Revisiting Multiple-Instance Learning via Embedded Instance Selection

James Foulds and Eibe Frank

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

Abstract. Multiple-Instance Learning via Embedded Instance Selection (MILES) is a recently proposed multiple-instance (MI) classification algorithm that applies a single-instance base learner to a propositionalized version of MI data. However, the original authors consider only one single-instance base learner for the algorithm — the 1-norm SVM. We present an empirical study investigating the efficacy of alternative base learners for MILES, and compare MILES to other MI algorithms. Our results show that boosted decision stumps can in some cases provide better classification accuracy than the 1-norm SVM as a base learner for MILES. Although MILES provides competitive performance when compared to other MI learners, we identify simpler propositionalization methods that require shorter training times while retaining MILES' strong classification performance on the datasets we tested.

1 Introduction

Multiple-instance (MI) learning is an alternative to the traditional supervised learning model in which learning examples are represented by a *bag* (i.e. multiset) of instances instead of a single feature vector. The MI framework was introduced by Dietterich et al. [7] in the context of a drug-activity prediction problem, where each molecule is represented by a bag of feature vectors corresponding to the conformations (shapes) that the molecule can adopt by rotating its internal bonds. In this problem domain the *standard MI assumption* applies: if and only if at least one instance in a bag is positive (i.e. at least one conformation bonds to the target binding site), then that bag is positive (i.e. the molecule will have the desired drug effect).

Dietterich et al. presented algorithms that learn MI concepts for the *musk* drug activity prediction problem by finding a hyper-rectangle to describe the positive region of instance space. Since then, many other MI learning algorithms have been proposed (see, for example, [1], [2], [10], [12], [14], [24], [25], [27], [28], [29], [30]).

Multiple-instance Learning via Embedded Instance Selection (MILES) is a recent MI learning approach presented by Chen et al. [6], which transforms MI data into a propositionalized form, to which a 1-norm support vector machine (SVM) classifier is applied. Chen et al. do not consider alternatives to the 1-norm SVM, but they do mention briefly that other single-instance base learners

are possible. In this paper we view the algorithm as a meta-classifier that can wrap around an arbitrary single-instance learner. We present an empirical study of the performance of the MILES algorithm using a variety of single-instance base learners on a diverse set of benchmark datasets. The goal of the study is to compare the relative performance of different base learners for MILES, and to compare MILES to existing MI algorithms, including other propositionalization methods. The paper is structured as follows. In Section 2, we describe the MILES algorithm. Section 3 details the experimental setup, the results of the experiment are given in Section 4, and we conclude in Section 5.

2 MILES

Multiple-Instance Learning via Embedded Instance Selection (MILES) [6] is an approach to MI learning based on the diverse density framework [14]. In contrast to standard diverse density algorithms, it embeds bags into a single-instance feature space. Most earlier diverse density-based methods have used the standard MI assumption mentioned above and further assume the existence of a single target point[1]. Instead, MILES uses a symmetric assumption, where multiple target points are allowed, each of which may be related to either positive or negative bags. Under this assumption, and using the *most-likely-cause* estimator from the diverse density framework, Chen et al. define a measure specifying the probability that a point x is a target point given a bag, regardless of the bag's class label:

$$Pr(x|B_i) \;\propto\; s(x, B_i) = \max_j \exp\left(- \frac{\|x_{ij} - x\|^2}{\sigma^2} \right) , \tag{1}$$

where x_{ij} are the instances in bag B_i, and σ is a predefined scaling factor. Note that $s(x, B_i)$ can be interpreted as a measure of similarity between a bag and an instance, determined by the instance x and the closest instance in the bag. MILES uses each instance in the training bags as a candidate for a target point. The candidates are represented as features in an instance-based feature space $\mathbb{F}_c$. Each bag in the training set is mapped into $\mathbb{F}_c$ via the mapping

$$m(B_i) = [s(x^1, B_i), s(x^2, B_i), \cdots, s(x^n, B_i)]^T , \tag{2}$$

where $x^i \in C$ is an instance from the set C of all instances in all of the training bags. When the class labels $c \in \Omega$ of the bags are appended, the resulting space $(\mathbb{F}_c|\Omega)$ is a single-instance feature space. The output of a single-instance classifier trained on this data is used to provide bag-level class labels for future data. The pseudocode of the MILES algorithm is provided in Algorithm 1.

