Beyond Trees: Adopting MITI to Learn Rules and Ensemble Classifiers for Multi-instance Data

Luke Bjerring and Eibe Frank

Department of Computer Science, University of Waikato
{lb54,eibe}@cs.waikato.ac.nz

Abstract. MITI is a simple and elegant decision tree learner designed for multi-instance classification problems, where examples for learning consist of bags of instances. MITI grows a tree in best-first manner by maintaining a priority queue containing the unexpanded nodes in the fringe of the tree. When the head node contains instances from positive examples only, it is made into a leaf, and any bag of data that is associated with this leaf is removed. In this paper we first revisit the basic algorithm and consider the effect of parameter settings on classification accuracy, using several benchmark datasets. We show that the chosen splitting criterion in particular can have a significant effect on accuracy. We identify a potential weakness of the algorithm—subtrees can contain structure that has been created using data that is subsequently removed—and show that a simple modification turns the algorithm into a rule learner that avoids this problem. This rule learner produces more compact classifiers with comparable accuracy on the benchmark datasets we consider. Finally, we present randomized algorithm variants that enable us to generate ensemble classifiers. We show that these can yield substantially improved classification accuracy.

1 Introduction

Multi-instance classification differs from standard propositional classification in that examples for learning consist of bags of instances. Potential application domains are drug activity prediction, where instances can be feature vectors describing different conformations of a molecule [5], and content-based image classification, where they are associated with different regions in an image [13]. In either case, a class label—indicating, e.g., whether a molecule is "active" or "inactive"—is available only for the entire example (i.e. bag), not the individual instances it contains, which renders this learning setting a challenging one.

In this paper we consider induction of decision trees and classification rules for multi-instance problems, and also consider ensemble learning. Decision tree induction is a popular learning method in standard propositional problems because of its computational efficiency and the interpretability of the output it generates. It can also yield highly competitive classification accuracy when used to learn ensembles. In this paper, we first revisit an existing decision tree induction method for multi-instance learning [2], called MITI, in Section 2, and evaluate its performance on a collection of benchmark datasets based on different configurations of the algorithm. Then, in Section 3, we show how we can apply a simple modification to this algorithm to yield a rule learner, which we call MIRI, that yields a compact set of classification rules for multi-instance problems. Finally, in Section 4, we show how accurate ensembles can be learned using randomisation, and summarise our main findings in Section 5.

2 The MITI Algorithm

The standard assumption in multi-instance learning—based on classification problems with two classes, positive and negative—is that a bag is positive if and only if it contains a positive instance, and negative otherwise [5, 12]. The key problem is that instance-level class labels are unknown for positive bags. All instances in negative bags must necessarily be true negative instances—otherwise the bag-level class label could not be negative. In contrast, it is possible that all but one of the instances in a positive bag are in fact false positives.

A common learning strategy under the standard assumption is to identify regions in instance space where positive bags overlap, i.e. regions of the instance space that contain positive instances from a non-trivial number of positive bags. This basic strategy was employed in the two oldest methods for multi-instance learning: in [5] a hyperrectangle is learned to describe the region where positive bags overlap and in [12] the maximum diverse density approach identifies parameters of a probabilistic model that is centered in such a region.

The multi-instance tree inducer (MITI) proposed by Blockeel *et al.* [2] is a learning algorithm based on the same standard assumption. It implements the top-down decision tree learning approach known from propositional tree inducers such as C4.5 [18], with two key modifications: (a) nodes are expanded in best-first order guided by a heuristic that aims to identify pure positive leaf nodes as quickly as possible, and (b) whenever a pure positive leaf node is created, all positive bags containing instances in this leaf node are deactivated.

A pure positive leaf node in this context is a node that only contains instances from positive bags. The assumption underlying this approach is that an instance's presence in a pure positive leaf is a strong indication that it is a true positive instance, and that all instances in the same bag that are not in the leaf should be eliminated from further consideration in the learning process because they are potentially false positives.

Pseudo code for MITI is shown in Algorithm 1. The algorithm is as originally presented in [2], with one small difference. On some of the datasets considered in our study, it can happen that the current node cannot be split any further because it contains identical instances from bags with different class labels. In that case, a leaf node is created based on the majority class.

