Explainable Multiple Instance Learning with Instance Selection Randomized Trees

Tomáš Komárek[1,2](🖂), Jan Brabec[1,2], and Petr Somol[3]

[1] Czech Technical University in Prague, FEE, Czech Republic
[2] Cisco Systems, Cognitive Intelligence, Prague, Czech Republic
{tomkomar,janbrabe}@cisco.com
[3] Avast Software, Prague, Czech Republic, petr.somol@avast.com

Abstract. Multiple Instance Learning (MIL) aims at extracting patterns from a collection of samples, where individual samples (called bags) are represented by a group of multiple feature vectors (called instances) instead of a single feature vector. Grouping instances into bags not only helps to formulate some learning problems more naturally, it also significantly reduces label acquisition costs as only the labels for bags are needed, not for the inner instances. However, in application domains where inference transparency is demanded, such as in network security, the sample attribution requirements are often asymmetric with respect to the training/application phase. While in the training phase it is very convenient to supply labels only for bags, in the application phase it is generally not enough to just provide decisions on the bag-level because the inferred verdicts need to be explained on the level of individual instances. Unfortunately, the majority of recent MIL classifiers does not focus on this real-world need. In this paper, we address this problem and propose a new tree-based MIL classifier able to identify instances responsible for positive bag predictions. Results from an empirical evaluation on a large-scale network security dataset also show that the classifier achieves superior performance when compared with prior art methods.

Keywords: Explainable AI · Network security · Randomized trees.

1 Introduction

Multiple Instance Learning (MIL) generalizes the traditional data representation as it allows individual data samples $\mathcal{B}_1, \mathcal{B}_2, \ldots$ (called bags) to be represented by a group of multiple d-dimensional feature vectors $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots\},\ \mathbf{x} \in \mathbb{R}^d$ (called instances), which are order independent and their counts may vary across bags. In the supervised classification, it is further assumed that each bag is associated with label y (e.g. $y \in \{-1, +1\}$ in the binary case), and the goal is to infer function $\mathcal{F}$ from dataset $\mathcal{D} = \{(\mathcal{B}, y)_1, (\mathcal{B}, y)_2, \ldots\}$, using algorithm $\mathcal{A}$, such that the function $\mathcal{F}$ can predict labels for new bags $\mathcal{F}(\mathcal{B}) = y$.

This formalism (originally introduced in [9]) has recently gained significant traction in domains dealing with complex data structures and/or labeling limitations [6]. A prime example of such a domain is network security and its problem of detecting infected users in computer networks. While network traffic logs can be converted into feature vectors relatively easily, their labeling is often very labor-intensive and time-consuming. This is caused not only by the volume of logs that needs to be processed by experienced threat analysts (and can not be outsourced due to privacy issues), but also by the fact that the individual log records may not carry enough discriminatory information for making verdicts about them and a broader context of the communication must be considered by the analysts[1]. Previous works [10, 18, 22, 13] have shown that MIL can greatly facilitate this problem as it enables to: *(i) formulate the problem more naturally:* users can be represented by bags of instances, where the instances correspond to users' communications with web servers; here the flexibility in bag sizes reflects the reality that each user can establish a different number of communications within a given time window, *(ii) reduce the label acquisition costs:* threat analysts do not have to pinpoint individual log records responsible for the infection; it is enough to provide labels for the whole users, *(iii) open new ways of acquiring labels:* since it is sufficient to know whether the user was infected in a particular time period or not, a completely separate source of data can be used for annotating (e.g. anti-virus reports) and thus benefit from cheaper and less ambiguous labels, *(iv) increase classification performance:* MIL classifiers can detect more types of infections as they make decisions by analysing the entire context of user's communications rather than individual log records in isolation.

The goal of this paper is to contribute to the above list of MIL advantages by further enabling to: *(v) explain the raised alerts:* although in the training phase the labels are supplied only for bags, in the application phase the model should be able to explain the raised alerts (positive bag predictions) by promoting instances responsible for the decisions. We argue that this capability is of great need, especially in applications where subsequent acting upon the raised alerts is associated with high costs (e.g. reimaging a user's computer) and therefore the verdicts need to be well justified. Time spent on the justification is usually strongly affected by the order in which the instances are investigated. Most prior approaches can not effectively prioritize instances of positive bags because they perform some sort of bag aggregation inside models to make learning on bags (i.e. sets of vectors) possible. The algorithm proposed in this paper works on an instance selection rather than the aggregation principle.