Chen et al. used the 1-norm SVM algorithm as the base classifier, due to the sparsity property of the algorithm — it is known to set most feature weights to zero, which effectively performs feature selection — and the fact that the

[1] This means that, roughly speaking, a bag is assumed to be positive if at least some of its instances are close to this point.

Algorithm 1 MILES

D = the set of training bags; C = all instances in the bags in D
L = a single-instance base learner; σ = the scaling factor

$\underline{train(D)}$
F = an empty set of instances
for (every bag $B_i = \{x_{ij} : j = 1, \cdots, n_i\}$ in D) **do**
    $t = MILES\_transform(B_i)$
    $t.setClassLabel(B_i.getClassLabel())$
    $F = F \cup \{t\}$
$L.train(F)$ //Can optionally perform feature selection here also

$\underline{MILES\_transform(B)}$, $B = \{x_j : j = 1, \cdots, n\}$ a bag
m(B) = an empty instance of dimension $|C|$
for (every instance x^k in C) **do**
    $d = \min_j \|x_j - x^k\|$; the kth element of $m(B)$ is $s(x^k, B) = e^{\frac{-d^2}{\sigma^2}}$
return $m(B)$

$\underline{classify(B)}$, $B = \{x_j : j = 1, \cdots, n\}$ a test bag
$t = MILES\_transform(B)$; **return** $L.classify(t)$

resulting learning problem is usually very high-dimensional. They do not consider alternative base learners, however. We investigate the use of alternative base learners for MILES, and compare the algorithm to other MI approaches.

3 Experiment Design

An extensive set of experiments was performed on a number of multi-instance datasets, using a wide range of MI algorithms and single-instance base learners. The experiments were performed using the WEKA workbench [26]. Each algorithm was evaluated on each dataset by 10 times stratified 10-fold cross-validation. Performance was measured using classification accuracy. We tested for significant differences between algorithms using the corrected resampled t-test [17] with significance level $\alpha = 0.05$.

WEKA implementations were used for all MI algorithms and single-instance base learners, with the exception of MILES and the 1-norm SVM, which were implemented specifically for the experiment. Default parameters were used for each algorithm unless otherwise specified. The MI algorithms were *MILES*, *MISMO* (SVM with the MI polynomial kernel [12]), *mi-SVM* [1], *Citation-KNN* [24], *EMDD* [29], *Adaboost + Optimal Ball* [2], *MIBoost* [27] (with the WEKA REPTree decision tree learner, with no automatic pruning but depth-limited to 3 levels, as the base classifier), *MILR* [27] (using the noisy-or model to combine instance-level probabilities), *MIWrapper* [10], and *SimpleMI* [8].[2]

[2] Where the algorithm was not explicitly named by the original authors, the name of the WEKA implementation has been used instead.

Of particular interest are MIWrapper and SimpleMI, which, similarly to MILES, are wrapper algorithms that apply a single-instance base learner to a propositionalized version of the given MI data. MIWrapper performs propositionalization by applying bag-level class labels to instances, and weighting the instances so that each bag has the same total weight. A single-instance model is built on the resulting dataset, and bag-level predictions are made by averaging the predicted probabilities of instances in a bag. SimpleMI performs propositionalization by averaging the attribute values of the instances in each bag, and appending the bag's class label to the resulting feature vector. The wrapper algorithms were evaluated using the following single-instance base learners: *C4.5* [20], *random forests* (100 trees) [5], *Adaboost* [11] + *C4.5* (10 iterations), *Adaboost* + *decision stumps* (1-level decision trees, 100 iterations), *bagging* [4] + *C4.5* (10 iterations), *2-norm SVMs* (SMO) [19] with a linear kernel and a radial-basis function kernel, the linear *1-norm SVM*, and *logistic regression*.

The datasets are described very briefly here, with the names of the datasets (as labeled in the results tables) italicized for convenience. The *musk1* and *musk2* datasets are the musk data used in [7]. Each bag represents a molecule, and the task is to predict whether the molecule emits a musky odour. *Eastwest* is the train direction prediction problem from the East West Challenge ILP contest [16]. *Westeast* is exactly the same problem as *eastwest*, except that the class labels are reversed. This is an interesting variation because *eastwest* is compatible with the standard MI assumption, while *westeast* is not [8].

