Propositionalisation of Multi-instance Data using Random Forests

Eibe Frank and Bernhard Pfahringer

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

Abstract. Multi-instance learning is a generalisation of attribute-value learning where examples for learning consist of labeled bags (i.e. multi-sets) of instances. This learning setting is more computationally challenging than attribute-value learning and a natural fit for important application areas of machine learning such as classification of molecules and image classification. One approach to solve multi-instance learning problems is to apply propositionalisation, where bags of data are converted into vectors of attribute-value pairs so that a standard propositional (i.e. attribute-value) learning algorithm can be applied. This approach is attractive because of the large number of propositional learning algorithms that have been developed and can thus be applied to the propositionalised data. In this paper, we empirically investigate a variant of an existing propositionalisation method called TLC. TLC uses a single decision tree to obtain propositionalised data. Our variant applies a random forest instead and is motivated by the potential increase in robustness that this may yield. We present results on synthetic and real-world data from the above two application domains showing that it indeed yields increased classification accuracy when applying boosting and support vector machines to classify the propositionalised data.

1 Introduction

Multi-instance learning is a generalisation of standard propositional learning–also called attribute-value learning–first introduced in [6]. Whereas propositional learning represents each example as one fixed-size vector of attribute-value pairs, multi-instance learning uses a bag of such vectors to represent examples and class labels are only associated with entire bags. The original learning assumption for multi-instance learning presented in [6] is applicable to two-class classification problems only and states that there is at least one vector in a positive bag that causes that bag to be a positive one. Negative bags are assumed to not contain any such "positive" vectors. Later work [18] has generalised this so-called multi-instance assumption to allow for arbitrary minimum and maximum counts of "positive" vectors as the necessary and sufficient condition for a positive bag.

Algorithms for multi-instance learning can be grouped into three classes. First there are dedicated new algorithms, with the most prominent one being Diversity Density [11]. Secondly, standard propositional learners like decision trees or

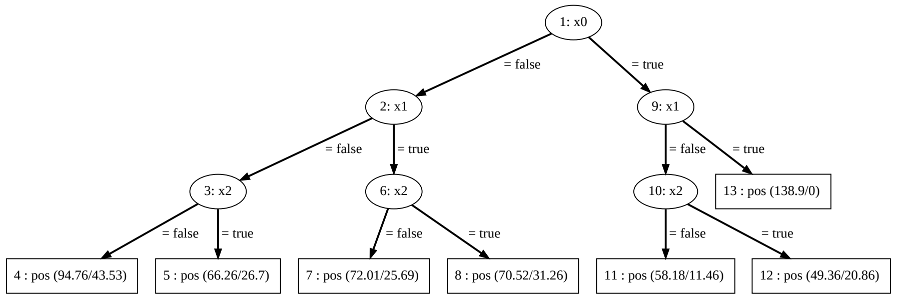

Fig. 1. Unpruned decision tree used for propositionalisation. In (x/y), x gives the total weight of all instances at the leaf node, and y gives the weight of all misclassified instances. Each leaf node is labeled *pos*, which means that for each leaf the sum of weights for positive instances is greater than the sum of weights for negative instances.

support vector machines, can be adapted–sometimes this is called upgraded–to deal with multi-instance data. Typical examples are MITI [3] and MISVM [1]. Thirdly, instead of adapting the algorithm, the data can be adapted, or propositionalised, to turn it into a standard propositional representation. As multi-instance learning is a special case of relational learning, any propositionalisation method from relational learning [9] could be applied, but generic relational learning methods often do not scale well. Therefore specialised multi-instance propositionalisation methods, inspired by general relational algorithms, have attracted some interest. One such method is PROPER [15], which is a specialisation of RelAggs [10]. On the other hand, the TLC algorithm [18] introduced a genuinely new way of propositionalisation, based on hierarchically partitioning the full instance space into sub-regions. The algorithm presented and analysed in this paper is a simple, yet effective extension of TLC.

The next section will describe this extension. Some implementation aspects are discussed in Section 3. Section 4 provides insights into the algorithm's behaviour by applying it to a synthetic multi-instance problem and Section 5 evaluates the method using a number of standard multi-instance benchmark datasets. Section 6 provides some pointers for future work and conclusions.

