



Multiple Instance Learning with Bag-Level Randomized Trees

Tomáš Komárek¹(✉) and Petr Somol^{2,3}

¹ Faculty of Electrical Engineering, Czech Technical University in Prague,
Technická 2, Prague 6, Czech Republic

`komartom@fel.cvut.cz`

² Institute of Information Theory and Automation, Czech Academy of Sciences,
Pod Vodárenskou věží 4, Prague 8, Czech Republic

`somol@utia.cas.cz`

³ Faculty of Management, University of Economics, Prague, Czech Republic

Abstract. Knowledge discovery in databases with a flexible structure poses a great challenge to machine learning community. Multiple Instance Learning (MIL) aims at learning from samples (called bags) represented by multiple feature vectors (called instances) as opposed to single feature vectors characteristic for the traditional data representation. This relaxation turns out to be useful in formulating many machine learning problems including classification of molecules, cancer detection from tissue images or identification of malicious network communications. However, despite the recent progress in this area, the current set of MIL tools still seems to be very application specific and/or burdened with many tuning parameters or processing steps. In this paper, we propose a simple, yet effective tree-based algorithm for solving MIL classification problems. Empirical evaluation against 28 classifiers on 29 publicly available benchmark datasets shows a high level performance of the proposed solution even with its default parameter settings. Data related to this paper are available at: <https://github.com/komartom/MIDatasets.jl>. Code related to this paper is available at: <https://github.com/komartom/BLRT.jl>.

Keywords: Multiple Instance Learning · Randomized trees
Classification

1 Introduction

Multiple Instance Learning (MIL) relaxes conditions for data representation. In MIL formalism, objects of interest are described by sets $\mathcal{B}_1, \mathcal{B}_2, \dots$ (called bags) consisting of multiple feature vectors of an identical size $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$, $\mathbf{x} \in \mathcal{X}$ (called instances). Each bag is attributed output variable y (e.g. label $y \in \{0, 1\}$ in a binary classification problem). The goal is to infer function \mathcal{F} from training dataset $\mathcal{D} = \{(\mathcal{B}, y)_1, (\mathcal{B}, y)_2, \dots\}$ that can predict output variables for previously unseen bags $\mathcal{F}(\mathcal{B}) = y$.

This relaxed formalism has received a lot of attention since its first formulation [11]. One of the recent application domains, where MIL formalism seems to fit better than traditional Single Instance Learning (SIL), is network security [20]. For example, in the work [16] dealing with a binary classification of network users as either infected or clean based on monitoring their web requests, the MIL approach enabled to (i) *describe the problem more naturally* – each user is represented by a bag with instances corresponding to individual requests; representing users with single feature vectors would be too restrictive, since the number of requests can vary from user to user, (ii) *save ground truth acquisition costs* – security analysts preparing a training dataset did not have to label individual requests as infection-related or benign; it was enough to provide labels on the (higher) user/bag level, resulting in a significantly lower number of items that needed to be annotated, (iii) *improve classification performance* – a MIL classifier modeling the global contextual information across multiple requests was able to achieve a higher classification precision than a SIL classifier analyzing individual requests one by one.

Although many MIL classifiers have been developed over the years, most of them work well only within their specific application domains and are less effective over a wider range of problems [8] (demonstrated in Sect. 4). A recent survey of MIL classifiers [1] has pointed out that approaches extracting global *bag-level* information achieve a higher performance in general than their counterparts operating on *instance-level* trying to infer instance labels from the bag ones (discussed in Sect. 2). Following this observation, we propose a novel bag-level MIL algorithm (Sect. 3) as an extension of traditional single-instance randomized trees [15] to the multiple-instance setting. Ensembles of tree-based learners (e.g. Extremely randomized trees [15] or Breiman’s Random Forests [5]) are known to perform very well across many domains even without a careful hyperparameter tuning [12], which makes a good case for research of a general MIL tree-based model formalism.

2 Related Work

A taxonomy proposed in the review [1] categorizes MIL classifiers into two groups according to the level at which they extract the discriminative information. The first *instance-level* category considers the discriminative information to lie at the level of individual instances. It is assumed that each instance \mathbf{x} can be attributed binary label $y_{\mathbf{x}} \in \{0, 1\}$ and that positive bags contain at least one positive instance $y_{\mathbf{x}} = 1$. Negative bags are supposed to not contain any positive instance. The instance labels are unknown (even in the training dataset), but are inferred during the learning process. Most algorithms based on this assumption build instance-level classifier $f(\mathbf{x}) = y_{\mathbf{x}}$ and produce bag labels by a simple aggregation of instance level scores $\mathcal{F}(\mathcal{B}) = \max_{\mathbf{x} \in \mathcal{B}} f(\mathbf{x})$. This category is mostly represented by earlier works, starting with the APR algorithm [11] proposed in 1997. APR (Axis-Parallel Rectangle) algorithm considers all instances in positive bags to be positive and expands/shrinks an hyper-rectangle in the

feature space to maximize the number of positive instances falling inside, while minimizing the number of negative ones. MI-SVM [2] initially also assumes that all instances in positive bags are positive, but then maximizes a margin between the positive and negative bags by selecting a single instance to represent each bag. EM-DD [25] uses Expectation-Maximization (EM) algorithm to search for the maximum of diverse density (DD) measure. A location in the feature space has a high DD measure if the number of nearest instances from different positive bags is high and from negative bags low. Other approaches are based e.g. on boosting (MILBoost [23]) or deterministic annealing [14, 17].