The mutagenicity prediction problem [22] was also used in the experiments. Three representations proposed by [21] for transforming the mutagenesis ILP problem into a multi-instance problem were used, which were labeled *muta-atoms*, *muta-bonds* and *muta-chains*. The *suramin* dataset [3] is another ILP-based drug activity prediction problem, where the task is to detect suramin analogues that can act as anti-cancer agents. The *thioredoxin* dataset is the thioredoxin-fold protein identification task proposed by [23].

Two sets of image data for Content-based Image Retrieval (CBIR) tasks were used, each containing three different image categories. These image databases provided six different image retrieval problems — one for each image category, with the task being to identify images belonging to the target category. The first image database was originally provided by [1], and contains MI bags representing photographs of *elephants*, *foxes* and *tigers* from the Corel dataset. The second CBIR dataset was the GRAZ02 [18] dataset, containing images of *bikes*, *cars* and *people*, with features derived from the Ohta colour space representations of the image as in [15].

4 Experimental Results and Analysis

This section presents a comparison of base learners for MILES, and compares the algorithm with other MI learning methods. The reader is referred to the first author's MSc thesis for more detailed experimental results [9].

Table 1. MILES: Percentage Accuracy for Non-Ensemble Base Learners

Dataset	1-Norm SVM	C4.5	Logistic Regression	SMO (LIN)	SMO (RBF)
musk1	83.3±11.8	84.1±11.9	84.8±11.3	86.9±10.4	89.1±10.1
musk2	91.6±8.3	82.5±12.1 •	85.8±11.0	88.4±9.7	79.5±12.5 •
eastwest	74.0±25.1	50.0±0.0 •	64.5±29.6	55.5±30.9	55.5±31.7
westeast	74.0±25.1	50.0±0.0 •	68.5±33.1	54.0±30.7	54.5±31.1
muta-atoms	74.8±14.4	80.8±8.1	83.8±7.2	80.8±8.8	83.7±9.2
muta-bonds	72.2±12.7	77.1±9.8	80.2±8.8	79.8±9.5	81.8±8.9 ○
muta-chains	75.9±9.2	79.3±9.5	73.5±9.4	77.9±8.5	78.6±10.3
suramin	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2
thioredoxin	88.1±5.1	84.3±7.1	87.1±3.9*	69.1±10.3 •	86.3±4.3
elephant	84.1±8.9	77.5±9.2	79.6±9.1	83.9±9.0	83.4±8.9
fox	63.0±9.5	56.8±11.2	63.6±8.9	64.8±9.5	64.2±9.7
tiger	80.7±8.3	69.7±9.3 •	80.0±9.2	81.5±8.4	81.7±8.9
bikes	78.4±4.2	72.5±5.7 •	72.4±4.8 •	80.1±4.9	78.7±4.9
cars	72.2±4.3	62.6±4.7 •	63.9±4.9 •	72.0±4.7	71.9±4.7
people	74.4±5.0	69.8±5.8 •	66.9±5.0 •	74.3±4.8	75.9±4.8

○, • statistically significant improvement or degradation vs 1-norm SVM

∗ Thioredoxin result obtained using the *SimpleLogistic* [13] implementation in WEKA, due to memory problems with *Logistic*.

Given the number of algorithms and datasets investigated, parameter tuning for all of the MI algorithms and base classifiers was infeasible. However, with the exception of the SVMs, the classification schemes used in the experiment had fairly robust parameter values already provided by the default settings in their WEKA implementations. We set the scaling parameter for MILES to $\sigma^2 = 8 \times 10^5$, as used by Chen et al. for *musk2*. As we found that the value for the 1-norm SVM regularization parameter selected by Chen et al. for *musk2* ($\lambda = 0.45$) produced poor results for MILES on many of the datasets, we performed internal cross-validation to select the best λ value for each fold. We evaluated six candidate values of λ via two-fold cross-validation on the training data for each fold of the ten repeats of ten-fold cross-validation, selecting the value which produced the highest classification accuracy. Finally, a greedy search was performed by iteratively evaluating adjacent candidates to the currently selected value via ten-fold cross-validation (again, on the training data of the fold). This internal cross-validation parameter search was performed using the GridSearch algorithm in WEKA. The candidate values were powers of ten between 10^{-1} and 10^{-6}. The same internal cross-validation method was also used to select the C regularization parameter for the 2-norm SVMs, with candidate values being powers of ten between 10^3 and 10^{-2}.