2 Propositionalisation using Random Forests

The multi-instance learning method presented in this paper is a simple extension of the Two-Level Classification (TLC) method as presented in [18]. TLC uses a heuristic approach to partition the instance space into regions. Once this has been done, a bag of instances (i.e. an example for learning in a multi-instance dataset) is propositionalised by counting how many instances of the bag fall into each region. These counts are attribute values in the propositionalised problem (i.e. there is one attribute for each region in the partition). Once each bag of

Bag		
x0	*x1*	*x2*
false	true	false
false	true	true
false	true	true
true	false	false
true	false	true
true	false	true
true	false	false
true	true	true
false	false	false

$\Rightarrow$

Instance												
r1	*r2*	*r9*	*r3*	*r6*	*r10*	*r13*	*r4*	*r5*	*r7*	*r8*	*r11*	*r12*
9	4	5	1	3	4	1	1	0	1	2	2	2

Fig. 2. Bag of instances and propositionalised form.

data has been propositionalised in this form, and each bag's classification has been attached to its propositionalised form, a standard single-instance learning algorithm for classification problems can be applied to the data.

The motivation for using this approach is that by dividing the instance space into regions and measuring occupancy, it becomes possible to describe the distribution of a bag's instances in instance space. This provides an alternative to simpler propositionalisation approaches that compute summary statistics such as the mean and standard deviation of the attribute values in a bag. In this manner, it is possible to preserve more information when propositionalising.

The question is how to define the regions in the instance space. TLC uses a standard single-instance decision tree to obtain a partition. To learn this tree, all instances from all bags are joined into a single dataset, discarding bag membership information, and labeled by their bag's class label. To make sure that large bags receive as much weight as small bags, each instance in this dataset is weighted by $\frac{1}{|X|} \times \frac{N}{b}$, where X is is the bag the instance comes from, N is the number of instances in the joined data, and b is the number of bags in the original dataset. In this manner, the sum of weights for the instances in the new dataset is N.

Let us consider an illustrative example, where we generated synthetic multi-instance data with three Boolean attributes $x0$, $x1$, and $x2$, for each bag. We generated 100 bags of instances, where each bag had between one and 10 instances, with equal probability for each bag size. When sampling instances, the joint probability distribution over the attributes was the uniform distribution, making all combinations of attribute values equally likely. The classification of each bag was determined as follows. A bag received the class label "positive" if it contained at least one instance for which both $x0$ and $x1$ had the value true. If it did not contain any instance with this property, the bag was labeled "negative". This relationship is an example of the classic (or "standard") assumption for multi-instance learning given in [6].

We then applied the above process to generate a partitioning, using an unpruned decision tree grown using information gain (based on the `REPTree` classifier with option `-P` in WEKA [8]). The resulting tree is shown in Figure 1. The leaf nodes in the tree show the majority class, which is "positive" in all cases, determined by examining the sum of weights of the instances in each class. This tree defines 13 regions, one for each node in the tree. Note that *all* nodes in the tree are used to define regions, not just the leaf nodes. In this example problem, (leaf) node 13 is the key region, because a bag is positive if and only if it has an instance in this region, but in general any node (or even combination of nodes) in the tree may need to be considered to determine class membership of a bag. Thus, occupancy counts for all regions are used as attribute values in the propositionalised data, so that the single-instance learner applied to the propositionalised data can identify the salient relationship.

Figure 2 shows an example bag and its propositionalised version, propositionalised using the tree in Figure 1. The class label has been omitted in this case, but the bag (and the resulting instance) are both positive because of membership in region 13. Note that the tree is explored in a breadth-first manner to generate the attribute values for the instance.

The hypothesis we investigate in this paper is that for large and messy real-world data a single tree may not be sufficient to obtain a robust learning method for multi-instance data. When considering standard classification problems, it is well-known that ensembles of trees such as random forests [5] outperform a single tree in terms of predictive performance. Here, we want to use an ensemble of trees for propositionalisation. The basic process is the same: the multi-instance dataset is converted into a single-instance dataset by attaching each instance's bag label to the instance and reweighting the instance, just as in TLC described above. Then, rather than learning a single tree, we learn an ensemble of trees using the random forest method, for example, using the `RandomForest` class in WEKA. If tree i in the ensemble of trees has l_i nodes (internal nodes + leaf nodes), then the ensemble defines $\sum_i l_i$ regions. To propositionalise a bag of instances, we then simply calculate the occupancy counts for all of these regions and create a feature vector of size $\sum_i l_i$. This vector is then labeled with the bag's label and can be processed using a single-instance learner.