2 Instance Selection Randomized Trees

The algorithm for learning Instance Selection Randomized Trees (ISRT) follows the standard top-down greedy procedure for building an ensemble of unpruned decision trees. Every tree learner recursively divides the training sample set into

[1] For example, a seemingly legitimate request to google.com might be in reality related to malicious activity when it is issued by malware checking Internet connection. Similarly, requesting ad servers in low volumes is considered as a legitimate behavior, but higher numbers might indicate Click-fraud infection.

two subsets until class homogeneity is reached or the samples can not be divided any further.

The main difference to the standard (single instance) tree-based learners applies in the way the conditions are evaluated inside the splitting nodes. In the MIL setting, the decision whether to send a sample (i.e. bag) to the left or right branch can no longer be based on a condition of type — if feature f is greater than value v — as the bag might contain multiple feature vectors (i.e. instances) that may or may not fulfill that condition. To cope with this problem, every node of ISRT (denoted as $\mathcal{N}_{\text{ISRT}}$) is further parametrized with vector $\mathbf{w}$, called instance selector, in addition to the feature index f and the threshold value v (Equation 1). The purpose of the instance selector is to select a single instance $\mathbf{x}^*$ from a bag $\mathcal{B}$ upon which the feature value comparison $x^*_f > v$ is made. The selection mechanism is implemented via calculating the inner product (denoted as $\langle\cdot,\cdot\rangle$) between the vector $\mathbf{w}$ and individual bag instances $\mathbf{x} \in \mathcal{B}$, followed by selecting the instance $\mathbf{x}^*$ associated with the maximum response. Note that if bags are of size one, then ISRT nodes behave like the traditional ones regardless of the extra parameter $\mathbf{w}$.

$$\mathcal{N}_{\text{ISRT}}(\mathcal{B}; \underbrace{f, v, \mathbf{w}}_{\Phi}) = \begin{cases} \text{left}, & \text{if } x^*_f > v,\ \mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{B}} \langle \mathbf{w}, \mathbf{x}\rangle, \\ \text{right}, & \text{otherwise.} \end{cases} \tag{1}$$

Assuming the positive class is the class of interest, we would like to train an instance selector (on a local training subset available to the considered node) to give maximum values to the instances of positive bags (and thus cause their selection) that are most responsible for these bags being positive. More specifically, the selector should assign low (i.e. negative) values to all instances of negative bags and high (i.e. positive) values to at least one instance from each positive bag. We do not force the selector to assign high values to all instances of positive bags, since not all of them are necessarily relevant. For example, not all websites visited by an infected user within the last 24 hours are automatically malicious. In fact, the vast majority of them will typically still be legitimate. These requirements lead to the following zero-one loss function for a single training data point $(\mathcal{B}, y)$:

$$\ell_{01}\left(\mathbf{w}; (\mathcal{B}, y)\right) = \mathbb{1}\left[y \max_{\mathbf{x}\in\mathcal{B}} \langle \mathbf{w}, \mathbf{x}\rangle < 0\right], \tag{2}$$

where $\mathbb{1}[\cdot]$ stands for an indicator function, which equals one if the argument is true and zero otherwise. If we approximate the indicator function $\mathbb{1}[z]$ with hinge loss surrogate $\max\{0, 1-z\}$, take average over the local training subset $\mathcal{S} \subseteq \mathcal{D}$ and add regularization term λ, we obtain Multiple Instance Support Vector Machines [2] optimization problem:

$$\operatorname*{argmin}_{\mathbf{w}} \ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{|\mathcal{S}|} \sum_{(\mathcal{B},y)\in\mathcal{S}} \max\{0, 1 - y \max_{\mathbf{x}\in\mathcal{B}} \langle \mathbf{w}, \mathbf{x}\rangle\}. \tag{3}$$