Table 2. MILES: Percentage Accuracy for Ensemble Base Learners

Dataset	1-Norm SVM	Adaboost + D. Stump	Random Forest	Adaboost + C4.5	Bagging + C4.5
musk1	83.3±11.8	88.0±11.6	87.0±11.4	85.8±12.0	86.0±11.5
musk2	91.6±8.3	83.2±11.5 •	81.7±11.2 •	83.2±11.3 •	83.7±11.5 •
eastwest	74.0±25.1	81.0±24.4	80.0±24.6	50.0±0.0 •	50.5±5.0 •
westeast	74.0±25.1	81.0±24.4	80.0±24.6	50.0±0.0 •	50.5±5.0 •
muta-atoms	74.8±14.4	83.9±8.6	82.0±8.2	79.5±8.5	80.5±7.7
muta-bonds	72.2±12.7	86.3±7.4 ○	79.7±10.5	80.1±9.9	77.4±8.9
muta-chains	75.9±9.2	86.0±8.0 ○	80.4±9.2	80.8±8.1	79.8±9.1
suramin	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2	62.0±46.1
thioredoxin	88.1±5.1	89.3±4.0	87.7±2.7	85.6±6.4	88.2±4.6
elephant	84.1±8.9	80.9±7.7	82.3±8.2	81.5±8.9	84.0±8.3
fox	63.0±9.5	61.6±10.9	64.9±10.2	59.4±11.6	61.4±10.3
tiger	80.7±8.3	80.5±8.9	78.6±9.0	75.4±9.3	75.7±8.4
bikes	78.4±4.2	78.0±5.0	79.2±4.4	78.0±4.5	77.7±5.1
cars	72.2±4.3	71.6±4.1	71.7±4.0	69.3±5.0	70.5±4.9
people	74.4±5.0	75.6±4.6	77.5±4.3	75.4±4.8	76.6±4.7

○, • statistically significant improvement or degradation vs 1-norm SVM

4.1 Comparison of Base Learners for MILES

A major goal of the experiment was to compare different base learners for MILES, particularly with respect to the 1-norm SVM. The results of this part of the experiment are displayed in Tables 1 and 2.

As the results show, the 1-norm SVM was competitive against the other base learners, with no other base learner consistently performing significantly better than it. However, Adaboost with decision stumps had two significant wins and only one significant loss versus the 1-norm SVM (Table 2).

The 2-norm SVMs with the linear and RBF kernels were both very competitive with the 1-norm method. The 2-norm SVM with the RBF kernel had one significant win and one significant loss versus the 1-norm SVM, while the 2-norm SVM with the linear kernel was only significantly worse than the 1-norm SVM on the *thieoredoxin* dataset (Table 1). These results indicate that the 1-norm SVM does not have a clear advantage over its 2-norm cousin as a base learner for MILES. In the observed experimental results, the increased sparsity of the 1-norm SVM did not translate into consistently superior classification accuracy, despite the high dimensionality of the datasets produced by the MILES transformation. However, the 1-norm SVM did outperform logistic regression, which produces linear models that do not exhibit any sparsity (Table 1).

The *eastwest* and *westeast* datasets were problematic for many MILES base learners, with half of the schemes performing little or no better than chance on these problems, although several schemes achieved accuracies of around 80%. Note that the results were similar or identical for both datasets, regardless of

the base learner. This is as expected, given that MILES is designed to use a symmetric MI assumption.

MILES' performance was consistent on the suramin problem. All base learning schemes achieved an accuracy of 65.0% on this dataset, except for bagging with C4.5, where an accuracy value of 62.0% was observed (Table 2). The small size of the dataset at least partially explains the consistency between schemes — it contains only 11 bags, albeit with many instances in each of those bags.

Random forests and Adaboost.M1 with decison stumps were the standout ensemble base learners for MILES (see Table 2). These classifiers only performed significantly worse than the 1-norm SVM on *musk2*. Furthermore, boosted decision stumps had the highest accuracy for any MILES base learner tried in the experiments on the *eastwest* / *westeast* datasets, all three *mutagenesis* datasets, *thieoredoxin*, and also matched the performance of the other base learners on *suramin*. Two of these results were significantly superior to the 1-norm SVM. Boosted decision stumps had slightly lower accuracies than the 1-norm SVM on five of the six image datasets, but these differences were not statistically significant. It should also be noted that parameter tuning with cross-validation was not necessary to achieve good results using Adaboost with decision stumps, unlike for the 1-norm SVM.