It is clear that the resulting propositionalised instances have many more attributes than in the single-tree-based TLC method. The size of the feature vector is determined by the size of the ensemble. Our hypothesis is that larger ensembles, and thus feature vectors, will generally lead to improved classification accuracy when applying a learning algorithm to the propositionalised data.

3 Implementation in the WEKA Workbench

The WEKA machine learning workbench has support for multi-instance data in recent versions of the software, facilitated by the availability of relation-valued attributes: each bag of instances is stored as the value of a relation-valued attribute. For the experiments reported in this paper, a new `PartitionGenerator` interface has been added to WEKA that is implemented by several tree learners. It has a method that returns an array with counts that indicates, for a given instance, in which regions of the partition this instance is present. A tree learner may fill in this array by traversing a tree in a breadth-first fashion. In the case of a tree ensemble, the vectors for the individual trees are simply concatenated. WEKA now also has a `PartitionMembershipFilter` that can apply any `PartitionGenerator` to a given dataset to obtain these vectors for all instances. In conjunction with a new `MultiInstanceWrapper` filter in the `MultiInstanceFilters` package for WEKA 3.7, this filter can be applied to multi-instance data. When this is done, the vectors for all instances in a bag are simply added together by this filter to yield a vector of membership counts for a bag.

For added convenience, the `MultiInstanceLearning` package contains a new classifier implementation called `TLC` (for Two-Level Classification [18]) that applies the filtering process, using the above two filters, based on a particular `PartitionGenerator` specified as a parameter, and then runs a standard single-instance classifier on the propositionalised data. This single-instance classifier can also be specified as a parameter. It is thus straightforward to run systematic experiments with different partition generators and single-instance classifiers. For the experiments in this paper, WEKA's `RandomForest` class and `REPTree` decision tree learner were modified to implement the `PartitionGenerator` interface and the modified code is now part of the official WEKA code repository.

4 A Synthetic Problem

Our hypothesis is that propositionalisation using a random forest yields a more robust classifier than propositionalisation based on a single decision tree. In this section, we test this hypothesis empirically by introducing different levels of noise and data redundancy in a very simple synthetic learning problem. In this learning problem, there is a single numeric attribute that completely determines the classification of a bag: if this attribute has at least one positive value in the bag concerned, the bag is classified "positive"; otherwise, its class label is "negative". As we want to test the robustness of learning algorithms, we modify this deterministic relationship by first duplicating the attribute and its values for a particular bag to yield n copies and then taking a certain percentage of these copies for the particular bag concerned and replacing all their positive attribute values by their additive inverses (i.e. attribute value x becomes $-x$ if $x > 0$), *without* changing the class label of the bag. Hence, the modified data has n attributes per bag instead of one, where some of the attributes of positive bags may be corrupted and do not correctly indicate that the bag is positive. A learning algorithm must thus exploit the redundancy in these attributes to achieve maximum accuracy and cannot rely on a single attribute alone.

The exact set-up of the experiment is as follows. Based on a particular seed for the pseudo random number generator, we generate 100 bags containing between one and five instances each, where each bag size is given equal probability. For each bag, we first generate uncorrupted attribute values by sampling from the uniform distribution over the range $[-0.5, 0.5)$. If one of these attribute values is positive, the bag's class label is set to "positive", otherwise it is set to "negative".

Once the class label has been determined, n copies of the uncorrupted attribute values for a bag are generated to yield n attributes. Then, a biased coin is flipped n times, where this coin has probability p of coming up heads. If heads is the result of the coin flip, all positive attribute values in the corresponding copy of the attribute for the bag concerned (if any) are replaced by their additive inverse to introduce non-determinism in the relationship between this attribute's values and the bag's classification. In this way, all negative bags in the data have n identical attributes, but the n attributes of a positive may bag differ, depending on the value of the chosen probability p.