The second *bag-level* category (mostly represented by later works) considers the discriminative information to lie at the level of bags. Representatives of this category do not use the notion of instance labels, which does not reflect the reality in many applications [1], but rather treat bags as whole entities. That is, a bag-level classifier $\mathcal{F}(\mathcal{B})$ extracts the information from the whole bag to make a decision about the class of \mathcal{B} , instead of aggregating individual instance-level decisions. Bag-level methods are further categorized as either *bag-space* or *embedded-space*. Since bags are non-vectorial objects, the *bag-space* methods define a distance function (or kernel) that can compare any two bags, e.g. \mathcal{B}_i and \mathcal{B}_j , and plug this function into a distance-based classifier such as k-NN or SVM. Examples of such functions are the minimal Hausdorff distance $d(\mathcal{B}_i, \mathcal{B}_j) = \operatorname{argmin}_{\mathbf{x}^i \in \mathcal{B}_i, \mathbf{x}^j \in \mathcal{B}_j} \|\mathbf{x}^i - \mathbf{x}^j\|$ measuring distance between the closest instances or the Earth Movers Distance (EMD) $d(\mathcal{B}_i, \mathcal{B}_j) = \frac{\sum_i \sum_j w_{ij} \|\mathbf{x}^i - \mathbf{x}^j\|}{\sum_i \sum_j w_{ij}}$, where weights w_{ij} are obtained through an optimization process that globally minimizes the distance subject to some constraints, see [24] for details. On the other hand, the *embedded-space* methods define a transformation mapping bags into single fixed-size vectors, which effectively converts the MIL problem into the standard SIL one. In the work of MI-Kernel [13], each bag is transformed by calculating simple statistics like the maximum, minimum or mean across all feature dimensions and concatenating the results into a single vector. MILES [7] maps each bag into a feature space defined by similarities to instances in the training bags. A sparse 1-norm SVM is then applied to select only the discriminative features (i.e. instances) and construct the classifier. Bag dissimilarity [9], on the other hand, measures similarities to the training bags rather than instances.

Most prior tree-based works fall into the *instance-level* category. MIForest [17] uses the deterministic annealing approach to uncover the instance labels during the tree growing. MITI [4] introduces a weighted Gini impurity measure and modifies the tree induction procedure to prioritize expansion of nodes with positive instances. MIOForest [21] extends MITI by implementing non-linear splitting rules instead of the traditional axis-orthogonal ones and by optimal combining of individual tree outputs within the forest. To our knowledge, the only tree-based MIL classifier that operates on the level of bags is RELIC [19]. Since we consider RELIC as the closest prior work, we discuss the differences with respect to the proposed solution in detail in Sect. 3.

3 Bag-Level Randomized Trees

The bag-level randomized trees (BLRT) are trained according to the classical top-down greedy procedure for building ensembles of unpruned decision trees. Individual tree learners recursively partition a training dataset by choosing binary splitting rules until pure sample sets are obtained.

The key difference, however, lies in the conditions that are evaluated inside the splitting nodes. While nodes of standard single-instance decision trees (Eq. 1) test only whether feature f of a given sample is greater than certain value v , nodes of the proposed MIL trees also count the number of instances within the sample (i.e. bag) that accomplish the condition. This absolute count is then normalized by bag size $|\mathcal{B}|$ and compared to value $r \in [0, 1)$ (Eq. 2).

$$\mathcal{N}_{\text{SIL}}(\mathbf{x}; f, v) = \begin{cases} \text{left,} & \text{if } x_f > v, \\ \text{right,} & \text{otherwise.} \end{cases} \tag{1}$$

$$\mathcal{N}_{\text{MIL}}(\underbrace{\mathcal{B}; f, v, r}_{\Phi}) = \begin{cases} \text{left,} & \text{if } \left[\frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \mathbb{1}[x_f > v] \right] > r, \\ \text{right,} & \text{otherwise.} \end{cases} \tag{2}$$

Parameter r denotes a relative count of instances \mathbf{x} inside bag \mathcal{B} that must satisfy the inner condition $x_f > v$ to be the whole bag passed to the left branch. It is the only additional parameter that needs to be learned from the training data together with f and v . Symbol $\mathbb{1}$ stands for an indicator function that equals one if its argument is true and zero otherwise.

Note that if bags are of size one, nodes \mathcal{N}_{MIL} behave like the traditional \mathcal{N}_{SIL} regardless the value of r parameters. The next special case is when the relative count takes extreme values, i.e. $r \in \{0, 0.9\}$ ¹. The proposed algorithm then becomes equivalent to the prior art solution known as RELIC [19]. Under this condition, the splitting rules act as either the universal or the existential quantifier. In particular, bags are tested in two possible ways: if there exists at least one instance that fulfills the inner condition or if the condition is satisfied by all instances. An experiment in Sect. 4 (Fig. 2), however, shows that the ability of the proposed algorithm to test situations also between these two extreme cases is highly beneficial on many datasets.