Although single C4.5 trees perform poorly (see Table 1), the results also show that boosted and bagged C4.5 trees perform well. However, boosted and bagged C4.5 performed no better than chance on *eastwest* and *westeast* and consequently suffered signficant losses against the 1-norm SVM on those datasets.

The strong performance of Adaboost.M1 with decision stumps is interesting, given the relationship between this model and the SVM model recommended by Chen et al (2006). Like support vector machines, the hypothesis learnt by Adaboost.M1 is a weighted linear threshold. When decision stumps are used, each weak learner corresponds to an attribute (i.e. the attribute that the decision stump splits on), and the weights for the weak learners perform a similar function to the attribute weights learnt by a linear SVM. As in SVMs, the solution is sparse because only a subset of the attributes is selected into the ensemble.

4.2 Comparison of MILES to Other Wrapper Algorithms

In this section we compare MILES to the two other wrapper algorithms — SimpleMI and MIWrapper — with respect to classification accuracy and training time, using Adaboost with decision stumps (100 stumps) as the base learner. Table 3 shows the classification accuracy and training time results for the algorithms.

The results show that in most cases all three propositionalization schemes give similar classification performance. There were no significant differences between MILES and SimpleMI for classification accuracy using this base learner. MILES was superior to MIWrapper on the *mutagenesis* datasets, but MIWrapper had significantly higher accuracy on the *people* dataset.

The results also show that there are substantial differences in training time. SimpleMI always had the shortest training time of the three methods for all

Table 3. Comparison of Wrapper Algorithms using Adaboost with Decision Stump Base Learner (100 Stumps)

	Percentage Accuracy			Training time (CPU Seconds)		
Dataset	MILES	SimpleMI	MIWrapper	MILES	SimpleMI	MIWrapper
musk1	88.0±11.6	83.2±12.3	84.7±10.7	4.3±0.3	3.0±0.1 ○	4.7±0.2 ●
musk2	83.2±11.5	78.7±11.9	79.7±10.6	284.7±41.5	3.5±0.0 ○	151.8±19.0 ○
eastwest	81.0±24.4	80.0±31.8	69.0±26.4	0.2±0.1	0.0±0.0 ○	0.2±0.0
westeast	81.0±24.4	81.5±31.5	69.0±26.4	0.2±0.1	0.0±0.0 ○	0.2±0.0
muta-atoms	83.9±8.6	80.3±8.4	66.5±2.3 ●	17.9±0.2	0.1±0.0 ○	0.7±0.0 ○
muta-bonds	86.3±7.4	85.8±7.7	73.2±8.4 ●	53.5±0.6	0.2±0.0 ○	3.1±0.1 ○
muta-chains	86.0±8.0	81.2±8.8	74.0±7.7 ●	88.1±0.8	0.2±0.0 ○	10.2±0.2 ○
suramin	65.0±45.2	53.0±48.1	65.0±45.2	3.7±0.3	0.0±0.0 ○	1.7±0.1 ○
thioredoxin	89.3±4.0	86.2±5.3	87.1±2.4	624.2±6.7	0.1±0.0 ○	32.2±0.4 ○
elephant	80.9±7.7	86.5±8.1	85.5±7.3	36.5±0.4	3.7±0.1 ○	15.8±0.2 ○
fox	61.6±10.9	67.0±10.5	65.7±9.6	34.0±0.4	3.3±0.9 ○	14.7±0.1 ○
tiger	80.5±8.9	82.5±8.7	81.8±8.5	30.4±0.4	1.8±0.3 ○	13.7±0.1 ○
bikes	78.0±5.0	80.3±4.9	79.2±4.6	451.2±1.3	12.3±0.2 ○	66.3±0.3 ○
cars	71.6±4.1	74.4±4.4	71.3±4.8	524.7±0.5	5.2±0.0 ○	73.4±0.3 ○
people	75.6±4.6	79.0±4.8	79.5±4.3 ○	385.1±0.4	4.5±0.0 ○	58.7±0.2 ○

○, ● statistically significant improvement or degradation vs MILES

datasets, almost always followed by MIWrapper, with MILES being the slowest of the wrapper algorithms on all datasets except *musk1*. This is unsurprising, given that the SimpleMI method only generates one instance for each training bag, without increasing the dimensionality of the feature space. Although MILES also generates one instance per training bag, the dimensionality of the feature space is almost always much higher, as the number of attributes is equal to the total number of instances in the training bags. In contrast, MIWrapper generates one instance for every instance in every bag, leaving the dimensionality of the feature space unchanged.

