which was designed to speed up the training process. However, he did not evaluate PWDD, except on a simple artificial dataset that is used to illustrate the underlying idea. One contribution of this paper is that we compare PWDD to maxDD (and variants thereof) on a collection of datasets and thus attempt to close this gap in the literature.

While maxDD uses a gradient ascent method to search for the point of maximum diverse density—with different starting points—PWDD only computes diverse density at exactly the points corresponding to instances in positive bags in the training data. For each positive bag, the algorithm selects the instance with the highest diverse density, and the output hypothesis is the average point of these selected instances.

This version of the DD algorithm performs no gradient optimization and is therefore extremely fast; however it requires the feature scaling vector to be known. In practice, this is typically not the case, so the method must be extended to find the best scaling vector. Maron proposes several alternatives (but does not name them; the names are ours):

- *Scaling First* For each instance in a positive bag, perform gradient ascent to optimize the feature scaling vector for the highest diverse density at that point. Select the vector that produced the highest diverse density, and use this for PWDD.
- *Iterative* Pick an initial scaling, then use PWDD to find the best point (or points) with that scaling. Then use gradient ascent to optimize the scaling vector at the best point(s). Repeat using the new scaling.

Maron also mentions potential variants where the selection of the best instance is incorporated into the gradient ascent optimization routine by replacing the $\max$ operator with the differentiable $softmax$ function. However, we do not consider these softmax-based variants in this paper.

Zhang and Goldman [25] later formulated the EM-DD algorithm, a variant of the maxDD algorithm that is based on the expectation-maximization (EM) approach. The algorithm starts with an initial guess h of the target concept, obtained by selecting a point from a positive bag. It then performs an iterative procedure consisting of an *expectation* step followed by a *maximization* step.

The expectation step selects the instance from each bag that is most likely to be the cause of that bag's label given the current hypothesis h, using the most-likely-cause

estimator. Then, the maximization step performs a gradient ascent search based on the selected instances to find a new h' that maximizes $DD(h')$. The current hypothesis h is reinitialized to h'. The EM loop is repeated until convergence.

Considering computational efficiency, EM-DD has an advantage over maxDD, as it selects only a single instance from each positive bag during the expectation step, which reduces the computational difficulty of the maximization step. Hence, EM-DD scales well with increasing bag size.

The authors of EM-DD initially reported improved performance over maxDD on the *musk* data, but Andrews et al. [1] pointed out that the original formulation of EM-DD selected the best hypothesis based on error rate on the test data. If this is corrected, the algorithm's accuracy is not generally superior to that of maxDD. However, it is still notable for its improved computational efficiency and this is why it is included in the experimental comparison presented in this paper.

Note that there are also algorithms related to the diverse density framework that are not based on direct optimization. They include DD-SVM [5] and its successor MILES [4], which create an instance-based feature space mapping by using diverse density to compute a similarity function between a bag and an instance (which corresponds to a feature in a new instance space), and build a support vector machine on the transformed dataset. The boosted diverse density approach presented in Section 6 generates a similar classifier because it also produces a linear combination of contributions from diverse density target points, where each training instance can potentially become a target point. However, in contrast to MILES, it allows the scaling of features in the original feature space to be adapted to the data at hand—automatically and individually for each target point. In Section 5, we compare the empirical performance of our approach with that of MILES.

4 A New Approach: QuickDD

In this section, we describe some simple variants of existing diverse density maximization algorithms that are designed to improve the computational efficiency of these techniques.

As Maron and Lozano-Pérez observed, the point of maximum diverse density is by definition close to instances from positive bags. A simple heuristic when searching for this point is therefore to only consider the exact locations of instances in positive bags. This heuristic is also used in PWDD, but then the best points from the different positive bags are averaged to form a hypothesis. We propose an even simpler approach, where the merging step is omitted, and we simply pick the best point from all positive bags as our target point.

We will refer to this heuristic as *QuickDD*, as we expect quicker execution over the standard gradient ascent approaches because the search space is greatly reduced in this method. We hypothesized that *(a)* merging of candidate points as in PWDD can produce undesirable target points and *(b)* gradient ascent search over instance space as in maxDD is unnecessary, as an instance from a positive training bag is a sufficient approximation of the target point in real-world problems. Regarding hypothesis *(a)*, the average of the highest diverse density points from each positive bag is not guaranteed

to be a high diverse density point. If the best points for the positive bags belong to different local diverse density maxima, the average point may be in the trough between the maxima, and may not be the best hypothesis.