To measure accuracy of a learning algorithm on this data, we apply stratified 10-fold cross-validation to estimate the value of the kappa statistic, which can be viewed as a normalised version of classification accuracy that is particularly useful when the classes are unbalanced. Kappa is computed as follows:

$$\kappa = \frac{a - a_r}{1 - a_r},$$

where a is the estimated classification accuracy of the learning algorithm we want to evaluate, and a_r is the expected accuracy of a random classifier that assigns instances randomly to classes in such a manner that it assigns the same number of instances to each class as the learning algorithm we are evaluating. If kappa is greater than zero, the learning algorithm exhibits accuracy greater than what would be expected by assigning classifications randomly to the bags occuring in the test folds of the cross-validation. A value of one is the maximum that can be achieved.

We compare propositionalisation using unpruned decision trees grown using the information gain (based on `REPTree` with option `-P` in WEKA) to propositionalisation using a random forest of size 10 (based on `RandomForest` with option `-K 1` in WEKA). LogitBoost with decision stumps and 100 boosting iterations was used as the learning algorithm for the propositionalised data (`LogitBoost` with option `-I 100` in WEKA). Figures 3 and 4 show the results obtained for different numbers of attributes (i.e. values of n) and two different noise levels (i.e. values of p). Figure 3 shows results for the case where $p = 0.1$ and Figure 4 shows results for the case where $p = 0.3$. Each point in the plot corresponds to an average over 10 different runs of the experiments, where data was generated from scratch for each run using a different seed for the pseudo random number generator, followed by 10-fold cross-validation on this fresh data. The error bars correspond to 95%-level confidence intervals.

The graphs show that both propositionalisation methods benefit from redundancy in the data: accuracy increases as the number of attributes increases. However, at the higher noise level we consider ($p = 0.3$), accuracy levels out earlier when using a single, deterministically-grown decision tree. In contrast, propositionalisation using random forests benefits more from adding redundancy in the input by including more attributes: we can see that for $p = 0.3$, and more than four attributes ($n > 4$), the random-forest-based method achieves a level of accuracy that is statistically significantly higher than that obtained using a single tree—the 95%-level confidence intervals for the two methods do not overlap.

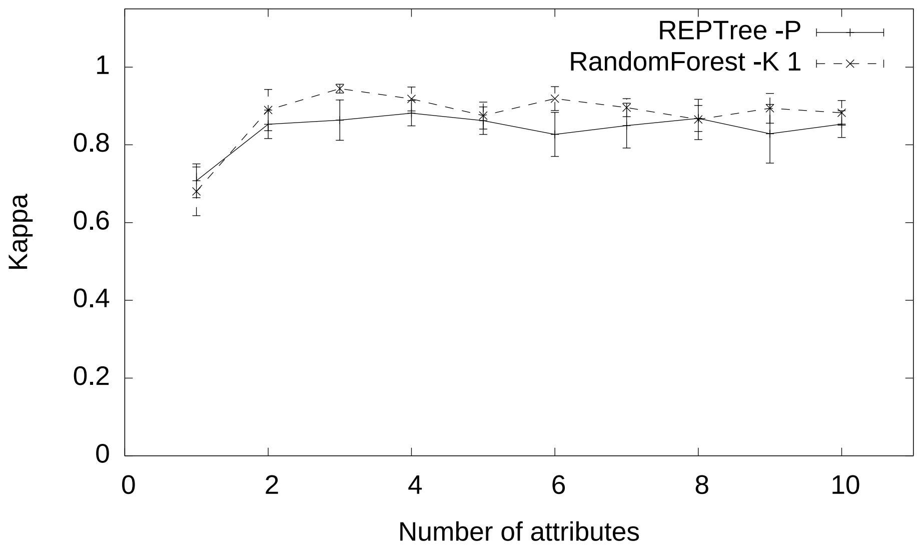

Fig. 3. Average kappa for $p = 0.1$.