4.3 Overall Comparison of Classification Accuracy

The classification accuracy of the best variants of the three wrapper schemes MILES, MIWrapper, and SimpleMI, as well as the accuracy of the best of the other MI algorithms listed in Section 3 are shown in Table 4. Interestingly, the best results for each type of scheme were seldom more than a few percentage points different from each other. Notable exceptions to this are the *eastwest* / *westeast* datasets, where the best MILES classifier was around ten percentage points ahead of the best MIWrapper classifier and the best non-wrapper scheme, and SimpleMI was fourteen percentage points ahead of the best MILES scheme. On the *suramin* dataset, MIWrapper with the linear SVM base learner achieved an accuracy of 95%, which was 30-40 percentage points ahead of all other schemes. However, this result was not statistically superior to the majority of the other schemes, possibly due to the small size of the dataset (11

Table 4. The Best Result For Each Type of Scheme

Dataset	Best MILES	%	Best MIWrapper	%	Best SimpleMI	%	Best Other MI Learners	%
musk1	SMO (RBF)	89.1	Random Forest	87.3	SMO (RBF)	86.2	EMDD	85.2
musk2	1-Norm SVM	91.6	SMO (RBF)	83.0	SMO (RBF)	83.8	EMDD	84.7
eastwest	Adaboost + D.Stump	81.0	Adaboost + D. Stump	69.0	C4.5	95.0	Adaboost + Opt.Ball	71.5
westeast	Adaboost + D.Stump	81.0	Adaboost + D. Stump	69.0	C4.5	95.0	MISMO	70.0
muta-atom	Adaboost + D. Stump	83.9	Random Forest	81.9	Random Forest	80.9	MIBOOST + REPTree	77.8
muta-bond	Adaboost + D. Stump	86.3	Random Forest	83.1	Random* Forest	85.8	MIBOOST + REPTree	84.4
muta-chains	Adaboost + D. Stump	86.0	Bagging + C4.5	85.3	Random Forest	83.5	MIBOOST + REPTree	82.3
suramin	1-Norm* SVM	65.0	SMO (LIN)	95.0	SMO (LIN)	74.0	Citation* KNN	65.0
thioredoxin	Adaboost + D. Stump	89.3	Adaboost + C4.5	88.0	Logistic Regression	87.6	Adaboost + Opt.Ball	90.3
elephant	1-Norm SVM	84.1	Random Forest	87.1	Random Forest	87.3	MIBOOST + REPTree	82.8
fox	Random Forest	64.9	Adaboost* + D. Stump	65.7	Adaboost +D.Stump	67.0	MIBOOST + REPTree	66.3
tiger	SMO (RBF)	81.7	Random Forest	84.3	Random Forest	82.9	MIBOOST + REPTree	82.2
bikes	SMO (LIN)	80.1	SMO (RBF)	83.2	1-Norm SVM	84.3	mi-SVM	83.5
cars	1-Norm SVM	72.2	Random Forest	74.8	Random Forest	76.5	Adaboost + Opt.Ball	72.2
people	Random Forest	77.5	Random Forest	82.6	Random Forest	81.5	MIBOOST + REPTree	78.9

∗ Scheme was best-equal with one or more other schemes.

bags). There was also a difference of around eight percentage points between the best MILES classifier and the best SimpleMI and MIWrapper classifiers on the *musk2* dataset. Note that the σ value for MILES used in these experiments was selected by [6] based on tuning experiments on a subset of the *musk2* dataset, so the results for MILES on that dataset may be optimistic.

As mentioned previously, Adaboost with decision stumps was the dominant base learner for MILES, being the best (or best-equal) scheme for six of the fifteen datasets. MIBoost was the strongest overall method of the other MI algorithms, and the random forests algorithm was the best overall base learner for MIWrapper and SimpleMI.

5 Conclusions

The goals of the study were to compare base learners for MILES, and to compare MILES to other state-of-the-art MI algorithms. The results indicate that the 1-norm SVM is not generally superior to the standard 2-norm SVM as a base learner for MILES, despite the sparsity property that was thought to be important for the high-dimensional feature space created by the MILES transformation [6]. Moreover, although the 1-norm SVM was a competitive base learner for MILES in the experiment, Adaboost with decision stumps exhibited higher classification accuracy for some problem domains and did not require parameter tuning.