Search of optimal splitting parameters $\Phi^* = (f, v, r)$ during the tree growth is implemented in a randomized manner. At each node construction, a set of candidate splitting rules $\mathcal{R} = \{\Phi_1, \dots\}$ is generated (based on local training subset $\mathcal{S} \subseteq \mathcal{D}$) among which the best one Φ^* is selected according to a score obtained by an impurity measure such as Information gain [18] or Gini impurity [6]. Specifically, for each feature f out of K randomly selected ones, T values of parameter v are drawn uniformly from interval $[x_f^{\min}, x_f^{\max})$, where x_f^{\min} and x_f^{\max} denote

¹ Technically, the value of 0.9 should be 1 minus the smallest representable value.

the minimum and the maximum value of feature f across all bags within the local sample set. For each such pair (f, v) , other T values of parameter r are generated uniformly from interval $[0, 1)$. In total, there are $K \times T \times T$ candidate splitting rules at maximum². A detail description of the tree induction procedure is given in Algorithm 1 in the form of pseudo code.

The above randomized approach is adopted from Extremely³ randomized trees [15] and generalized to MIL setting by adding the third parameter r (i.e. the relative count). Unlike CART algorithm, used e.g. in Breiman’s Random Forests [5], the randomized search does not require to go over all possible splitting points on selected features, which could be prohibitively expensive in this MIL variant of trees. Furthermore, the explicit randomization in combination with ensemble averaging makes the decision boundary more smooth, resulting in models with better or equal accuracy than that of Random Forests [15].

Algorithm 1 builds M fully grown decision trees. Each tree is trained on the whole sample set rather than a bootstrap replica as realized e.g. in Random Forests. Training on the full original sample set minimizes bias. Variance is reduced by the strong randomization in the splitting parameters combined with the output aggregation across multiple trees. From the computational point of view, the time complexity of the learning procedure is, assuming balanced trees, $\Theta(MKT^2N_I \log N_B)$, where N_B and N_I denote the number of bags and the number of instances within the bags, respectively⁴.

In the testing mode, assuming a binary classification problem (i.e. $y \in \{0, 1\}$), predictions of individual trees are aggregated by a simple arithmetic average to produce final prediction score $\hat{y} \in [0, 1]$.

4 Experiments

The proposed algorithm is evaluated on 29 real-life datasets that are publicly available e.g. on <https://doi.org/10.6084/m9.figshare.6633983.v1>. The datasets with meta descriptions are listed in Table 1. These classification problems are well known and cover a wide range of conditions in terms of application domains (molecule, scene, image, text, audio spectrogram, etc.), ratios of positive and negative samples (e.g. imbalanced Corel datasets), feature dimensions (from 9 to 6519) and average numbers of bag instances (from 4 to 185). For more details about the datasets we refer the reader to a recent study of MIL datasets [8].

The same collection of datasets was also used in the evaluation of 28 MIL classifiers (including their variants) implemented in the MIL matlab toolbox [22]. The last two columns of Table 1 summarize the results from the evaluation available also through <http://homepage.tudelft.nl/n9d04/milweb/>. We report only those classifiers that achieved the highest performance by means of AUC

² If x_f^{\min} equals to x_f^{\max} , no splitting rules are generated on feature f .

³ Term *extremely* corresponds to setting $T = 1$.

⁴ When bags are of size one (i.e. $N_I = N_B = N$) and $T = 1$, the complexity is equivalent to the complexity of Extremely randomized trees $\Theta(MKN \log N)$.

Algorithm 1. Induction algorithm for bag-level randomized trees (binary classification problem $y \in \{0, 1\}$ and numerical features are assumed).

Function Train($\mathcal{D}; M, K, T$)

Input : A training set $\mathcal{D} = \{(\mathcal{B}, y)_1, \dots\}$,
 a number of trees to grow M ,
 a number of randomly selected features K ,
 a number of generated thresholds T .
Output: An ensemble of bag-level randomized trees \mathcal{E} .
 $\mathcal{E} = \emptyset$
foreach tree in $1 \dots M$ **do**
 | $\mathcal{E} = \mathcal{E} \cup \{\text{BuildTree}(\mathcal{D}; K, T)\}$
return \mathcal{E}

Function BuildTree($\mathcal{S}; K, T$)

Input : A local training subset $\mathcal{S} \subseteq \mathcal{D}$.
Output: A node with left and right followers or a leaf.
if all y in \mathcal{S} are equal **then**
 | **return** leaf(y)
 $\mathcal{R} = \text{GenerateCandidateSplittingRules}(\mathcal{S}; K, T)$
 $\Phi^* = \text{argmax}_{\Phi \in \mathcal{R}} \text{Score}(\mathcal{S}, \Phi)$
 $\mathcal{S}_{\text{left}} = \{\mathcal{B} \in \mathcal{S} \mid \mathcal{N}_{\text{MIL}}(\mathcal{B}; \Phi^*) = \text{left}\}$
 $\mathcal{S}_{\text{right}} = \mathcal{S} \setminus \mathcal{S}_{\text{left}}$
if $\mathcal{S}_{\text{left}} = \emptyset$ or $\mathcal{S}_{\text{right}} = \emptyset$ **then**
 | **return** leaf($\frac{1}{|\mathcal{S}|} \sum_{y \in \mathcal{S}} y$)
return node($\mathcal{N}_{\text{MIL}}(\cdot; \Phi^*)$, BuildTree($\mathcal{S}_{\text{left}}$), BuildTree($\mathcal{S}_{\text{right}}$))