If the optimal feature scaling is already known, the execution of the maxDD algorithm with the QuickDD heuristic is very efficient: we merely compute the diverse density at each instance from each positive bag, and select the location of the instance with the highest diverse density, without performing any gradient optimization. Again, this is simpler than the PWDD approach, as we do not find the best instance from each positive bag and average the results, but merely select the best instance and use this as the target point.

If the optimal feature scaling is not known in advance, we must incorporate some method to compute it. The methods proposed by Maron for PWDD can easily be adapted for QuickDD:

- *Scaling Only* For each instance in a positive bag, perform gradient ascent to optimize the feature scaling vector at that point. The hypothesis is the point and associated scaling vector that produces the highest diverse density. Here, although the gradient ascent optimization must be performed as many times as for maxDD, the number of parameters is halved because the coordinates of the target point are not optimized.
- *Iterative* Pick an initial scaling, then compute the diverse density of the points in all positive bags with respect to that scaling, and select the point (or points) with the highest diverse density. Then use gradient ascent to optimize the scaling vector at the best point(s). Repeat using the new scaling.

A motivation for using QuickDD *Iterative* over PWDD *Iterative* is that the former monotonically increases the diverse density of the current hypothesis in each iteration—since it will not pick a scaling vector or target point location with a lower DD value than that of the current hypothesis—and thus is guaranteed to converge to a local DD maximum (or saddle point), while the averaging step in PWDD means that there are no such guarantees for that algorithm.

Note that the *Scaling Only* method is also applicable to EM-DD, by restricting the gradient search performed in each iteration to only optimize the scaling parameters for a fixed location in instance space.

We also consider the following simplifications of the above approaches:

- *No Scaling* Initialize all entries of the feature scaling vector to the same value (i.e. 1.0 if we follow [14]). Compute the diverse density for each instance in each positive bag, and select the point with the highest diverse density. Do not perform any gradient ascent optimization.
- *Scaling Once* Initialize the scaling vector as above. Compute the diverse density of the points in all positive bags with respect to that scaling, and select the point with the highest diverse density. Then use gradient ascent to optimize the feature scaling vector at that point. This is equivalent to the QuickDD *Iterative* method with the maximum number of iterations set to one.
- *Scaling Last* This is an adaption of the *Scaling Once* method to PWDD. Execute PWDD to find a point with high diverse density with respect to an initial feature

Table 1. Datasets used in the Experiments

Name	Number of Bags	Number of Attributes	Avg. Number of Instances per Bag	Min. Number of Instances per Bag	Max. Number of Instances per Bag
musk1	92	166	5.2	2	40
musk2	102	166	64.7	1	1044
muta-atoms	188	10	8.6	5	15
muta-bonds	188	16	21.3	8	40
muta-chains	188	24	28.5	8	52
elephant	200	230	7.0	2	13
fox	200	230	6.6	2	13
tiger	200	230	6.1	1	13
maron	50	2	50.0	50	50

Table 2. Percentage Accuracy for EM-DD, maxDD and PWDD

Dataset	EM-DD	maxDD	PWDD Scaling First		PWDD Iterative		PWDD Scaling Last		PWDD No Scaling	
musk1	85.4±11.6	86.8±11.5	86.9±10.2		86.5±11.5		85.5±11.5		48.9±4.9	●
musk2	85.6±9.8	85.7±9.6	86.3±10.6		85.1±10.0		84.9±10.2		62.7±9.9	●
muta-atoms	72.2±8.4	72.2±10.1	36.1±7.3	●	42.9±15.0	●	64.1±6.5	●	66.5±2.3	
muta-bonds	73.0±10.1	73.9±9.5	52.1±17.0	●	70.4±8.9		73.2±8.5		66.5±2.3	
muta-chains	73.5±11.1	79.1±7.6	67.8±7.0		65.2±11.1		80.3±7.8		66.5±2.3	
elephant	75.9±10.3	81.9±8.9	78.5±6.3	10	82.6±8.7	○	82.6±8.8	○	56.4±5.9	●
fox	60.3±8.5	61.3±10.8	59.7±8.9		62.3±10.7		60.4±10.9		55.4±5.3	
tiger	71.9±10.3	75.4±9.7	72.0±9.6		70.5±11.0		71.4±9.9		49.9±6.8	●
maron	93.4±11.0	96.4±7.7	94.2±10.0		94.4±9.9		94.4±9.9		61.6±14.4	●

○, ●: significant increase or decrease vs EM-DD; number in small font: completed runs

scaling vector, then perform a gradient ascent search to optimize the feature scaling vector at that point, i.e. perform a single iteration of PWDD *Iterative*.