To approximately solve this non-convex optimization problem in linear time, we adapted Pegasos solver [21] (originally designed for conventional SVMs) to the MIL setting. The resulting pseudo-code is given in Algorithm 1. It is a stochastic sub-gradient descent-based solver, which at each iteration t updates the current solution $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta^t \nabla^t$ (row 9 in Algorithm 1) using step size $\eta^t = 1/(t\lambda)$ and sub-gradient ∇^t of the objective function (Equation 3) estimated on a single randomly chosen training sample. To avoid building strong classifiers inside nodes, which would go against the randomization principle for constructing diverse independent trees [4], we restrict the selectors to operate on random low-dimensional subspaces. Input zero-one vector $\mathbf{s} \in \{0,1\}^d$ then serves as a mask defining the feature subspace. By taking element-wise product with that vector (i.e. $\mathbf{s} \odot \mathbf{w}$ or $\mathbf{s} \odot \mathbf{x}$), only feature positions occupied by ones remain effective. In Section 4.2, we empirically demonstrate that using sparse selectors, where the number of effective dimensions equals to the square root of the total dimensions d rounded to the closest integer (i.e. $\sum_f s_f = [\sqrt{d}]$), plays a crucial role in the overall ensemble performance. This subspace size ensures that in high dimensions the selectors will be approximately orthogonal and thus independent [12].

Algorithm 1: ISRT's routine for training selectors.

```
Function TrainSelector($\mathcal{S}; \lambda, E, \mathbf{s}$)
    Input  : Training set of bags along with labels $\mathcal{S} = \{(\mathcal{B}, y)_1, \ldots\}$,
             the regularization $\lambda > 0$,
             the number of epochs $E > 0$,
             the zero-one vector defining feature subspace $\mathbf{s}$.
    Output: Instance-level selector $\mathbf{w}^t$ approximately solving Problem 3.
1   extend all instances by a bias term $[\mathbf{x}, 1]$ including the subspace vector $[\mathbf{s}, 1]$
2   $t \leftarrow 1$
3   $\mathbf{w}^0 \leftarrow$ random vector $w_f^0 \sim N(0,1)$  // length($\mathbf{w}$) = length($\mathbf{x}$) = length($\mathbf{s}$)
4   $\mathbf{w}^1 \leftarrow \mathbf{s} \odot \mathbf{w}^0$                         // $\odot$ element-wise product
5   for 1 in $E$ do
6       for 1 in $|\mathcal{S}|$ do
7           $(\mathcal{B}, y) \leftarrow$ (class-balanced) random draw (with replacement) from $\mathcal{S}$
8           $\mathbf{x}^* \leftarrow \mathrm{argmax}_{\mathbf{x} \in \mathcal{B}} \langle \mathbf{w}^t, \mathbf{x} \rangle$
9           $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \frac{1}{t\lambda}\left(\lambda \mathbf{w}^t - \mathbb{1}\left[y\langle \mathbf{w}^t, \mathbf{x}^*\rangle < 1\right] y(\mathbf{s} \odot \mathbf{x}^*)\right)$
10          $t \leftarrow t + 1$
11  return $\mathbf{w}^t$[start : end $-$ 1]                  // removing the bias term
```

Being equipped with the routine for training selectors, we can represent bags in a local training subset $\{(\mathcal{B}, y)_1, \ldots\}$ with selected instances $\{(\mathbf{x}^*, y)_1, \ldots\}$. Now, on top of this representation, a standard search for the best splitting parameters, based on measuring purity of produced subsets (e.g. Information gain [20] or Gini impurity [5]), can be executed. This allows us to build an ensemble of ISRT in the same way as (Extremely) Randomized Trees [11] are. In particular, unlike e.g. Breiman's Random Forests [4], where each tree is grown on a bootstrap replica of the training data, the Randomized Trees (as well as our ISRT) are grown on the complete training set, which yields to a lower bias. Variance is then reduced by output aggregation of more diversified trees. The higher diversification is achieved through stronger randomization in splitting nodes, as for each feature f out of $[\sqrt{d}]$ randomly selected ones, only a limited number[2] T of uniformly drawn threshold values v from $[x_f^{\min}, x_f^{\max})$ is considered for splitting rather than every sample value as realized in Breiman's Random Forests. In the case of ISRT, the randomization is even stronger due to the fact that selected instances may vary from node to node. The whole training procedure of ISRT is summarized in Algorithm 2. Time complexity of the algorithm, assuming balanced trees, is $\Theta(M\sqrt{d}\,T\;E\;N_I \log N_{\mathcal{B}})$, where M is the number of constructed trees, E the number of epochs for training selectors, N_I the average number of instances in bags and $N_{\mathcal{B}}$ the number of training bags.