Function GenerateCandidateSplittingRules($\mathcal{S}; K, T$)

Output: A set of candidate splitting rules $\mathcal{R} = \{\Phi_1, \dots\}$.
 $\mathcal{R} = \emptyset$
foreach feature f in K randomly selected ones (without replacement) **do**
 | Find extremes x_f^{\min} and x_f^{\max} on given feature f across all bags $\mathcal{B} \in \mathcal{S}$
 | **if** $x_f^{\min} \neq x_f^{\max}$ **then**
 | | **foreach** value v in T uniformly drawn values from $[x_f^{\min}, x_f^{\max}]$ **do**
 | | | **foreach** value r in T uniformly drawn values from $[0, 1]$ **do**
 | | | | $\mathcal{R} = \mathcal{R} \cup \{\Phi\}$, $\Phi = (f, v, r)$
return \mathcal{R}

metric⁵ at least on one problem. This selection yields to 13 classifiers that are listed in Table 2 together with references to their original papers.

⁵ Area Under a ROC Curve showing the true positive rate as a function of the false positive rate. AUC is agnostic to class imbalance and classifier's threshold setting.

Table 1. Metadata about 29 used datasets together with classification scores and standard deviations presented in percent ($AUC \times 100$). Best results are in bold face. Stars denote statistically significant ($\alpha = 0.05$) differences according to Welch’s t-test.

| Dataset | | | | BLRT | Best prior art | |
|------------------|----------|-------|-----------|---------------------|---------------------|-----------|
| Name | Bags +/- | Feat. | Avg.inst. | AUC | AUC | Algorithm |
| Musk1 | 47/45 | 166 | 5 | 96.8 (1.6) * | 92.9 (1.3) | MI-SVM g |
| Musk2 | 39/63 | 166 | 65 | 91.2 (1.8) | * 95.3 (1.5) | MILES g |
| C. African | 100/1900 | 9 | 4 | 96.2 (0.2) | 95.7 (0.4) | minmin |
| C. Beach | 100/1900 | 9 | 4 | 98.9 (0.2) * | 90.7 (0.9) | RELIC |
| C. Historical | 100/1900 | 9 | 4 | 99.2 (0.1) * | 92.9 (0.5) | EM-DD |
| C. Buses | 100/1900 | 9 | 4 | 97.4 (0.2) | * 99.5 (0.1) | minmin |
| C. Dinosaurs | 100/1900 | 9 | 4 | 96.4 (0.1) | * 99.9 (0.0) | MILES p |
| C. Elephants | 100/1900 | 9 | 4 | 97.1 (0.1) | 96.9 (0.2) | minmin |
| C. Food | 100/1900 | 9 | 4 | 99.4 (0.1) * | 97.2 (0.2) | minmin |
| Fox | 100/100 | 230 | 7 | 73.3 (1.4) * | 69.8 (1.7) | MILES g |
| Tiger | 100/100 | 230 | 6 | 92.6 (1.0) * | 87.2 (1.7) | MILES g |
| Elephant | 100/100 | 230 | 7 | 95.8 (0.9) * | 91.1 (1.2) | MI-SVM g |
| Protein | 25/168 | 9 | 138 | 74.9 (2.3) | * 89.5 (1.4) | minmin |
| Harddrive1 | 191/178 | 61 | 185 | 99.6 (0.2) * | 98.6 (0.1) | MILES g |
| Harddrive2 | 178/191 | 61 | 185 | 99.5 (0.1) * | 98.6 (0.2) | RELIC |
| Mutagenesis1 | 125/63 | 7 | 56 | 92.1 (1.3) | 91.0 (0.5) | cov-coef |
| Mutagenesis2 | 13/29 | 7 | 51 | 86.0 (3.5) | 84.0 (3.4) | EMD |
| B. BrownCreeper | 197/351 | 38 | 19 | 99.5 (0.0) * | 96.5 (0.5) | RELIC |
| B. WinterWren | 109/439 | 38 | 19 | 99.8 (0.1) * | 99.3 (0.1) | summin |
| B. Pacifics. | 165/383 | 38 | 19 | 96.1 (0.2) * | 95.7 (0.3) | MILES g |
| B. Red-breasted. | 82/466 | 38 | 19 | 99.2 (0.2) | 98.7 (0.4) | MILBoost |
| UCSBBreast. | 26/32 | 708 | 35 | 84.5 (2.5) | * 92.2 (3.1) | cov-coef |
| Newsgroups1 | 50/50 | 200 | 54 | 78.8 (2.6) | * 89.8 (1.6) | meanmin |
| Newsgroups2 | 50/50 | 200 | 31 | 63.0 (4.0) | * 78.1 (1.4) | meanmean |
| Newsgroups3 | 50/50 | 200 | 52 | 76.3 (4.1) | 77.4 (1.5) | meanmean |
| Web1 | 21/54 | 5863 | 29 | 86.5 (2.6) | * 91.9 (0.0) | MI-SVM g |
| Web2 | 18/57 | 6519 | 30 | 50.7 (7.8) | * 90.1 (0.5) | MI-SVM g |
| Web3 | 14/61 | 6306 | 34 | 73.4 (6.7) | * 91.8 (0.4) | MI-SVM g |
| Web4 | 55/20 | 6059 | 31 | 80.0 (4.2) | * 99.4 (0.0) | mean-inst |