5 Experimental Results

In this section we present the results of an empirical study of the classification performance and training time of the algorithms discussed above. The algorithms were evaluated on a variety of two-class datasets by averaging the results of ten repeats of ten-fold cross-validation, measuring both classification accuracy and training time. The datasets used were:

- *elephant*, *fox*, *tiger* Content-based image retrieval datasets, originally provided by [1]. The MI bags represent photographs of animals, and the task is to predict whether an image contains the target animal (elephants, foxes and tigers, respectively).
- *musk1*, *musk2* The musk data used in [7]. Each bag represents a molecule, and the task is to predict whether the molecule emits a musky odour. The *musk2* dataset is larger, both in terms of the number of molecules, and the number of instances per molecule.
- *mutagenesis* The mutagenicity prediction problem [18], widely used as a benchmark for ILP algorithms. The learning problem is to identify mutagenic molecules.

Table 3. Training Times in CPU seconds (s), minutes (m) or hours (h) for EM-DD, maxDD and PWDD

Dataset	EM-DD	maxDD	PWDD Scaling First	PWDD Iterative	PWDD Scaling Last	PWDD No Scaling
musk1	2.7m±35.2s	22.7m±2.6m ○	9.1m±43.2s ○	6.6s±1.5s ●	2.8s±0.6s ●	0.4s±0.0s ●
musk2	29.6m±27.7m	23.6h±9.3h ○	24.5h±7.1h ○	4.9m±2.1m ●	1.6m±45.8s ●	36.1s±4.9s ●
muta-a	0.9s±0.4s	6.4m±2.6m ○	27.4s±1.4s ○	1.9s±0.8s ○	0.8s±0.1s	0.9s±0.0s
muta-b	6.4s±2.2s	49.8m±2.4m ○	7.1m±19.0s ○	14.5s±4.1s ○	9.4s±0.5s ○	7.7s±0.1s
muta-c	16.9s±3.7s	6.1h±2.5h ○	32.8m±50.2s ○	27.0s±7.1s ○	21.3s±0.8s ○	18.9s±0.3s
elephant	20.1m±3.9m	12.7h±4.2h ○	165.8h±47.1h 10	28.5m±33.5m	19.3m±15.9m	6.1s±0.1s ●
fox	14.9m±3.5m	13.7h±3.6h ○	19.7h±19.0h ○	1.9m±2.1m ●	2.3m±3.7m ●	4.9s±0.1s ●
tiger	9.5m±2.4m	6.2h±2.2h ○	11.2h±7.4h ○	1.9m±1.4m ●	2.2m±1.8m ●	3.8s±0.1s ●
maron	2.5s±0.1s	4.2m±12.5s ○	1.3m±2.2s ○	4.0s±1.4s ○	1.2s±0.0s ●	1.1s±0.0s ●

○, ●: statistically significant increase or decrease vs EM-DD; number in small font: completed runs

Three representations of molecules were used [17]: *muta-atoms*, *muta-bonds* and *muta-chains*.

– *maron* An artificial dataset based on one used in [14]. For each bag, 50 instances were sampled from a uniform distribution in $[0, 100] \times [0, 100] \subseteq \mathbb{R}^2$. Instances were *positive* if and only if they were within a 5×5 square in the middle of the domain, thus implementing the standard MI assumption. We generated 25 positive and 25 negative bags.

Key statistics of these datasets are summarized in Table 1.

The experiments were performed using WEKA [22], on 3.00 GHz Intel Pentium 4 CPU machines. All implementations were based on those of maxDD and EM-DD in WEKA, which use a quasi-Newton method with BFGS updates rather than plain gradient search. The details of this method can be found in Appendix B of [23]. For numeric stability, the negative logarithm of DD is minimized instead of maximizing DD directly.

The default behavior of the WEKA implementation of maxDD is to only consider instances from the *largest* positive bag as starting points for the optimization; we modified this to consider instances from *all* positive bags, to be consistent with the original description of maxDD. EM-DD was executed using instances from three random positive bags as starting points [25].