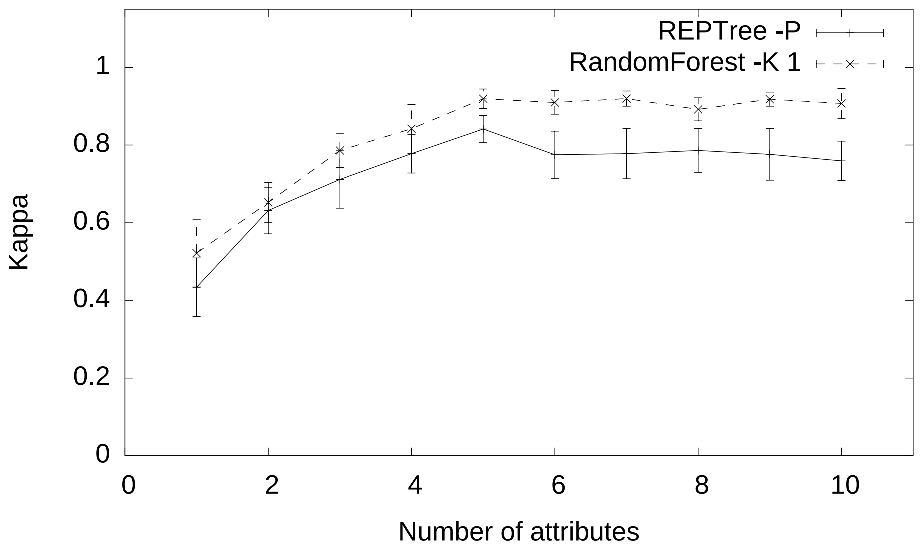

Fig. 4. Average kappa for $p = 0.3$.

Table 1. 10-times 10-fold cross-validated classification accuracy obtained using linear support vector machines.

Dataset	Decision Tree	Forest (10 trees)	Forest (50 trees)	Forest (100 trees)
musk1	84.4±13.2	87.7±10.7	89.1±10.6	89.4± 10.3
musk2	76.0±12.9	80.7±11.6	80.4±11.1	80.5± 10.9
mutagenesis3-atoms	86.5± 8.5	86.6± 8.1	86.7± 8.0	86.7± 8.1
mutagenesis3-bonds	86.1± 7.3	86.7± 7.5	86.7± 7.6	87.1± 7.5
mutagenesis3-chains	83.4± 8.9	84.5± 9.0	84.8± 9.6	85.3± 9.3
thioredoxin	88.4± 4.2	88.7± 4.6	89.3± 4.2	89.6± 4.2
suramin	65.0±45.2	61.0±46.9	57.0±47.7	55.0± 47.9
elephant	80.9± 9.8	86.0± 8.0	88.8± 7.4 ∘	88.9± 7.0 ∘
fox	63.0± 9.2	62.7± 9.1	65.0±10.0	66.2± 8.1
tiger	80.1± 9.1	81.1± 9.6	83.6± 8.2	84.3± 7.9
bikes	79.6± 4.6	81.6± 4.2	83.8± 4.3 ∘	84.3± 3.8 ∘
cars	67.8± 4.7	72.4± 4.4 ∘	75.4± 4.8 ∘	76.1± 4.4 ∘
people	78.4± 4.6	80.7± 4.5	82.5± 3.9 ∘	83.4± 3.9 ∘

∘ statistically significant improvement

5 Experiments on Real-world Data

To evaluate the performance of the random-forest-based propositionalisation approach, we performed experiments on benchmark multi-instance datasets that have previously been used in the literature. For propositionalisation, we used unpruned decision trees grown deterministically using information gain (`REPTree` with option `-P` in WEKA), and random forests with 10, 50, and 100 trees (`RandomForest` with options `-I 10`, `-I 50`, and `-I 100` in WEKA). Classification accuracy was estimated using 10-times 10-fold stratified cross-validation. Propositionalisation was performed separately based on each of the 100 training sets of the repeated cross-validation, so that the test data was never used in the propositionalisation process. We evaluated two learning algorithms in conjunction with the propositionalisation methods: linear support vector machines with $C = 1$ (`SMO` with option `-no-checks -N 2` in WEKA), using WEKA's `NonSparseToSparse` filter to create input data in sparse format, and boosted decision stumps, using 100 boosting iterations (`LogitBoost` in WEKA with option `-I 100`). The corrected resampled paired t-test [13] was used to establish statistical significance when considering observed differences in estimated accuracy, with a significance level of 0.05. All experiments were performed using the WEKA Experimenter interface [8].