In the testing phase, a bag to be classified is propagated through individual trees and the final score $\hat{y} \in [-1, 1]$ is calculated as an average of leaves scores the bag falls into (Algorithm 3). However, besides the prediction score, the ISRT can also output a histogram of selection counts over the bag instances $\mathbf{i}$. This information might help to identify relevant instances upon which the decision was made and thus serve as an explanation for positive bag predictions. For example, an explanation — an user was found to be infected because it has communicated with these three hostnames (out of hundreds) within the last 24 hours — can greatly speed up the work of threat analysts and reinforce their trust in the model if the hostnames will be shown to be indeed malicious. On the other side, it should be noted that this approach can not explain a positive prediction that would be based on an absence of some type of instance(s) in the bag. For example, there might be malware, hypothetically, its only visible behaviour would be preventing an operating system (or other applications like an anti-virus engine) from regular updates. For the same reason, negative predictions in general can not be explained with this approach.

3 Related Work

Surveys on MIL [1, 6] categorize classification methods into two major groups according to the level at which they operate. Methods from the first *instance-level* group (proposed mostly by earlier works) construct instance-level classifiers $f(\mathbf{x}) \rightarrow \{-1, +1\}$ as they assume that the discriminative information lies on the level of instances. Meaning that a bag is positive if it contains at least one instance carrying a characteristics positive signal, and negative if does not contain any such instance. Bag-level predictions are then obtained by aggregating instance-level verdicts $\mathcal{F}(\mathcal{B}) = \max_{\mathbf{x}\in\mathcal{B}} f(\mathbf{x})$. MI-SVM [2], which is defined in its primary form in Equation 3, is a representative example of this group.

The second *bag-level* group involves methods (proposed mainly by later works) assuming that the discriminative information lies at the level of bags. These

[2] Term *extremely* in Extremely Randomized Trees [11] corresponds to setting $T = 1$.