All iterative algorithms were restricted to a maximum of 10 iterations. We also applied normalization of attributes to the $[0, 1]$ interval to all datasets except for *maron*, as all of the algorithms typically performed poorly without this. We tested for significant differences between algorithms using the corrected resampled *t*-test [15] with significance level $\alpha = 0.05$.

Tables 2 and 3 display the accuracy and training time results for the three pre-existing algorithms EM-DD, maxDD and PWDD. Note that some of the 100 runs for the more expensive methods did not complete in time for submission. In those cases, no significance test was performed and the number of completed runs is given in small font next to the corresponding entry.

The results are consistent with the observations in [1], who disputed the superior classification performance of EM-DD over maxDD that was reported in earlier work: EM-DD performed similarly to maxDD on all datasets. However, its training time was

Table 4. Percentage Accuracy for EM-DD and maxDD, Using 3 Random Positive Bags for Starting Points

Dataset	EM-DD	maxDD
musk1	85.4±11.6	87.0±11.4
musk2	85.6±9.8	85.8±10.1
muta-atoms	72.2±8.4	71.5±8.8
muta-bonds	73.0±10.1	74.1±9.5
muta-chains	73.5±11.1	79.2±7.7
elephant	75.9±10.3	81.9±8.5 ○
fox	60.3±8.5	60.8±10.8
tiger	71.9±10.3	75.6±9.4
maron	93.4±11.0	96.4±7.7

○, ● statistically significant increase or decrease vs EM-DD

Table 5. Training Times in CPU seconds (s), minutes (m) or hours (h) for EM-DD and maxDD, Using 3 Random Positive Bags for Starting Points

Dataset	EM-DD	maxDD
musk1	2.7m±35.2s	1.8m±27.0s ●
musk2	29.6m±27.7m	4.2h±3.6h ○
muta-atoms	0.9s±0.4s	13.1s±2.8s ○
muta-bonds	6.4s±2.2s	3.2m±41.6s ○
muta-chains	16.9s±3.7s	12.7m±2.1m ○
elephant	20.1m±3.9m	19.5m±5.2m
fox	14.9m±3.5m	15.6m±5.3m
tiger	9.5m±2.4m	8.8m±2.9m
maron	2.5s±0.1s	24.3s±1.9s ○

○, ● statistically significant increase or decrease vs EM-DD

several orders of magnitude lower in all cases. Hence, EM-DD is a worthwhile candidate in practical applications of MI learning.

When interpreting this result, it is important to remember that maxDD was executed using all instances from positive training bags as starting points for the gradient search, while EM-DD only used instances from three random positive bags. To isolate the effect of this modification, we performed a separate experiment where we used the same heuristic to reduce the number of starting points for the search in maxDD. Even with this modification maxDD frequently remains orders of magnitude slower than EM-DD. This can be seen from the results shown in Tables 4 and 5. Note that with this change to maxDD, the difference in accuracy on the elephant dataset becomes statistically significant.

Table 2 shows that all variants of PWDD suffered at least one significant loss in classification accuracy against EM-DD. In particular, PWDD struggled on the mutagenesis datasets, where most variants of the algorithm failed to improve on the 66.5% accuracy rate obtained by predicting the majority class. PWDD *No Scaling*, the variant of PWDD where no gradient search was used to optimize the scaling vector, only exceeded the majority class baseline on three datasets (*elephant*, *fox* and *maron*), demonstrating the importance of scaling features appropriately. PWDD *Scaling First* also performed quite poorly, indicating that undesirable scaling vectors were chosen. Moreover, PWDD *Scaling First* exhibited larger training times than even maxDD on the image datasets,

Table 6. Percentage Accuracy for QuickDD Variants vs EM-DD

Dataset	EM-DD	QuickDD No Scaling	QuickDD Scaling Only	QuickDD Iterative	EM-DD Scaling Only	QuickDD Scaling Once
musk1	85.4±11.6	49.0±4.9 ●	86.1±11.4	86.7±11.1	84.1±12.2	86.4±10.4
musk2	85.6±9.8	62.0±7.3	86.1±11.0	87.4±11.3	83.7±10.5	87.2±11.4
muta-atoms	72.2±8.4	64.5±4.8 ●	70.9±8.2	68.5±8.2	75.6±9.5	68.5±8.2
muta-bonds	73.0±10.1	73.6±7.0	74.0±9.4	76.7±8.3	71.8±9.5	76.7±8.3
muta-chains	73.5±11.1	75.5±7.2	80.4±8.5	78.5±7.9	73.4±8.9	78.4±7.9
elephant	75.9±10.3	72.7±9.6	80.7±9.1 $_{30}$	81.1±8.8	76.2±9.9	81.8±8.9
fox	60.3±8.5	53.7±11.3	60.3±11.2 $_{57}$	64.0±10.3	61.0±10.4	64.0±10.3
tiger	71.9±10.3	60.0±8.9 ●	74.3±10.1	75.1±10.1	72.5±10.5	75.5±9.6
maron	93.4±11.0	61.4±14.3 ●	96.2±8.4	96.2±8.4	89.4±13.5	96.8±8.4