The experimental results are shown in Tables 1 and 2. They include results for the two well-known *musk* datasets [6], where the task is to determine whether a molecule is active based on its geometric properties. Another task included is mutagenicity prediction [16], which was considered for multi-instance tree and rule learning in [19], based on three different representations of molecules as bags of instances *muta-atoms, muta-bonds* and *muta-chains*. We also include the *thioredoxin* protein identification task [17] and the *suramin* data, which is another drug activity prediction problem: identifying suramin [4] analogues that can act as anti-cancer agents. Image classification is another important application area of multi-instance learning methods. We include two sets of content-based im-

Table 2. 10-times 10-fold cross-validated classification accuracy obtained using LogitBoost with 100 boosting iterations.

Dataset	Decision Tree	Forest (10 trees)	Forest (50 trees)	Forest (100 trees)
musk1	84.6±12.9	86.2±12.0	88.4±10.4	88.4± 11.4
musk2	76.5±12.6	80.9±10.8	82.5±11.7	82.7± 12.2
mutagenesis3-atoms	85.5± 9.3	86.4± 8.7	86.0± 8.7	85.5± 8.3
mutagenesis3-bonds	86.7± 7.3	87.7± 7.1	87.6± 7.5	87.2± 7.0
mutagenesis3-chains	87.7± 8.1	86.5± 8.6	87.4± 7.6	87.8± 8.3
thioredoxin	90.2± 5.0	90.6± 4.7	91.2± 4.8	91.9± 4.2
suramin	49.5±47.9	56.5±48.0	51.5±48.4	56.0± 48.3
elephant	79.8± 9.4	83.7± 8.5	86.3± 7.5	86.4± 7.0
fox	63.0±10.0	62.4±11.2	65.9±10.9	64.4± 10.3
tiger	78.2± 8.8	81.4± 9.0	83.2± 8.5	83.5± 9.2
bikes	79.7± 4.5	81.2± 4.0	82.3± 4.7	83.2± 4.6 ◦
cars	70.1± 5.1	72.9± 3.9	74.8± 5.0 ◦	75.1± 4.5 ◦
people	77.3± 4.6	79.6± 4.1	80.4± 4.1	81.0± 3.8 ◦

◦ statistically significant improvement

age classification datasets. The first set consists of the *elephant, fox* and *tiger* [1] datasets and the second one contains the *bikes, cars* and *people* datasets. The latter set is based on Ohta-based features as in [12], and derived from the GRAZ02 dataset [14].

Table 1 shows that, when applied in conjunction with a linear support vector machine, using random forests for propositionalisation is preferable to using a single deterministic decision tree. This is particularly apparent in the case of the image classification datasets. Noteworthy improvements in predictive accuracy are obtained for all six image classification problems when using 100 trees in the random forests, and in four cases the improvement is statistically significant. The results also show that accuracy generally improves as more trees are included in the random forests. Again, bigger improvements are obtained in the image classification datasets. This is consistent with our hypothesis that random-forest-based classification can better exploit redundancy in the input data because the features in the image classification datasets are likely to be highly redundant.

Table 2 shows a similar picture when using 100 boosted decision stumps instead of linear support vector machines. Using random forests with 100 trees instead of a single unpruned decision tree yields higher estimated accuracy for 12 out of the 13 datasets. In the case of the three image classification datasets *bikes, cars*, and *people*, the improvement in accuracy is statistically significant.

Comparing the performance of boosted stumps and support vector machines when using propositionalisation based on random forests with 100 trees, we can see that support vector machines produce better accuracy on the image classification datasets, whereas there is no clear difference on the other datasets. Given that 100 boosted decision stumps can only test a maximum of 100 regions in the partitioned instance space, this indicates that high accuracy on the image classification datasets requires consultation of more than 100 regions to yield accurate classifications.

It is instructive to compare the accuracy obtained in the experiments presented here to that obtained in [2] (Table 4, semi-random ensemble), which

Table 3. Average training time in seconds in 10-times 10-fold cross-validated classification, obtained using linear support vector machines.