Since an exact experimental protocol is provided as a part of the referenced evaluation, we followed that protocol precisely. For each dataset, the protocol provides indexes of all splits in 5-times repeated 10-fold cross-validation. The material, however, does not specify any approach for hyperparameter optimization. Therefore, we evaluated the proposed model using default parameter settings. We set the number of trees to grow to $M = 500$ that should ensure convergence of the ensemble, the number of randomly selected features at each

split to square root of the feature dimension $K = \sqrt{D}$, which is the default value for tree-based models, and the number of uniformly drawn values of v and r to $T = 8$.

Table 1 summarizes results from the evaluation in terms of average scores and standard deviations. Although among the prior art (28 MIL classifiers) there is no single winning solution and almost each problem is associated with a different classifier, which demonstrates the difficulty and diversity of MIL problems, the proposed model was able to outperform the best prior art algorithm for a given dataset in 17 out of 29 cases. The most significant improvement with respect to the prior art is on the group of image classification problems (Fox, Tiger and Elephant) and on some scene classification problems (Corel Beach and Corel Historical). On the other hand, the proposal is less accurate on text classification problems (Newsgroups⁶ and Web), Protein and Breast datasets.

From Table 2 showing ranking of algorithms in the evaluation, it can be observed that the second best classifier with the lowest average rank (MI-SVM [2] with Gaussian kernel) ranked first only three times. Overall, the proposed algorithm works very reliably even without any hyperparameter tuning. Indeed, the proposal never ended on any of the last three positions, which is unique among all classifiers. It should be stressed though that not all prior art classifiers were evaluated on all 29 datasets. Column N/A of Table 2 indicates the number of missing evaluations.

Table 2. Number of times that each algorithm obtained each rank in the evaluation.

| Rank position | | | | | | | | | | | | | | Avg. rank | N/A | Algorithm |
|---------------|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----------|-----|------------------|
| 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | 11. | 12. | 13. | 14. | | | |
| 17 | 1 | 0 | 3 | 2 | 2 | 1 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 3.1 | 0 | BLRT <i>ours</i> |
| 0 | 2 | 1 | 4 | 2 | 1 | 0 | 5 | 3 | 0 | 3 | 0 | 1 | 2 | 7.5 | 5 | EM-DD [25] |
| 0 | 1 | 1 | 6 | 4 | 1 | 1 | 1 | 3 | 2 | 2 | 3 | 1 | 1 | 7.5 | 2 | MILBoost [23] |
| 3 | 3 | 3 | 5 | 1 | 1 | 0 | 2 | 2 | 1 | 0 | 3 | 1 | 1 | 6.0 | 3 | MI-SVM g [2] |
| 1 | 0 | 2 | 2 | 5 | 2 | 6 | 6 | 1 | 2 | 0 | 1 | 0 | 0 | 6.5 | 1 | MILES p [7] |
| 2 | 5 | 6 | 1 | 0 | 2 | 1 | 1 | 2 | 0 | 1 | 3 | 0 | 4 | 6.5 | 1 | MILES g [7] |
| 1 | 5 | 3 | 1 | 2 | 2 | 2 | 1 | 4 | 2 | 2 | 1 | 1 | 2 | 6.9 | 0 | mean-inst [13] |
| 1 | 2 | 1 | 2 | 0 | 0 | 8 | 4 | 1 | 5 | 0 | 3 | 2 | 0 | 7.8 | 0 | cov-coef [13] |
| 0 | 3 | 2 | 0 | 0 | 3 | 0 | 1 | 3 | 6 | 3 | 4 | 4 | 0 | 8.9 | 0 | RELIC [19] |
| 2 | 3 | 2 | 1 | 3 | 4 | 4 | 0 | 0 | 2 | 5 | 0 | 1 | 1 | 6.7 | 1 | minmin [9] |
| 0 | 2 | 2 | 2 | 0 | 2 | 3 | 1 | 2 | 2 | 5 | 4 | 2 | 1 | 8.6 | 1 | summin [9] |
| 0 | 2 | 3 | 4 | 5 | 3 | 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 6.7 | 2 | meanmin [9] |
| 3 | 0 | 0 | 1 | 2 | 1 | 0 | 3 | 3 | 4 | 1 | 3 | 5 | 1 | 8.9 | 2 | meanmean [9] |
| 0 | 1 | 2 | 1 | 2 | 1 | 2 | 0 | 2 | 0 | 2 | 1 | 2 | 0 | 7.5 | 13 | EMD [24] |

⁶ Except for Newsgroup3 where the proposal is competitive with the best prior art.

The non-parametric Wilcoxon signed ranks test [10] (testing whether two classifiers have equal performance) confirmed at significance level $\alpha = 0.05$ that the proposed bag-level randomized trees are superior to any other involved method. The test compared pair-wisely the proposal with every prior art method, each time using an intersection of their available datasets. The two most similarly performing methods are mean-inst [13] (p-value 0.037) and MI-SVM [2] with Gaussian kernel (p-value 0.022).