Algorithm 2: ISRT's routine for building an ensemble of trees.

```
Function BuildTreeEnsemble($\mathcal{D}; M, T, E, \Lambda$)
    Input  : Training set of bags along with labels $\mathcal{D} = \{(\mathcal{B}, y)_1, \ldots\}$,
             the number of trees to grow $M > 0$,
             the number of considered threshold values $T > 0$,
             the number of epochs $E > 0$ (for training selectors).
    Output: Ensemble of Instance Selection Randomized Trees $\mathcal{E}$.
1   $\mathcal{E} \leftarrow \emptyset$
2   for 1 in $M$ do
3       $\mathcal{E} \leftarrow \mathcal{E} \cup \{\text{BuildTree}(\mathcal{D}; T, E, \Lambda)\}$
4   return $\mathcal{E}$

Function BuildTree($\mathcal{S}; T, E$)
    Input  : Local training subset $\mathcal{S} \subseteq \mathcal{D}$.
    Output: Node with followers or Leaf with a prediction score.
5   if all class labels $y$ in $\mathcal{S}$ are equal then
6       return Leaf ($y$)
7   $\Phi^* \leftarrow$ FindBestSplittingParameters($\mathcal{S}; T, E$)
8   if $\Phi^* = \emptyset$ then
9       return Leaf ($\frac{1}{|\mathcal{S}|} \sum_{y \in \mathcal{S}} y$)
10  $\mathcal{S}_{\text{left}} \leftarrow \{(\mathcal{B}, y) \in \mathcal{S} \mid \mathcal{N}_{\text{ISRT}}(\mathcal{B}; \Phi^*) = \text{left}\}$
11  $\mathcal{S}_{\text{right}} \leftarrow \mathcal{S} \setminus \mathcal{S}_{\text{left}}$
12  if $\mathcal{S}_{\text{left}} = \emptyset$ or $\mathcal{S}_{\text{right}} = \emptyset$ then
13      return Leaf ($\frac{1}{|\mathcal{S}|} \sum_{y \in \mathcal{S}} y$)
14  return Node ($\Phi^*$, BuildTree($\mathcal{S}_{\text{left}}$), BuildTree($\mathcal{S}_{\text{right}}$))

Function FindBestSplittingParameters($\mathcal{S}; T, E$)
    Output: Triplet of splitting parameters $\Phi$ as defined in Equation 1.
15  $\Phi^* \leftarrow \emptyset$
16  $\mathbf{s} \leftarrow$ zero-one vector of length $d$ with $\lceil \sqrt{d} \rceil$ ones at random positions
17  $\mathbf{w} \leftarrow$ TrainSelector($\mathcal{S}; \lambda = 1, E, \mathbf{s}$)
18  $(\mathbf{X}^*, \mathbf{y}) \leftarrow$ represent each pair $(\mathcal{B}, y) \in \mathcal{S}$ with $(\text{argmax}_{\mathbf{x} \in \mathcal{B}} \langle \mathbf{w}, \mathbf{x} \rangle, y)$
19  foreach feature $f$ in $\lceil \sqrt{d} \rceil$ randomly selected ones (without replacement)
      having non-constant values in $\mathbf{X}$ (i.e. $x_f^{min} \neq x_f^{max}$) do
20      foreach value $v$ in $T$ uniformly drawn values from $[x_f^{min}, x_f^{max})$ do
21          $\Phi \leftarrow (f, v, \mathbf{w})$
22          update $\Phi^* \leftarrow \Phi$ if Score($\Phi, \mathbf{X}^*, \mathbf{y}$) is the best so far found score
23  return $\Phi^*$
```

Algorithm 3: ISRT's prediction routine.

```
Function Predict($\mathcal{B}; \mathcal{E}$)
    Input  : Bag to be classified $\mathcal{B}$ with ensemble of trees $\mathcal{E}$.
    Output: Bag score $\hat{y} \in [-1, 1]$ and histogram of selected instances $\mathbf{i}$.
1   $\hat{y} \leftarrow 0$
2   $\mathbf{i} \leftarrow$ zero vector of length $|\mathcal{B}|$
3   foreach Tree in $\mathcal{E}$ do
4       P ← Tree                      // pointer to Node and Leaf structures
5       $\mathbf{i}' \leftarrow$ zero vector of length $|\mathcal{B}|$
6       while P is of type Node do
7           $(b, i_{\mathbf{x}^*}) \leftarrow \mathcal{N}_{\mathrm{ISRT}}(\mathcal{B}; \mathsf{P}.\Phi^*)$    // $i_{\mathbf{x}^*}$ index of selected instance
8           $\mathbf{i}'[i_{\mathbf{x}^*}] \leftarrow \mathbf{i}'[i_{\mathbf{x}^*}] + 1$
9           $\mathsf{P} \leftarrow \mathsf{P}.b$                  // continue in left or right branch
10      $\hat{y} \leftarrow \hat{y} + \mathsf{P}.y$
11      $\mathbf{i} \leftarrow \mathbf{i} \oplus \left(\mathbf{i}'/(\sum_i \mathbf{i}'[i])\right)$          // $\oplus$ element-wise addition
12  $\hat{y} \leftarrow \hat{y}/|\mathcal{E}|$
13  $\mathbf{i} \leftarrow \mathbf{i}/|\mathcal{E}|$
14  return $(\hat{y}, \mathbf{i})$
```