Dataset	Decision Tree	Forest (10 trees)	Forest (50 trees)	Forest (100 trees)
musk1	0.1±0.1	0.1± 0.1	0.4± 0.0 ○	1.0± 0.1 ○
musk2	0.9±0.1	2.2± 0.3 ○	13.7± 2.0 ○	36.3± 6.1 ○
mutagenesis3-atoms	0.1±0.0	0.8± 0.1 ○	9.5± 0.7 ○	62.9± 18.4 ○
mutagenesis3-bonds	0.1±0.0	2.2± 0.1 ○	23.3± 2.2 ○	127.7± 27.7 ○
mutagenesis3-chains	0.3±0.0	4.2± 0.2 ○	59.0±13.1 ○	263.3± 47.7 ○
thioredoxin	3.1±0.1	55.9± 2.9 ○	665.2±93.5 ○	1919.3±145.9 ○
suramin	0.0±0.0	0.2± 0.0 ○	0.8± 0.1 ○	1.8± 0.1 ○
elephant	0.3±0.0	0.8± 0.0 ○	7.5± 0.4 ○	41.4± 18.3 ○
fox	0.3±0.0	0.9± 0.1 ○	10.4± 0.6 ○	73.2± 27.3 ○
tiger	0.2±0.0	0.6± 0.0 ○	5.8± 0.3 ○	21.3± 2.8 ○
bikes	1.0±0.1	5.9± 0.2 ○	105.8±17.5 ○	400.0± 75.4 ○
cars	1.4±0.1	9.5± 0.4 ○	213.8±34.6 ○	729.8±133.4 ○
people	0.9±0.1	5.0± 0.2 ○	76.6±13.6 ○	317.9± 61.3 ○

○ statistically significant degradation

evaluated random forests of size 100 grown using a modified version of the MITI algorithm [3], a tree inducer designed for multi-instance learning. Exactly the same experimental protocol, based on the same 10-times 10-fold cross-validation runs, was applied in [2]. Propositionalisation using 100 trees, applied in conjunction with linear support vector machines (Table 1), produces higher estimated accuracy for eight of the twelve datasets considered both here and in [2], and lower accuracy for four datasets. Overall, accuracy obtained using propositionalisation appears very competitive.

Note that, computationally, support vector machines are well suited for the propositionalised data: the propositionalised bags yield very sparse feature vectors because most regions defined by a decision tree will not contain any instances of any particular bag. Hence, most attribute values in the propositionalised data will be zero, yielding sparse attribute vectors. Sparse vectors can be dealt with very efficiently in support vector machines because dot products of sparse vectors can be computed by iterating over the non-zero elements in the vectors only. WEKA supports data in sparse format, where only non-zero values in the instances are explicitly represented, and the `NonSparseToSparse` filter can be used to create this data.

Tables 3 and 4 show training times, including the propositionalisation process, averaged over the 10 runs of 10-fold cross-validation. It can be seen that using random forests to propositionalise the data significantly increases training time in all cases. One reason is that a tree ensemble needs to be grown, rather than a single tree. (Note that this process can be parallelised.) Another reason is that the instances in the propositionalised data have many more attributes when using random forests than when using a single deterministic tree because an ensemble of trees is used instead of a single tree and a single tree in a random forest is generally larger than a single deterministically grown tree, where attribute selection using information gain aims to minimise tree size. The results also show that applying a linear support vector machine is faster than applying boosting due to the fact that sparse data can be processed efficiently.

Table 4. Average training time in seconds in 10-times 10-fold cross-validated classification, obtained using LogitBoost with 100 boosting iterations.