Besides the above evaluation, we also provide comparison to other tree-based MIL algorithms in Table 3, namely RELIC [19], MIOForest [21], MIForest [17], MITI [4] and RF [5]. Except for RELIC, all of them operate on *instance-level*; labels are assigned to instances and a bag is positive if it contains at least one positive instance. RF represents a naive approach where standard single-instance Random Forests are trained directly on instances that inherited bag labels. Reported classification accuracies in Table 3 are taken from the work of MIOForest [21]. Unfortunately, the classifiers were evaluated only on five pioneering datasets (i.e. Musk1-2 and the image classification problems) and their implementations are not publicly available. As can be seen from Table 3, the proposal clearly outperforms all the prior tree-based MIL solutions on these datasets.

Table 3. Comparison with other tree-based MIL classifiers. Scores refer to accuracy in percent ($\text{ACC} \times 100$). The prior art results are taken from the work of MIOForest [21].

| | BLRT ^{ours} | RELIC | MIOForest | MIForest | MITI | RF |
|----------|----------------------|-------|-----------|----------|------|----|
| Musk1 | 96 | 83 | 89 | 85 | 84 | 85 |
| Musk2 | 91 | 81 | 87 | 82 | 88 | 78 |
| Fox | 75 | 66 | 68 | 64 | N/A | 60 |
| Tiger | 90 | 78 | 83 | 82 | N/A | 77 |
| Elephant | 93 | 80 | 86 | 84 | N/A | 74 |

In Fig. 1, we assess various variants of the proposed algorithm. Dots in each subplot represent the 29 datasets. Their (x, y) coordinates are given by AUC scores obtained by the tested variants. If a dot lies on the diagonal (i.e. $x = y$ line), there is no difference between the two tested variants from that particular dataset perspective. The first two **subplots (a-b)** illustrate the influence of the ensemble size. It can be observed that it is significantly better to use 100 trees than 10 trees, but building 500 trees usually does not bring any additional performance. Also, according to **subplot (c)**, there is almost no difference between Information gain [18] and Gini impurity measure [6] scoring functions for selecting splitting rules. The next **subplot (d)** indicates that using higher values (e.g. 16 instead of the default 8) for parameter T (i.e. the number of randomly generated values for parameters v and r at each split) might lead to over-fitting

on some datasets. In **subplot (e)** we tested a variant with an absolute count⁷ instead of the relative one used in Eq. 2. The variant with the absolute count, however, performed significantly worse on the majority of datasets. The last **subplot (f)** compares the proposed algorithm with its simplified alternative, where traditional Random Forests are trained on a non-optimized bag representation. To do so, all bags $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$ are transformed into single feature vectors $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ of values $b_{\mathcal{B}}^{(f,v)} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \mathbb{1}[x_f > v]$, where for each feature f eight equally-spaced values v are generated from interval $[x_f^{\min}, x_f^{\max}]$ that is estimated beforehand on the whole training sample set. As a result, the non-optimized bag representation is eight times longer than the dimensionality of instances. As can be seen from subplot (f), the Random Forests trained on the non-optimized bag representation are far inferior to the proposed algorithm on all datasets except one. This result highlights the importance to simultaneously optimize the representation parameters with the classification ones as proposed in Sect. 3.

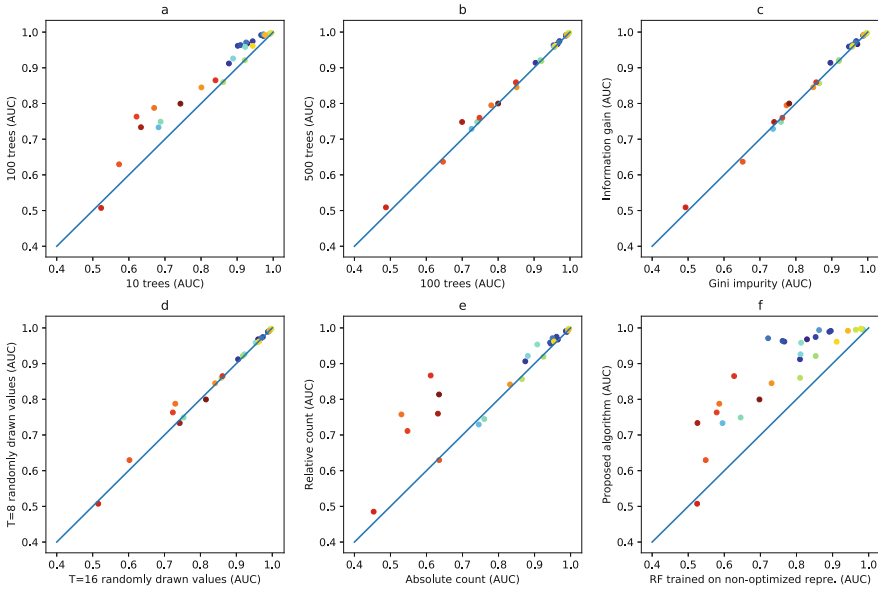


Fig. 1. Pair-wise comparisons of various configurations of the proposed algorithm on the 29 datasets. Subplots (a-b) illustrate the influence of the ensemble size, subplot (c) the impact of selected impurity measure, subplot (d) the effect of parameter T, subplot (c) the performance of the variant with the absolute count and subplot (d) compares the proposed algorithm with RF trained on the non-optimized bag representation.