Standard propositional tree induction normally proceeds in depth-first fashion, which can be implemented in a non-recursive fashion by storing unexpanded nodes in a last-in first-out (LIFO) queue. In MITI, this LIFO queue is replaced by a priority queue in which nodes are sorted in descending order according to

Algorithm 1 Pseudo code for MITI, based on [2].

```
let Q = root node
while Q is not empty do
    remove the first node N from Q
    if N is pure positive then
        make N a positive leaf and deactivate all bags with instances in N
    else if N is pure negative then
        make N a negative leaf
    else
        find the best split S for N
        if N cannot be split then
            make N a leaf with majority label and deactivate bags if necessary
        else
            split N according to S and add the child nodes of N to Q
        end if
    end if
    sort Q
end while
```

the proportion of positive instances they contain. When calculating this proportion, each instance is weighted by $1/|B|$, where $|B|$ is the size of the bag that contains the instance, to give each bag the same total weight, namely 1. Assuming w_p is the sum of weights of "positive" instances in the node concerned (note that this includes any potential false positives in the node), and w_n is the sum of weights of negative instances, the ratio $\frac{w_p}{(w_p+w_n+k)}$ is used to sort the nodes in the priority queue, where k is a parameter to the algorithm. This measure is called the *tozero(k)* estimate in [2].

Given numeric attributes, MITI applies binary splits to divide the data into two subsets at each internal node. Split selection is another important aspect of tree induction. [2] considers several measures to identify the best split at a particular node in the tree, but finds negligible differences for most of them. In the following, we consider two of the split selection criteria from [2]: *max-bepp*, which uses the maximum of the two estimated proportions of positives for the two subsets created by a split as the split quality score, and the standard *Gini index*, applied with the same estimate of the proportion of positives. Note that in the Prolog implementation of MITI kindly provided by the authors of [2], the Gini index is calculated without taking subset weights into account—the Gini-based impurity scores from each of the two subsets concerned are combined using a simple unweighted average. In this paper, we use branch weights in the standard fashion to combine the two subset scores for a split when calculating its Gini index.

2.1 Experimental Results

In [2], Blockeel *et al.* evaluate classification accuracy of MITI on synthetic data and two real-world multi-instance domains— the *musk* and *mutagenesis* problems—but splitting criteria are only compared on the synthetic data. In this section, we present a more extensive evaluation of MITI on benchmark data, including data from image classification problems, where we consider two splitting criteria (*max-bepp* and *Gini index*) and two values for the parameter k in the *tozero(k)* heuristic: 5, the default value from [2], and 0, which means that an unbiased estimate of the proportion of positives is applied. We also report tree size, which gives an indication of interpretability and is not considered in [2].

All experimental results presented in this paper are based on stratified 10-fold cross-validation, repeated 10 times, to yield 100 performance estimates for each dataset/algorithm combination. Tables show average accuracy as well as standard deviation across the 100 estimates. To test for statistical significance of individual differences, the corrected resampled paired t-test [16] is used, which is a conservative version of the standard paired t-test that is adjusted for dependency of estimates due to data reuse. This test is the standard test available in the Experimenter facility available in the WEKA workbench [10], which we used for the experiments. The significance level was left at the default value 0.05. Algorithms were implemented in Java and integrated into WEKA.

Table 1 shows estimated classification accuracy for the datasets included in our experiments. Table 2 shows tree size. The datasets used are those employed in [7].[1] These include mutagenicity prediction [19]—which was originally considered for multi-instance tree and rule learning in [21]—based on three different representations of molecules as bags of instances (*muta-atoms, muta-bonds, muta-chains*), the well-known trains problem from ILP [15] (*eastwest, westeast*), the two *musk* datasets [5], the *thioredoxin* protein identification task [20], and two groups of content-based image classification datasets (*elephant, fox, tiger* [1] and *bikes, cars, people* respectively, the latter group with Ohta-based features as in [14], derived from the GRAZ02 dataset [17]). We also included the synthetic *maron* problem [12, 8]. In this problem, instances are uniformly distributed in a 2D space and bags are classified as positive if they contain at least one instance that is located in a small area in the center of this space.