Dataset	Decision Tree	Forest (10 trees)	Forest (50 trees)	Forest (100 trees)
musk1	0.2±0.1	0.7±0.1 ○	3.5± 0.2 ○	7.6± 0.5 ○
musk2	1.1±0.2	5.5±0.6 ○	30.9± 3.5 ○	70.6± 8.7 ○
mutagenesis3-atoms	0.6±0.0	9.7±0.4 ○	53.7± 1.9 ○	129.3± 11.1 ○
mutagenesis3-bonds	0.9±0.1	14.4±0.5 ○	82.9± 3.2 ○	215.2± 23.6 ○
mutagenesis3-chains	1.2±0.1	21.6±0.8 ○	134.7± 8.5 ○	427.2± 37.2 ○
thioredoxin	7.0±0.3	106.2±3.6 ○	946.0±59.2 ○	2641.3±126.3 ○
suramin	0.0±0.0	0.2±0.0 ○	1.1± 0.1 ○	2.5± 0.2 ○
elephant	0.6±0.1	9.2±0.3 ○	49.6± 1.3 ○	116.6± 9.4 ○
fox	0.7±0.1	11.0±0.4 ○	60.5± 1.3 ○	147.2± 11.3 ○
tiger	0.5±0.1	8.0±0.4 ○	42.6± 1.3 ○	95.7± 7.0 ○
bikes	7.1±0.2	90.5±1.4 ○	598.0±12.0 ○	1545.8± 38.2 ○
cars	10.7±0.3	136.8±1.7 ○	952.5±23.0 ○	2381.3±131.4 ○
people	6.0±0.2	76.2±1.1 ○	496.3± 7.7 ○	1208.0± 49.9 ○

○ statistically significant degradation

6 Conclusions

Multi-instance learning is an interesting and useful generalisation of propositional learning. This paper has presented a simple, yet effective extension of the TLC propositionalisation method that grows random forests for propositionalising multi-instance data. The new method's increased robustness with regard to noise in the input was demonstrated with a synthetic example, and a comprehensive evaluation on benchmark datasets representing image and molecule classification problems also shows improved accuracy for the new method, albeit at the cost of a considerable increase in runtime.

The standard random forest method applied in this paper chooses split points on numeric attributes deterministically when considering these attributes for splitting. However, as the instances' class labels used in this process are simply taken to be their bags' labels, they may be incorrect. Hence, it would be interesting to apply a method that chooses split points randomly. This is what the Extra-Trees algorithm [7] for growing a tree ensemble does. Applying it to propositionalisation of multi-instance data is a promising avenue for future research.

References

1. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Proc Conf on Neural Information Processing Systems. pp. 561–568. MIT Press (2003)
2. Bjerring, L., Frank, E.: Beyond trees: Adopting miti to learn rules and ensemble classifiers for multi-instance data. In: Proc 14th Australasian Conf on Artificial Intelligence. pp. 41–50. Springer (2011)
3. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: Proc 22nd Int Conf on Machine Learning. pp. 57–64. ACM (2005)

4. Braddock, P.S., Hu, D.E., Fan, T.P., Stratford, I., Harris, A.L., Bicknell, R.: 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)
5. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
6. 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)
7. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. 63(1), 3–42 (2006)
8. 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)
9. Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining. In: Relational Data Mining, pp. 262–286. Springer (2000)
10. Krogel, M.A., Rawles, S., Zelezný, F., Flach, P.A., Lavrac, N., Wrobel, S.: Comparative evaluation of approaches to propositionalization. In: Proc 13th Int Conf on Inductive Logic Programming. pp. 197–214. Springer (2003)
11. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: Proc Conf on Neural Information Processing Systems. pp. 570–576. MIT Press (1998)
12. Mayo, M.: Effective classifiers for detecting objects. In: Proc 4th Int Conf on Computational Intelligence, Robotics, and Autonomous Systems (2007)
13. Nadeau, C., Bengio, Y.: Inference for the Generalization Error. Machine Learning 52(3), 239–281 (2003)
14. Opelt, A., Pinz, A., Fussenegger, M., Auer, P.: Generic object recognition with boosting. IEEE Transaction on Pattern Analysis and Machine Intelligence 28(3), 416–431 (2006)
15. Reutemann, P., Pfahringer, B., Frank, E.: A toolbox for learning from relational data with propositional and multi-instance learners. In: Proc 17th Australian Conf on Artificial Intelligence. pp. 1017–1023. Springer (2004)
16. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: Proc 4th Int Workshop on Inductive Logic Programming. pp. 217–232. GMD (1994)
17. 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)
18. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Proc 14th European Conf on Machine Learning. pp. 468–479. Springer (2003)
19. 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)