⁷ The sum in Eq. 2 is not normalized by bag size $|\mathcal{B}|$ and parameter r can take values from interval $[1, \text{argmax}_{\mathcal{B} \in \mathcal{S}} |\mathcal{B}|]$.

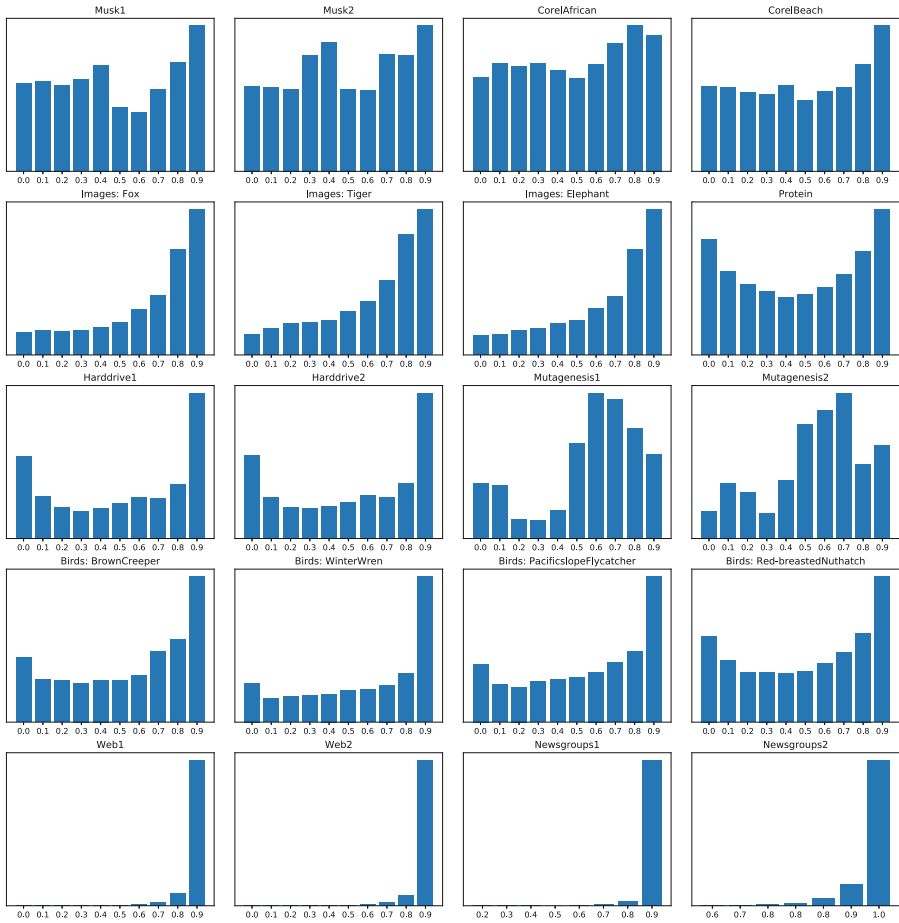


Fig. 2. Histograms of learned values of r parameters (Eq. 2). Datasets from the same source (e.g. Musk1-2, Harddrive1-2 and so forth) usually have very similar distributions that differ from the others.

Finally, Fig. 2 shows histograms of learned values of r parameters for some datasets. The first observation is that datasets from the same source (e.g. Fox, Tiger and Elephant) have very similar distributions. This demonstrates that the learned knowledge of randomized trees is not totally random as it might appear to be from the algorithm description. The next observation is that in almost all histograms (except for Mutagenesis problems) one or both extreme values of the parameter (i.e. $r \in \{0, 0.9\}$) are the most frequent ones. As discussed in Sect. 3, the behavior of splitting rules (Eq. 2) with extreme values is approaching to the behavior of the universal or existential quantifier. On Web and Newsgroup datasets, this behavior is even dominant, meaning that the algorithm reduces to the prior art solution RELIC [19]. In the rest cases, however, the added

parameter enabled to learn important dataset properties, which is supported by the high level performance reported in this section.

5 Conclusion

In this paper, we have proposed a tree-based algorithm for solving MIL problems called bag-level randomized trees (BLRT)⁸. The algorithm naturally extends traditional single-instance trees, since bags with single instances are processed in the standard single-instance tree way. Multiple instance bags are judged by counting the percent of their instances that accomplish the condition testing whether a feature value is greater than a certain threshold. Judging this percent value is done through an additional parameter that is learned during the tree building process.

Extreme values of the parameter reduce the proposal to the prior art solution RELIC [19]. Unlike other prior art tree-based algorithms, the proposal operates on the *bag-level*. Ability to analyze global bag-level information is most likely responsible for the superior performance. On the other hand, the algorithm does not identify positive instances within positively classified bags, which can be useful in some applications (e.g. object tracking [3]).

The algorithm falls into the category of *embedded-space* methods, since the learning procedure can be decoupled into two steps: embedding bags into single feature vectors and training traditional trees on top of the new representation. Features of the new representation then correspond to the counted percent values. The presented single-step approach, however, jointly optimizes the representation and the tree classifier.