Considering classification accuracy, we can see that adjusting the value of the parameter k is important when using the special-purpose *max-bepp* split selection heuristic from [2] in MITI. Using the raw estimated proportion of positives (k=0) yields significantly more accurate classifiers on the *thioredoxin* problem, but applying a biased estimate of proportion (k=5) produces significantly higher accuracy on all but one of the image classification problems. In contrast, the results obtained using the Gini index appear less sensitive to the choice of k, but $k = 0$ yields higher estimated accuracy for all datasets apart from *musk1*, with two significant differences (not shown in the table)—on *cars* and *thioredoxin* respectively. The results for the *Gini index* with $k = 0$ are the best ones overall: this method dominates the *max-bepp* baseline in 12 out of 15 cases and yields statistically significant improvements in two cases. The results provide evidence that (a) biasing the estimate of proportion is generally detrimental when using the standard *Gini index* in MITI, and (b) the special-purpose *max-bepp* heuristic is generally inferior to the *Gini index.*

The results in Table 2 reinforce this message: trees grown using the *Gini index* with $k = 0$ are often substantially smaller than those generated using the other three variants. This also has a strong impact on runtime (not shown here) because the smaller trees can be grown more quickly.

[1] Excluding the suramin data, which contains missing values.

Dataset	MITI *max-bepp* *k*=5	MITI *Gini index* *k*=5	MITI *max-bepp* *k*=0	MITI *Gini index* *k*=0
eastwest	55.5±32.5	60.5± 34.3	67.0±32.7	62.5± 35.1
westeast	30.5±31.7	52.5± 34.4	47.0±26.4	58.5± 35.6
musk1	70.4±16.5	83.3± 12.7 ○	60.3±14.5	82.2± 12.7 ○
musk2	71.0±15.3	73.2± 13.1	62.7±14.5	74.4± 14.1
muta-atoms	80.2± 8.2	82.0± 8.3	80.5± 8.4	84.3± 7.9
muta-bonds	80.6± 7.9	81.2± 8.3	85.7± 8.8	81.9± 8.4
muta-chains	83.4± 7.7	84.4± 7.2	83.5± 8.5	87.2± 8.5
maron	50.0±10.1	55.6± 17.4	48.6±22.4	56.2± 22.9
elephant	77.6± 9.4	77.4± 9.3	72.0±10.5	77.9± 9.5
fox	61.7± 8.7	60.8± 9.6	49.7±11.0 ●	61.7± 10.4
tiger	74.7±10.0	70.3± 10.6	62.8±10.9 ●	74.0± 9.8
bikes	76.5± 5.2	74.6± 5.0	68.4± 5.0 ●	76.1± 5.1
cars	67.9± 4.3	63.7± 5.0	58.2± 5.8 ●	69.6± 4.9
people	73.4± 5.6	73.0± 4.8	66.1± 5.7 ●	74.8± 5.3
thioredoxin	35.7±11.0	62.7± 14.5 ○	80.0± 8.1 ○	82.1± 9.6 ○

○, ●: statistically significant compared to 2nd column

Table 1. Classification accuracy for different parameter settings in MITI.