methods build directly bag-level classifiers $\mathcal{F}(\mathcal{B})$ extracting information from whole bags to make decisions about their class rather than aggregating individual instance-level verdicts. Considering the global bag-level information is necessary, e.g., in a case when the positive label is caused by a co-occurrence of two specific types of instances. Since bags are non-vectorial objects, these methods typically first transform bags into single fixed-size vectors and then train an off-the-shelf classifier on top of them. Works of [13, 22] use Bag-of-Words approach, where a vocabulary of prototype instances (i.e. words) is first found by a clustering algorithm and then each bag is represented by a histogram counting how many instances fall into each cluster. Work of [19] rather proposes to use the Neural Network (NN) formalism with pooling layers (e.g. based on max/mean aggregation) to achieve simultaneous optimization of the bag representation (first layer(s) followed by a pooling layer) and the classifier (one or more of subsequent layers) by means of back-propagation. The closest work to this paper is on Bag-level Randomized Trees (BLRT) [14]. The main difference to the ISRT is in the conditions evaluated inside the splitting nodes:

$$\mathcal{N}_{\mathrm{BLRT}}(\mathcal{B}; \underbrace{f, v, r}_{\Phi}) = \begin{cases} \text{left}, & \text{if } \left[\dfrac{1}{|\mathcal{B}|}\sum_{\mathbf{x}\in\mathcal{B}} \mathbb{1}\left[x_f > v\right]\right] > r, \\ \text{right}, & \text{otherwise}, \end{cases} \tag{4}$$

where the additional parameter r determines a percentage of instances that must satisfy the inner condition $x_f > v$ to be the whole bag passed to the left branch. Surprisingly, this method has been shown to be significantly better than any of the prior 28 MIL classifiers on 29 datasets [14], although it can not extract a multivariate pattern from a single instance unless the bag contains only that one instance. This can be illustrated, e.g., on an inability to separate these two bags: $\{(0,0),(1,1)\}$ and $\{(0,1),(1,0)\}$. From none of the above bag-level methods can be trivially inferred which instances are responsible for positive bag predictions.

4 Experiments

In this section, we evaluate the proposed ISRT algorithm first on a private dataset from the network security field (Section 4.1) and then on 12 publicly available datasets from six other domains (Section 4.2). On both types of datasets, the algorithm is compared with state-of-the-art approaches: BLRT[3], NN[4] and MI-SVM[5] briefly reviewed in the previous section.

4.1 Private Dataset

The network security dataset represents a real-world problem of classifying users of computer networks as either infected or clean. The dataset contains metadata about network communications from more than 100 international corporate networks of various types and sizes.

Users in the dataset are represented as bags of instances, where the individual instances correspond to the established communications between users and hostnames within 24 hours. High-level statistics about each such communication are computed from URL strings of made HTTP requests within that time window. The procedure is as follows. First, URL strings of all HTTP requests originating from a given user and targeting to a particular hostname are collected. Then, each URL string is converted into a feature vector by extracting a well-known set of URL features. The used feature set has been already described e.g. in works [16, 10, 15] that were primarily focused on detection of URLs generated by malicious applications. As such, the feature set incorporates a lot of domain knowledge. Examples of the features are: the number of occurrences of reserved URL characters (i.e. ’_’,’-’,’?’,’!’,’@’,’#’,’&’,’%’), the digit ratio, the lower/upper case ratio, the vowel change ratio, the number of non-base64 characters, the maximum length of lower/upper case stream, the maximum length of consonant/vowel/digit stream, etc. In sum, there are 359 features. Finally, to represent the communication with a single instance, the extracted URL feature vectors are aggregated using a maximum as the aggregation function.

The dataset consists of three sets: training, validation and testing. Training set is the largest one and was collected during the period of five working days in January 2021. Validation and testing sets then cover the first and the last Wednesday in February 2021, respectively. In total, there are 118,108 unique users. On average, each has 24 instances. Detailed statistics about the dataset along with the number of infected/clean users are given in Table 1.

[3] We used implementation from `https://github.com/komartom/BLRT.jl`.
[4] We used implementation from `https://github.com/CTUAvastLab/Mill.jl`.
[5] MI-SVM is trained with Algorithm 1 for complete feature space ($\mathbf{s}$ is vector of ones).

Dataset	Training set	Validation set	Testing set
Date range	18-22 Jan 2021	3 Feb 2021	24 Feb 2021
URL strings	1,186,465,181	239,079,385	264,342,073
Communications	2,829,316	581,826	554,425
Infected users	1,830	430	380
Clean users	115,700	24,987	23,695

Table 1. Specification of the network security dataset. The aim is to train a user-level model. Counts of URL strings and communications correspond to the sum over all users and illustrate the potentially higher labeling requirements on these lower levels.