○, ●: significant increase or decrease vs EM-DD; number in small font: completed runs

Table 7. Training Times in CPU seconds (s), minutes (m) or hours (h) for QuickDD Variants vs EM-DD

Dataset	EM-DD	QuickDD No Scaling	QuickDD Scaling Only	QuickDD Iterative	EM-DD Scaling Only	QuickDD Scaling Once
musk1	2.7m±35.2s	0.4s±0.0s ●	8.2m±39.6s ○	7.7s±2.9s ●	36.2s±7.7s ●	1.7s±0.6s ●
musk2	29.6m±27.7m	36.2s±4.9s ●	22.7h±8.9h ○	6.0m±3.5m ●	8.1m±7.1m ●	4.2m±3.0m ●
muta-a	0.9s±0.4s	0.6s±0.0s ●	58.0s±3.1s ○	3.3s±1.9s ○	0.3s±0.1s ●	0.7s±0.0s
muta-b	6.4s±2.2s	4.5s±0.1s ●	14.3m±39.2s ○	49.9s±25.6s ○	3.3s±0.8s ●	5.0s±0.2s
muta-c	16.9s±3.7s	10.4s±0.2s ●	55.4m±3.2m ○	1.3m±44.4s ○	12.0s±2.5s ●	11.8s±0.4s ●
elephant	20.1m±3.9m	3.1s±0.1s ●	80.7h±54.8h $_{30}$	10.6m±13.1m	1.5h±45.8m ○	4.8m±6.7m ●
fox	14.9m±3.5m	2.5s±0.0s ●	29.3h±26.3h $_{57}$	2.7m±3.3m ●	16.5m±7.1m	1.2m±1.6m ●
tiger	9.5m±2.4m	1.9s±0.0s ●	8.7h±4.4h ○	1.5m±1.4m ●	8.2m±4.8m	39.7s±35.6s ●
maron	2.5s±0.1s	0.5s±0.0s ●	55.7s±1.7s ○	4.4s±1.6s ○	0.9s±0.0s ●	1.2s±0.0s ●

○, ●: statistically significant increase or decrease vs EM-DD; number in small font: completed runs

indicating that optimizing the scaling vector only can result in hard optimization problems when the corresponding candidate target point is not appropriate.

However, the *Iterative* and *Scaling Last* variants of PWDD were quite competitive with EM-DD overall, both in terms of classification accuracy and training time. Both achieved a significant win against EM-DD on the *elephant* data, while suffering a significant loss on *muta-atoms*, with no other significant differences. Both were significantly faster than EM-DD on all datasets except the three *mutagenesis* problems and *elephant*, exhibiting very fast training times on the two *musk* datasets. It is noteworthy that the single-iteration *Scaling Last* variant was very competitive with *Iterative* PWDD, where the maximum number of iterations was set to ten.

The results for QuickDD are summarized in Tables 6 and 7, and compared to EM-DD. We can see that except for the *No Scaling* variant — which performed poorly, as expected — and not withstanding the incomplete results for QuickDD *Scaling Only*, there were no significant differences for any of the QuickDD variants against EM-DD with respect to classification accuracy. Additionally, several QuickDD variants were superior in terms of training time.

The tables also show results for the EM-DD variant *Scaling Only*. It was significantly faster than the original EM-DD algorithm on six of the nine datasets, and only significantly slower on the *elephant* dataset, without any significant differences in classification accuracy.