Dataset	MITI *max-bepp* *k*=5	MITI *Gini index* *k*=5	MITI *max-bepp* *k*=0	MITI *Gini index* *k*=0
eastwest	25.8± 6.5	10.6± 4.3 ●	11.0± 1.9 ●	10.0± 3.2 ●
westeast	36.8± 5.3	23.4± 6.0 ●	14.8± 3.0 ●	12.8± 3.1 ●
musk1	20.1± 2.2	20.6± 2.2	50.4± 2.2 ○	22.8± 3.6 ○
musk2	43.1± 16.0	41.3±13.2	44.7± 2.2	33.1± 3.7
muta-atoms	261.7± 16.9	163.1±10.0 ●	62.7± 3.5 ●	62.5± 4.3 ●
muta-bonds	286.4± 30.5	157.8±11.4 ●	67.9± 4.2 ●	62.2± 6.0 ●
muta-chains	419.7± 44.5	198.1±11.4 ●	90.4± 7.5 ●	55.6± 7.5 ●
maron	902.5± 32.2	177.7±34.9 ●	30.5± 2.4 ●	17.5± 3.0 ●
elephant	46.6± 22.0	218.7±39.0 ○	102.0± 2.7 ○	32.3± 3.5
fox	166.5± 50.7	167.4±18.5	107.8± 3.3 ●	51.2± 4.8 ●
tiger	55.8± 16.8	160.2±20.2 ○	79.5± 5.0 ○	32.4± 4.1 ●
bikes	248.9± 83.1	634.7±96.3 ○	288.3±10.3	84.4± 4.2 ●
cars	219.4± 70.6	474.3±42.3 ○	387.4±15.6 ○	110.2± 5.5 ●
people	165.9± 45.4	662.7±61.5 ○	285.4±14.8 ○	90.2± 5.3 ●
thioredoxin	2202.2±207.5	250.4±43.7 ●	42.2± 3.3 ●	35.9± 5.7 ●

○, ●: statistically significant compared to 2nd column

Table 2. Tree size for different parameter settings in MITI.

The results on the *maron* data are particularly noteworthy because here the standard multi-instance assumption is known to hold by construction. Note that the *max-bepp* split selection criterion requires only one of the two subsets created by a split to exhibit high purity for it to be rated highly. The other subset can be poor and may thus need to be expanded into a large subtree—unless positive data in this subset can be successfully deactivated before this happens. In contrast, the *Gini index* combines impurity scores from both subsets in a weighted fashion.

3 MIRI: Using MITI to Learn Rule Sets

Whenever a positive leaf node is created, the MITI algorithm disables all instances of all bags that are associated with this leaf: any positive bag that has at least one positive instance in the leaf is disabled. The corresponding data is removed from all unexpanded nodes waiting in the priority queue and will thus not influence subsequent tree growth. However, tree structure that has already been created is left untouched. Conceptually, this is a potential drawback of the algorithm because data is removed from partially grown subtrees elsewhere in the overall tree structure. Splitting and node selection decisions that generated those existing incomplete subtrees should be revised to accommodate the new data distribution. At the very least, one would expect this to produce a more compact classifier because the amount of relevant training data is reduced.

Implementing this idea yields an algorithm whose output can be more naturally represented as a set of classification rules: when a positive leaf is encountered in the basic MITI algorithm, all positive bags associated with the leaf are removed from the training data, the path from the root node to this leaf node is turned into an if-then rule, and the algorithm is restarted on the remaining data. The tree structure is discarded and grown from scratch on the reduced data. We call this algorithm MIRI, for multi-instance rule induction.

Clearly, this approach will not generate any output that is due to potentially suboptimal split and node selection decisions based on outdated data because the entire tree structure is discarded after a positive leaf node has been turned into an if-then rule. When no positive leaf node can be created, the algorithm stops and appends a final default rule to the rule set that predicts the negative class. This will normally only happen when all positive data has been exhausted because the priority queue used in the best-first expansion method is ordered based on the proportion of positive data in each node located in the queue. Consequently it is appropriate to create a “catch-all” rule that simply predicts the negative class when the first negative leaf node is encountered.

There are pathological scenarios where positive data remains that is not covered by any positive rule, namely when there are identical instances that are located in both positive and negative bags. In that case it can happen that a node has to be turned into a leaf node even if it contains both positive and negative data. If the sum of weights for the negative instances in this node is greater than the sum of weights for the positive instances, then the node is turned into a negative leaf and the algorithm stops. On the other hand, if the positive data outweighs the negative data, the node is turned into a positive leaf and the associated positive bags are deactivated in the standard manner. This heuristic does not appear to cause problems on the benchmark datasets we consider.