The results also show that when appropriate base learners are used, MILES is competitive in classification performance with any MI algorithm we considered. However, the simpler MIWrapper and SimpleMI methods almost always perform just as well as MILES, despite being significantly superior in terms of CPU training time. To achieve good classification accuracy in a wide variety of cases, random forests can be recommended as a base learner for MIWrapper and SimpleMI.

Perhaps the most interesting result of the experiments is the effectiveness of the extremely simple propositionalization methods SimpleMI and MIWrapper in comparison to MILES. The results also confirm their good performance when compared to more sophisticated dedicated MI algorithms (see also [8]). It appears to be an open problem to find MI algorithms that are superior to these simple propositionalization techniques on benchmark datasets, or to find problems where dedicated MI algorithms are more effective than propositionalization.

In future work, it would be interesting to compare MILES, SimpleMI and MIWrapper to other multi-instance propositionalization methods such as TLC [25], and the recent CCE [30] and BARTMIP [28] algorithms. In particular, BARTMIP is an algorithm that is closely related to MILES, where propositionalization is performed based on distances from *bags*, rather than distances from *points* as in the latter algorithm, so a thorough comparison of those two algorithms would be particularly insightful.

Acknowledgement

We would like to thank Michael Mayo for providing us with the multi-instance version of the GRAZ02 dataset.

References

1. S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, pages 577—584, 2002.
2. P. Auer and R. Ortner. A boosting approach to multiple instance learning. In *ECML*, pages 63–74, 2004.

3. P. S. Braddock, D. E. Hu, T. P. Fan, I.J. Stratford, A. L. Harris, and R. Bicknell. A structure-activity analysis of antagonism of the growth factor and angiogenic activity of basic fibroblast growth factor by suramin and related polyanions. *Br. J. Cancer*, 69(5):890–898, 1994.
4. L. Breiman. Bagging predictors. *ML*, 24(2):123–140, 1996.
5. L. Breiman. Random forests. *ML*, 45(1):5–32, 2001.
6. Y. Chen, J. Bi, and J. Z. Wang. MILES: Multiple-instance learning via embedded instance selection. *IEEE PAMI*, 28(12):1931–1947, 2006.
7. T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *AI*, 89(1-2):31–71, 1997.
8. L. Dong. A comparison of multi-instance learning algorithms. Master's thesis, University of Waikato, 2006.
9. J. Foulds. Learning instance weights in multi-instance learning. Master's thesis, University of Waikato, 2008.
10. E. Frank and X. Xu. Applying propositional learning algorithms to multi-instance data. Technical report, Dept. of Computer Science, University of Waikato, 2003.
11. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.
12. T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *ICML*, pages 179–186, 2002.
13. N. Landwehr, M. Hall, and E. Frank. Logistic model trees. In *ECML*, pages 241–252, 2003.
14. O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *NIPS*, 1997.
15. M. Mayo. Effective classifiers for detecting objects. In *CIRAS*, 2007.
16. D. Michie, S. Muggleton, D. Page, and A. Srinivasan. A new East-West challenge. Technical report, Oxford University Computing Laboratory, 1994.
17. C. Nadeau and Y. Bengio. Inference for the Generalization Error. *ML*, 52(3):239–281, 2003.
18. A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE PAMI*, 28(3):416–431, 2006.
19. J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.
20. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
21. P. Reutemann. Development of a propositionalization toolbox. Master's thesis, Albert Ludwigs University of Freiburg, 2004.
22. A. Srinivasan, S. Muggleton, R. D. King, and M. J. E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In *ILP*, pages 217–232, 1994.
23. C. Wang, S. D. Scott, J. Zhang, Q. Tao, D. Fomenko, and V. Gladyshev. A study in modeling low-conservation protein superfamilies. Technical report, Dept. of Comp. Sci., University of Nebraska-Lincoln, 2004.
24. J. Wang and J.-D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *ICML*, pages 1119–1125, 2000.
25. N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problems. In *ECML 2003*, pages 468–479, 2003.
26. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques (2nd Edition)*. Morgan Kaufmann, 2005.
27. X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *PAKDD*, pages 272–281, 2004.

28. M.-L. Zhang and Z.-H. Zhou. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, in press.
29. Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *NIPS*, pages 1073–1080, 2001.
30. Z.-H. Zhou and M.-L. Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *KAIS*, 11(2):155–170, 2007.