Table 8. Percentage Accuracy for QuickDD Scaling Once and PWDD Scaling Last

Dataset	PWDD Scaling Last	QuickDD Scaling Once
musk1	85.5±11.5	86.4±10.4
musk2	84.9±10.2	87.2±11.4
muta-atoms	64.1±6.5	68.5±8.2
muta-bonds	73.2±8.5	76.7±8.3
muta-chains	80.3±7.8	78.4±7.9
elephant	82.6±8.8	81.8±8.9
fox	60.4±10.9	64.0±10.3
tiger	71.4±9.9	75.5±9.6
maron	94.4±9.9	96.8±8.4

No significant differences were observed

Table 9. Training Times in CPU seconds (s) or minutes (m) for QuickDD Scaling Once and PWDD Scaling Last

Dataset	PWDD Scaling Last	QuickDD Scaling Once	
musk1	2.8s±0.6s	1.7s±0.6s	●
musk2	1.6m±45.8s	4.2m±3.0m	○
muta-atoms	0.8s±0.1s	0.7s±0.0s	
muta-bonds	9.4s±0.5s	5.0s±0.2s	●
muta-chains	21.3s±0.8s	11.8s±0.4s	●
elephant	19.3m±15.9m	4.8m±6.7m	●
fox	2.3m±3.7m	1.2m±1.6m	
tiger	2.2m±1.8m	39.7s±35.6s	●
maron	0.9s±0.0s	0.9s±0.0s	

○, ● statistically significant increase or decrease vs PWDD Scaling Last

QuickDD *Iterative* yielded an equal number of significant wins and losses for training time against EM-DD, but the wins were by a large margin on the slowest datasets, while the losses occurred only on datasets where the training times were already short. Furthermore, QuickDD *Iterative* and *Scaling Once* both had a higher classification accuracy than EM-DD on eight of the nine datasets, though these differences were not individually statistically significant.

It is interesting to compare the runtime of QuickDD *Scaling Only* with that of maxDD from Table 3. The former was faster on the *musk* and *mutagenesis* datasets, but slower on the image datasets. This behaviour is similar to that of PWDD *Scaling First*, which we discussed above. In both cases, the dimensionality of the search space for the gradient optimization routine is halved relative to maxDD, but training time increases, implying the occurrence of harder optimization problems.

Similarly to the PWDD case, QuickDD *Iterative* performed just as well with one iteration (*Scaling Once*) as with a maximum of ten, with dramatic reductions in training time. Thus, repeated iterations appear unnecessary for both PWDD and QuickDD *Iterative*. This indicates that the initial scaling factor of 1.0 for all attributes may be sufficient for finding the location of a good hypothesis, perhaps aided by the dataset normalization step performed.

There were no significant differences in accuracy between the single-iteration versions of PWDD and QuickDD (Table 8), but QuickDD *Scaling Once* was faster that

PWDD *Scaling Last* on eight of the nine datasets, (Table 9) with only one loss with respect to training time. Additionally, as Table 7 shows, QuickDD *Scaling Once* was significantly faster than EM-DD on seven of the nine datasets and still faster on the same seven datasets when the *Scaling Only* heuristic was applied in EM-DD. This is despite the fact that all points in all positive training bags were considered as candidate target points by the algorithm, while EM-DD only considered instances from three random bags as starting points. This shows that QuickDD *Scaling Once* is faster overall than all previous algorithms, while retaining the classification performance of the slower methods.

6 Boosting Diverse Density Learning

The above results show that simple heuristics can improve the runtime of diverse density learning. However, they do not increase accuracy in a significant manner. In this section, we discuss what modifications are required to successfully apply boosting to diverse density learning, and present experimental results demonstrating that significant increases in accuracy can be obtained in this manner. We use the *Real AdaBoost* algorithm described in [10]. In contrast to the original AdaBoost method [9], which is based on 0/1 predictions from the weak learner, this boosting method can exploit predictions that are class probability estimates.

As in the original AdaBoost, *Real AdaBoost* is a sequential process for learning an ensemble of weak classifiers. In each iteration, a weak classifier is learned based on a reweighted version of the training data. Initially, all examples receive the same weight. In subsequent iterations of the boosting process, the weight of an example, i.e. bag B in the context considered here, is updated based on the current hypothesis h using the following equation:

$$w := w \times e^{-0.5 \log \frac{Pr(\{+,-\}|B,h)}{1-Pr(\{+,-\}|B,h)}}$$

where $Pr(\{+,-\}|B,h)$ is the predicted class probability for the observed class of the bag (either $+$ or $-$). Thus, the square root of the predicted odds ratio for the observed class label determines the update. To reduce the likelihood of overfitting, the exponent can be moderated by multiplying it with a shrinkage value $s \in (0,1]$.