The algorithm just described implements the standard separate-and-conquer rule learning strategy, where a rule is generated, the data covered by this rule is removed (i.e. separated out), and the remaining data is used to generate further rules. In contrast to most separate-and-conquer rule learners, a partial tree structure is induced to find the next rule to add to the rule set. In the context of propositional rule learning, where each example for learning consists of a single instance, this strategy is used in the rule learner PART [9], which generates a partial decision tree using the C4.5 tree learner [18].

It appears wasteful to generate a partial tree just to subsequently discard it. In practice, on the datasets we consider, MIRI’s runtimes are within an order of magnitude of MITI’s ones (which never requires more than a few seconds

	Classification accuracy		Classifier size		
Dataset	MITI	MIRI	MITI	MIRI	
eastwest	62.5±35.1	69.0±33.9	13.1± 4.7	7.9± 2.1	●
westeast	58.5±35.6	67.0±32.7	21.2± 6.0	11.6± 2.0	●
musk1	82.2±12.7	80.6±12.6	23.6± 3.8	21.8± 3.4	●
musk2	74.4±14.1	75.1±12.9	47.4± 7.9	36.4± 5.3	●
muta-atoms	84.3± 7.9	82.9± 8.3	241.7±27.9	95.2±11.2	●
muta-bonds	81.9± 8.4	81.6± 7.9	270.1±48.3	98.6±13.7	●
muta-chains	87.2± 8.5	83.3± 8.2	233.5±59.5	81.2±14.6	●
maron	56.2±22.9	59.6±25.1	54.3±16.7	18.7± 4.3	●
elephant	77.9± 9.5	78.7± 9.3	75.2±16.1	31.9± 3.5	●
fox	61.7±10.4	59.9±10.8	166.1±34.0	61.1± 6.1	●
tiger	74.0± 9.8	75.7± 9.9	66.4±17.3	31.4± 3.9	●
bikes	76.1± 5.1	76.5± 5.0	295.4±33.7	102.2± 7.0	●
cars	69.6± 4.9	67.9± 4.5	435.3±48.6	145.1±10.9	●
people	74.8± 5.3	73.5± 4.9	326.2±36.5	115.7± 8.2	●
thioredoxin	82.1± 9.6	82.9± 8.5	84.3±27.7	48.0±13.2	●

○, ●: statistically significant difference

Table 3. Accuracy and classifier size for MITI and MIRI ($k = 0$, *Gini index*).

to generate a tree when using $k = 0$ and the *Gini index*): MIRI is never more than five times slower. The best-first node expansion strategy is very effective in homing in on positive leaf nodes, which means that little additional tree structure is generated before a rule can be obtained. In the best case, only one path is created because only nodes leading to the relevant leaf node are expanded. Note also that many challenging multi-instance problems exhibit large bags of instances, which means that creation of a rule removes a substantial amount of instance-level data that will not need to be considered in subsequent iterations.

3.1 Experimental Results

Table 3 shows classification accuracy and classifier size for MITI and MIRI. Classifier size is measured by counting the number of tests in all positive rules included in the classifier. In MITI, positive rules correspond to leafs with a positive classification. In both cases, $k = 0$ was used (no bias in the estimated proportion of positives), and the *Gini index* was applied for split selection.

The results paint a clear picture: there is no statistically significant difference in classification accuracy between MITI and MIRI on the benchmark datasets we consider, but the classifiers learned by MIRI are significantly more compact in all cases. Hence, MIRI's ability to discard structure grown from outdated data does not have a significant impact on classification accuracy. Nevertheless, for data mining practitioners who are concerned with interpretability of the output, MIRI appears to provide a useful alternative to MITI.

4 Building Ensemble Classifiers

Although individual decision trees and rule sets can provide valuable insight into the structure underlying a dataset, and are thus an important tool for descriptive data mining, they are known to be inferior to ensemble classifiers in predictive tasks. A well-known strategy for generating an ensemble classifier is randomisation [6], in which the learning algorithm is randomised such that different classification models can be obtained from the same dataset, thereby yielding an ensemble. Predictions are then commonly obtained by voting.