We trained the proposed ISRT algorithm with the following hyper-parameter values: the number of trees to grow $M = 100$, the number of considered threshold values $T = 8$ and the number of epochs for training selectors $E = 10$. The computational time for training took 25 minutes on a single c4.8xlarge AWS instance[6]. We also used the same hyper-parameter settings (i.e. $M = 100$ and $T = 8$) to train the prior art tree-based algorithm BLRT[7]. In the case of MIL Neural Network (NN), we performed a grid search over the following configurations: the instance layer size $\{10, 30, 60\}$, the aggregation layer type $\{\text{mean, max, mean-max}\}$ and the bag layer size $\{5, 10, 20\}$. We used rectified linear units (ReLU), ADAM optimizer, mini-batch of size 32, and the maximum number of epochs 1000. MI-SVM classifier was trained for regularization values $\lambda \in \{10^{-5}, 10^{-4}, \dots, 1\}$ and the number of epochs $E = 100$. The final configuration, in both cases, was selected based on the highest achieved performance on the validation set in terms of the area under the Precision-Recall curve.

Efficacy results of the above trained models on the validation and testing set are shown in Figure 1 using the Precision-Recall and ROC curves. Different points on the Precision-Recall curve correspond to different decision threshold values of a particular model and indicate the percentage of alarms that are actually correct (precision) subject to the percentage of detected infected users in the whole dataset (recall / true positive rate). Perfect recall (score 1) means that all infected users are detected, while perfect precision means that there are no false alarms. ROC curve then provides information about the volume of false alarms as the percentage of monitored healthy users (false positive rate)[8]. Points on the ROC curve might also serve for calculating precision under different imbalance ratios of infected to clean users [3].

[6] 36 virtual Intel Xeon CPUs @ 2.9GHz and 60Gb of memory.

[7] It was shown in the work of BLRT [14], and we confirm that for ISRT in Section 4.2, that tuning of these parameters usually does not bring any additional performance.

[8] While precision answers to the question: "With how big percentage of false alarms the network administrators will have to deal with?", false positive rate gives answer to: "How big percentage of clean users will be bothered?".

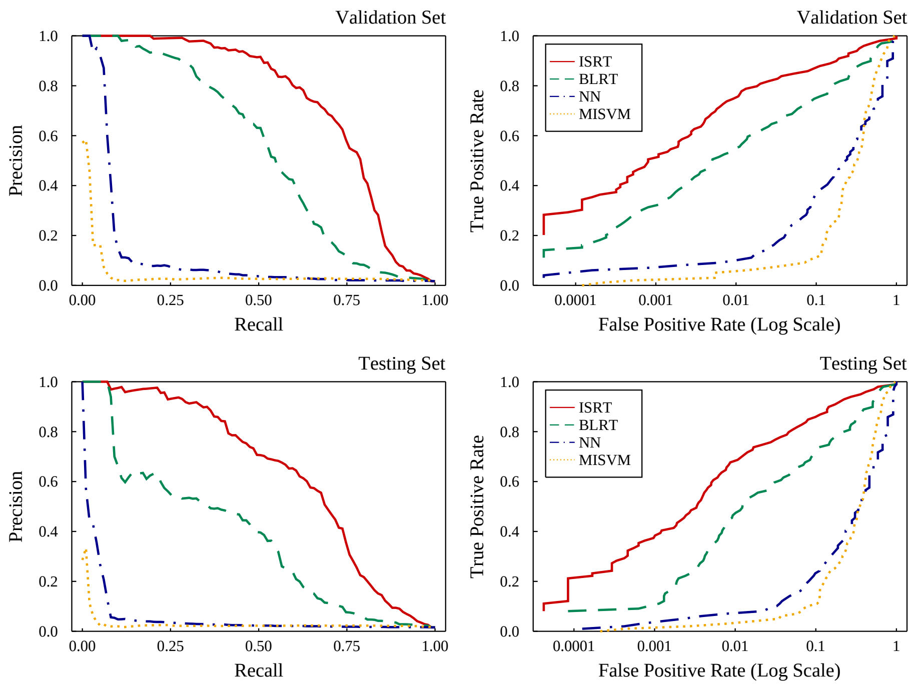

Fig. 1. Precision-Recall and ROC curves of individual models on the validation (Feb 3, 2021) and testing (Feb 24, 2021) set of the private network security dataset.