Boosting diverse density learning becomes computationally feasible by applying the QuickDD *Scaling Once* variant discussed above as the weak learner. *Real AdaBoost* requires the underlying learning algorithm to be able to deal with weighted examples, but it is straightforward to modify diverse density learning to do this by replacing the likelihood function with a weighted likelihood and adapting the gradient correspondingly. If w_i is the weight of a bag, then we now maximize:

$$\sum_i w_i^+ \log Pr(+|B_i^+, h) + \sum_i w_i^- \log Pr(-|B_i^-, h).$$

However, application of *Real AdaBoost* to the datasets considered above does not yield significant improvements in accuracy compared to applying stand-alone QuickDD *Scaling Once* itself directly to the data. We found that two further changes to the diverse density method are critical to render application of boosting successful:

Table 10. Percentage Accuracy for Boosted QuickDD *Scaling Once* and (optimized) MILES

Dataset	No boosting	10 boosting iterations	100 its. shrink. 0.5	Best MILES configuration
musk1	86.4±10.4	88.2±11.5	89.8±10.9	89.1
musk2	87.6±11.4	88.1±10.0	90.8±9.1	91.6
mutagenesis3-atoms	68.5±8.2	80.6±7.6 ∘	84.7±7.2 ∘	83.9
mutagenesis3-bonds	76.7±8.3	80.8±8.9	87.6±7.5 ∘	86.3
mutagenesis3-chains	78.4±7.9	80.3±7.8	84.6±7.8	86.0
tiger	75.6±9.6	81.1±9.4	82.1±9.2	81.7
fox	64.0±10.3	62.2±9.1	64.4±8.7	64.9
elephant	81.7±9.0	84.5±8.2	86.9±7.9	84.1

∘ statistically significant increase vs baseline
(no significance tests were performed wrt MILES-based results)

1. **Symmetric learning** Diverse density learning as discussed so far requires the user to decide prior to learning which class is to be treated as the positive class. The first modification is to eliminate this requirement: the basic algorithm is run twice, in each class treating one class as the positive class and the other class as the negative one. The final concept output is then the one of the two point-and-scaling concepts—one representing a negative target point and one representing a (traditional) positive one— that maximizes the (weighted) conditional loglikelihood. This means that different classes can be viewed as the positive class in different iterations of the boosting process.
2. **One-sided prediction** Perhaps the most important change is to localize the influence of each diverse density classifier in the instance space. To this end, we change the diverse density model so that the probability predicted for the positive class can never drop below 0.5 (and, consequently, the probability for the negative class can never exceed 0.5). The new model is:

$$Pr(+|B_i, h) = 1 - 0.5 \times \prod_j (1 - Pr(+|B_{ij}, h))$$

This has the effect that the algorithm can abstain from making a prediction in the boosting process: a predicted probability of 0.5 means that the odds ratio becomes 1 in the above weight update. Thus, the influence of a weak classifier can be restricted to a small area around the concept that was found. The likelihood is optimized wrt this adjusted model.

Table 10 compares the accuracy of the boosted QuickDD *Scaling Once* algorithm with the above modifications to that of stand-alone QuickDD *Scaling Once*, which is the baseline in the left-most column, on the real-world datasets used in our study. Results for boosting with 10 iterations and no shrinkage, and boosting with 100 iterations and shrinkage 0.5 are included. To account for class imbalance, the boosting process was initialized with a model that predicts the class prior probabilities from the training data. No shrinkage was applied to the predictions of this initial model. The feature scaling was initialized to 100.0 for the mutagenesis datasets when boosting because poor results were obtained for value 1.0, most likely due to the more localized models being used. Note that runtime (not shown) is linear in the number of boosting iterations.

For reference the table also contains the best results obtained from different variants of the state-of-the-art MILES multi-instance learning method [4], taken from [8]. As discussed at the end of Section 3, MILES produces a similar model. The results for MILES were generated under the same experimental conditions and are thus directly comparable. Note that these results are for the *best* configurations tried—in several cases the performance of the standard MILES approach could be improved by replacing the 1-norm support vector machine from [4] with another learning algorithm [8]—so they are likely to be optimistic. Despite this optimistic bias, we can see that boosted diverse density learning is highly competitive.