In the propositional context, the random forest method [3] has proven particularly successful. Consequently we apply the basic strategy of this method to the multi-instance learning algorithms discussed above and evaluate whether similar gains in predictive accuracy can be obtained. In the random forest method, a decision tree learner is randomised by introducing non-determinism in the attribute selection step that is performed at each node. More specifically, rather than choosing the best split amongst all m available attributes, l attributes are selected at random first, where this randomly chosen subset can be different for each node, and then the best split amongst those l attributes is picked (where split quality is measured using a standard criterion such as the *Gini index*).

We can directly apply this method in MITI, and, consequently, also in MIRI. Large values of l decrease randomness and thus diversity, small values increase diversity but may yield ensemble members that are individually not very accurate. Both, accuracy of individual ensemble members and their diversity, will affect the accuracy of the final vote-based ensemble classifier.

4.1 Experimental Results

We generated empirical results using 100 ensemble members based on two values of l by applying WEKA's *RandomCommittee* method in conjunction with both MITI and MIRI as the base learner, yielding four configurations in total. Recent versions of WEKA allow parallel computation of ensemble members using *RandomCommittee* on multiple cores and this was exploited to obtain the results in a timely manner. The *Gini index* was used in MITI and MIRI and an unbiased estimate of positive proportion was applied ($k = 0$). The two values for l we consider are $l = 1$, which implies completely random attribute selection, and $l = \sqrt{m} + 1$, where m is the number of attributes in the dataset concerned, yielding a semi-random strategy. Results are provided in Table 4.

These results show that there is no noteworthy difference between MITI and MIRI ensembles in the case of semi-random attribute selection. However, when selecting attributes completely randomly, the MIRI-based ensemble performs worse. Thus selecting informative attributes appears more important when MIRI is used. Comparing semi-random selection with completely random selection, we can see that the latter strategy generally performs worse. The win/loss ratio is 10/4 in favour of semi-random selection in the case of MITI, although none of the differences are individually statistically significant. The semi-random selection method appears to have an edge on the datasets with a larger number of attributes (the image datasets and the *musk* problems) but on the datasets with a small number of attributes (*maron* and *mutagenesis*) there is no advantage. This makes sense intuitively: when there are many attributes, it is less likely that any one of them will be relevant to the classification.

Comparing these results to the ones in Tables 3 for individual trees and rule sets, we can see that the ensemble approach yields substantial improvements

Dataset	MITI Ensemble semi-r.	MITI Ensemble random	MIRI Ensemble semi-r.	MIRI Ensemble random
eastwest	72.5±31.3	72.5±27.9	75.5±32.2	72.5±29.6
westeast	37.5±32.9	31.5±33.1	48.5±34.4	30.5±33.3
musk1	86.5±11.5	80.7±12.6	85.0±11.7	78.6±12.3 ●
musk2	79.1±10.9	74.2±12.5	77.5±11.7	72.4±13.1
muta-atoms	85.3± 7.9	86.9± 7.7	85.4± 7.7	87.5± 7.1
muta-bonds	85.4± 7.5	86.3± 7.6	85.0± 7.5	85.7± 7.4
muta-chains	87.7± 8.6	87.1± 8.4	85.6± 8.4	86.4± 7.6
maron	56.2±22.9	66.4±22.7	59.6±25.1	60.6±21.5
elephant	88.5± 7.1	87.7± 6.8	86.9± 7.7	82.8± 8.5
fox	68.3± 8.8	61.4±10.7	66.3± 8.7	55.8±10.4 ●
tiger	82.8± 8.1	80.6± 8.4	83.2± 8.2	78.5± 9.7
bikes	84.1± 4.8	83.4± 4.6	84.9± 4.8	82.9± 4.8
cars	77.8± 4.1	76.3± 4.4	77.9± 4.1	74.6± 4.7 ●
people	81.9± 4.1	82.2± 4.0	82.4± 4.0	81.3± 4.1
thioredoxin	90.4± 6.1	89.4± 5.5	87.9± 7.0	86.9± 4.7

○, ●: statistically significant compared to 2nd column

Table 4. Classification accuracy for MITI and MIRI ensembles.

most cases. Thus it is clear that the success of randomisation in the propositional case translates into the realm of multi-instance problems.