As a side effect, the algorithm inherits all desirable properties of tree-based learners. It is assumption-free, scale invariant and robust to noisy and missing features. It can handle both numerical and categorical features. And, it can be easily extended to multi-class and regression problems.

Acknowledgments. This research has been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS16/235/OHK3/3T/13.

References

1. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. *Artif. Intell.* **201**, 81–105 (2013). <https://doi.org/10.1016/j.artint.2013.06.003>
2. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: *Proceedings of the 15th International Conference on Neural Information Processing Systems, NIPS 2002*, pp. 577–584. MIT Press, Cambridge (2002). <http://dl.acm.org/citation.cfm?id=2968618.2968690>

⁸ Source codes are accessible at <https://github.com/komartom/BLRT.jl>.

3. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 983–990, June 2009. <https://doi.org/10.1109/CVPR.2009.5206737>
4. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005, pp. 57–64. ACM, New York (2005). <http://doi.acm.org/10.1145/1102351.1102359>
5. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001). <https://doi.org/10.1145/1102351.1102359>
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: *Classification and Regression Trees*. CRC Press, Boca Raton (1984)
7. Chen, Y., Bi, J., Wang, J.Z.: Miles: multiple-instance learning via embedded-instance selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(12), 1931–1947 (2006). <https://doi.org/10.1109/TPAMI.2006.248>
8. Cheplygina, V., Tax, D.M.J.: Characterizing multiple instance datasets. In: Feraĝen, A., Pelillo, M., Loog, M. (eds.) *SIMBAD 2015*. LNCS, vol. 9370, pp. 15–27. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24261-3_2
9. Cheplygina, V., Tax, D.M., Loog, M.: Multiple instance learning with bag dissimilarities. *Pattern Recogn.* **48**(1), 264–275 (2015). <https://doi.org/10.1016/j.patcog.2014.07.022>. <http://www.sciencedirect.com/science/article/pii/S0031320314002817>
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006). <http://dl.acm.org/citation.cfm?id=1248547.1248548>
11. Dietterich, T.G., Lathrop, R.H., Lozano-Prez, T.: Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.* **89**(1), 31–71 (1997). [https://doi.org/10.1016/S0004-3702\(96\)00034-3](https://doi.org/10.1016/S0004-3702(96)00034-3). <http://www.sciencedirect.com/science/article/pii/S0004370296000343>
12. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **15**(1), 3133–3181 (2014). <http://dl.acm.org/citation.cfm?id=2627435.2697065>
13. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: *Proceedings of the Nineteenth International Conference on Machine Learning, ICML 2002*, pp. 179–186. Morgan Kaufmann Publishers Inc., San Francisco (2002). <http://dl.acm.org/citation.cfm?id=645531.656014>
14. Gehler, P.V., Chapelle, O.: Deterministic annealing for multiple-instance learning. In: *Artificial Intelligence and Statistics*, pp. 123–130 (2007)
15. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. *Mach. Learn.* **63**(1), 3–42 (2006)
16. Kohout, J., Komárek, T., Čech, P., Bodnár, J., Lokoč, J.: Learning communication patterns for malware discovery in https data. *Expert Systems with Applications* **101**, 129–142 (2018). <https://doi.org/10.1016/j.eswa.2018.02.010>. <http://www.sciencedirect.com/science/article/pii/S0957417418300794>
17. Leistner, C., Saffari, A., Bischof, H.: MIForests: multiple-instance learning with randomized trees. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) *ECCV 2010*. LNCS, vol. 6316, pp. 29–42. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15567-3_3
18. Quinlan, J.R.: Induction of decision trees. *Mach. Learn.* **1**(1), 81–106 (1986). <https://doi.org/10.1023/A:1022643204877>
19. Ruffo, G.: Learning single and multiple instance decision trees for computer security applications. University of Turin, Torino (2000)

20. Stiborek, J., Pevný, T., Reháč, M.: Multiple instance learning for malware classification. *Expert Syst. Appl.* **93**, 346–357 (2018). <https://doi.org/10.1016/j.eswa.2017.10.036>. <http://www.sciencedirect.com/science/article/pii/S0957417417307170>
21. Straehle, C., Kandemir, M., Koethe, U., Hamprecht, F.A.: Multiple instance learning with response-optimized random forests. In: 2014 22nd International Conference on Pattern Recognition, pp. 3768–3773, August 2014. <https://doi.org/10.1109/ICPR.2014.647>
22. Tax, D.M.J.: A matlab toolbox for multiple-instance learning, version 1.2.2, Faculty EWI, Delft University of Technology, The Netherlands, April 2017
23. Zhang, C., Platt, J.C., Viola, P.A.: Multiple instance boosting for object detection. In: Weiss, Y., Schölkopf, B., Platt, J.C. (eds.) *Advances in Neural Information Processing Systems*, vol. 18, pp. 1417–1424. MIT Press (2006). <http://papers.nips.cc/paper/2926-multiple-instance-boosting-for-object-detection.pdf>
24. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. *Int. J. Comput. Vision* **73**(2), 213–238 (2007)
25. Zhang, Q., Goldman, S.A.: Em-dd: an improved multiple-instance learning technique. In: *Advances in Neural Information Processing Systems*, pp. 1073–1080. MIT Press (2001)