7 Conclusions

Our results show that PWDD *Iterative*, a previously proposed MI learning algorithm that has not received much attention in the literature, perhaps due to a lack of published empirical results, is very competitive with the more well-known EM-DD algorithm, both in terms of classification accuracy and training time. Moreover, we found that the repeated iteration of the algorithm is in fact unnecessary on the datasets we considered, as similar accuracy could be achieved with a single iteration of PWDD.

Our simplified QuickDD *Iterative* variant of PWDD, which provides convergence guarantees, improved results further. When restricted to a single iteration (QuickDD *Scaling Once*), the algorithm was very competitive with PWDD and EM-DD for classification accuracy, while enjoying faster training times. Our results show that instances from positive training bags are often a sufficient representation for the location of diverse density target points. This heuristic dramatically reduces the search space, enabling more efficient algorithms for learning diverse density concepts.

We also showed how boosting can be applied in conjunction with QuickDD *Scaling Once* to obtain state-of-the-art accuracy on the datasets investigated. Three changes to the algorithm were necessary to obtain improved accuracy using boosting: incorporation of bag weights, symmetric treatment of classes, and enabling one-sided prediction. With these changes, boosting diverse density learning appears to be a viable and practical alternative to other advanced methods for multi-instance learning.

References

1. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Neural Information Processing Systems. pp. 561–568. MIT Press (2003)
2. Andrews, S., Hofmann, T.: Multiple-instance learning via disjunctive programming boosting. In: NIPS (2003)
3. Auer, P., Ortner, R.: A boosting approach to multiple instance learning. In: European Conference on Machine Learning. pp. 63–74. Springer (2004)
4. Chen, Y., Bi, J., Wang, J.Z.: MILES: Multiple-instance learning via embedded instance selection. IEEE Pattern Analysis and Machine Intelligence 28(12), 1931–1947 (2006)
5. Chen, Y., Wang, J.Z.: Image categorization by learning and reasoning with regions. Journal of Machine Learning Research 5, 913–939 (2004)

6. Chevaleyre, Y., Zucker, J.D.: Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. Application to the mutagenesis problem. In: Conference of the Canadian Society for Computational Studies of Intelligence. pp. 204–214. Springer (2001)
7. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence 89(1–2), 31–71 (1997)
8. Foulds, J.R., Frank, E.: Revisiting multiple-instance learning via embedded instance selection. In: Proc 21st Australasian Joint Conference on Artificial Intelligence. pp. 300–310. Auckland, New Zealand, Springer (2008)
9. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning. pp. 148–156. Morgan Kaufmann (1996)
10. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2), 337–407 (2000)
11. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.: Multi-instance kernels. In: International Conference on Machine Learning. pp. 179–186. Morgan Kaufmann (2002)
12. Krogel, M.A., Wrobel, S.: Feature selection for propositionalization. In: International Conference on Discovery Science. pp. 430–434. Springer (2002)
13. Maron, O.: Learning from ambiguity. Ph.D. thesis, Massachusetts Institute of Technology (1998)
14. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: Neural Information Processing Systems. MIT Press (1998)
15. Nadeau, C., Bengio, Y.: Inference for the Generalization Error. Machine Learning 52(3), 239–281 (2003)
16. Ray, S., Craven, M.: Supervised learning versus multiple instance learning: an empirical comparison. In: International Conference on Machine Learning. pp. 697–704. Omnipress (2005)
17. Reutemann, P.: Development of a Propositionalization Toolbox. Master's thesis, Albert Ludwigs University of Freiburg (2004)
18. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: Inductive Logic Programming. pp. 217–232. GMD-Studien (1994)
19. Viola, P.A., Platt, J.C., Zhang, C.: Multiple instance boosting for object detection. In: NIPS (2005)
20. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: International Conference on Machine Learning. pp. 1119–1125. Morgan Kaufmann (2000)
21. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: European Conference on Machine Learning. pp. 468–479. Springer (2003)
22. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
23. Xu, X.: Statistical Learning in Multiple Instance Problems. Master's thesis, University of Waikato (2003)
24. Xu, X., Frank, E.: Logistic regression and boosting for labeled bags of instances. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 272–281. Springer (2004)
25. Zhang, Q., Goldman, S.: EM-DD: An improved multiple-instance learning technique. In: Neural Information Processing Systems. pp. 1073–1080. MIT Press (2002)
26. Zhang, Q., Yu, W., Goldman, S., Fritts, J.: Content-based image retrieval using multiple-instance learning. In: International Conference on Machine Learning. pp. 682–689. Morgan Kaufmann (2002)