It is also interesting to compare these results to those that can be obtained with other high-performance multi-instance classifiers on the same datasets. As an indicative baseline, we can draw on the results for various variants of the well-known MILES method for multi-instance learning [4] that are presented in [7], and the best results for two simple propositionalisation methods that can also be found in [7]. The estimated accuracies for the *muta-atoms, muta-chains, thioredoxin, elephant, fox, bikes* and *cars* datasets obtained from the semi-random MITI ensembles are greater than the best ones in [7], which were generated under exactly the same experimental conditions. The only real-world dataset where accuracy is noticeably below the best result in [7] is *musk2*.

On the *elphant, fox, tiger* and *musk* datasets, we can also compare to the results in [11], which are for the so-called *MIForest* method (Table 1 in [11]). This method generates a random forest ensemble for multi-instance learning using optimisation based on deterministic annealing. The estimated accuracy for our semi-random MITI ensemble is greater for four of the five datasets, indicating that our method is indeed competitive.

5 Conclusions

In this paper we have (a) presented a comparison of multi-instance decision trees learned by different MITI configurations on a collection of benchmark datasets, (b) shown how a simple modification enables us to learn rule sets rather than trees—yielding the MIRI algorithm—and (c) considered the effect of randomisation for ensemble learning using both MITI and MIRI.

Our results provide evidence that the standard *Gini index* is an appropriate splitting criterion for MITI, in particular if an unbiased estimate is used for the proportion of positives ($k = 0$): trees are generally more accurate and compact than those learned using the special-purpose *max-bepp* criterion. We have also shown that MIRI generates even more compact classifiers than MITI while maintaining comparable accuracy. Finally, we obtained highly competitive classification accuracy by applying randomisation to generate MITI and MIRI-based ensembles.

References

1. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS. pp. 561–568. MIT Press (2003)
2. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: ICML. pp. 57–64. ACM (2005)
3. Breiman, L.: Random forests. ML 45(1), 5–32 (2001)
4. Chen, Y., Bi, J., Wang, J.Z.: MILES: Multiple-instance learning via embedded instance selection. IEEE PAMI 28(12), 1931–1947 (2006)
5. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. AI 89(1–2), 31–71 (1997)
6. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. ML 40(2), 139–157 (2000)
7. Foulds, J., Frank, E.: Revisiting multiple-instance learning via embedded instance selection. In: AUS-AI. pp. 300–310. Springer (2008)
8. Foulds, J.R., Frank, E.: Speeding up and boosting diverse density learning. In: DS. pp. 102–116. Springer (2010)
9. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: ICML. pp. 144–151. Morgan Kaufmann (1998)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. 11(1), 10–18 (2009)
11. Leistner, C., Saffari, A., Bischof, H.: MIForests: multiple-instance learning with randomized trees. ECCV pp. 29–42 (2010)
12. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: NIPS. pp. 570–576. MIT Press (1998)
13. Maron, O., Ratan, A.L.: Multiple-instance learning for natural scene classification. In: ICML. pp. 341–349. Morgan Kaufmann (1998)
14. Mayo, M.: Effective classifiers for detecting objects. In: CIRAS (2007)
15. Michie, D., Muggleton, S., Page, D., Srinivasan, A.: To the international computing community: A new East-West challenge. Tech. rep., Oxford University (1994)
16. Nadeau, C., Bengio, Y.: Inference for the Generalization Error. ML 52(3), 239–281 (2003)
17. Opelt, A., Pinz, A., Fussenegger, M., Auer, P.: Generic object recognition with boosting. IEEE PAMI 28(3), 416–431 (2006)
18. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
19. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: ILP. pp. 217–232. GMD (1994)
20. Wang, C., Scott, S., Zhang, J., Tao, Q., Fomenko, D., Gladyshev, V.: A study in modeling low-conservation protein superfamilies. Tech. rep., Department of Comp. Sci., University of Nebraska-Lincoln (2004)
21. Zucker, J., Chevaleyre, Y.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem. In: Proc Conf of the Canadian Society for Computational Studies of Intelligence. pp. 204–214 (2001)