As can be seen from Figure 1, the proposed ISRT algorithm has a clear superior performance as it produces equal or less false alarms than any other involved method at any arbitrary recall on both sets. The second best method is the tree-based BLRT algorithm. It has a decent performance on the validation test (Feb 3, 2021), but on the testing set (Feb 24, 2021) the performance drops notably. This decrease in model performance over time is known as an aging effect — a model becomes obsolete as the distribution of incoming data shifts. Resistance to that is an important model characteristic, from an application point of view, albeit not always evaluated by researchers [17]. We attribute this decay to the fact that the BLRT model can extract only global bag-level univariate statistics computed across all instances within a bag. Because of this, it might be difficult to effectively separate a multivariate malicious signal hidden in a single instance from an abundant user background, which can evolve over time. On the other hand, the discriminative signal apparently does not lie on the local instance-level completely as the instance-based classifier MI-SVM performs poorly. Probably the ability to combine these two approaches, by selecting and judging individual instances according to the need while collecting global evidence, might be the reason why the ISTR model excels in this task. Interestingly, the NN model is able to reliably detect only a very limited number of infections, although in the recent work [18] it has been shown to perform well on a similar task. One reason might be that the NN model is sensitive to some hyper-parameters, which we did not tune. Another one might be that there is a substantial difference in the time window (5 minutes vs. 24 hours) per which the users are classified. Shrinking the time window to five minutes on our data leads to one instance (i.e. communication with a unique hostname) per bag on average, which would eliminate the need of MIL and its benefits (i.e. lower labeling costs and richer contextual information).

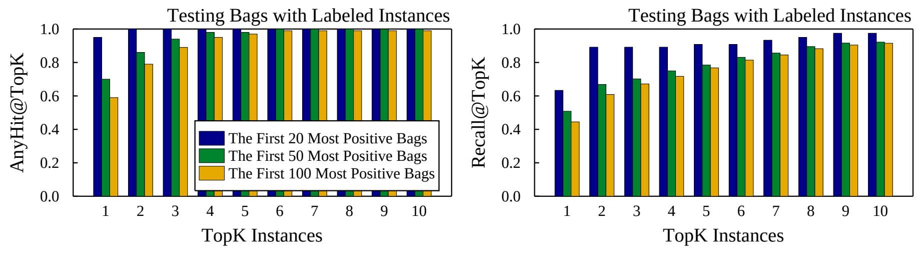

Fig. 2. Assessment of ISRT explanations as an information retrieval task. AnyHit@TopK (on the left) shows the percentage of bags for which at least one relevant (i.e. positively labeled) instance appeared among the top K instances. Since bags can contain multiple relevant instances, Recall@TopK (on the right) shows how many of them are among the top K instances. The perfect Recall@1 can not be achieved unless all bags have only one relevant instance.

As announced in the introduction to this paper, the main benefit of the newly proposed ISRT over the prior-art BLRT, besides higher performance on some datasets, is the ability to explain the positive bag predictions. The provided explanations are in the form of assigned scores to individual instances (according to the number of times they have been selected during the bag class prediction), upon which they can be sorted and presented to the end-user for judgment. Analogically to the information retrieval task, the goal is to place the most relevant instances at top positions. To assess this ability on the network security dataset, we used our internal deny list of known malicious hostnames to label instances[9] (i.e. communications with hostnames) of bags that have been classified as positive by the ISRT model on the testing set. On the first 20, 50 and 100 most positive bags (for which we had at least one positive instance-level label), we calculated AnyHit@TopK and Recall@TopK metrics. Figure 2 presents the results for the first top ten (TopK) instance positions.

It can be observed from the left subplot of Figure 2 (AnyHit@TopK) that threat analysts investigating the first 20 most positive bags would encounter the first piece of evidence for the infection (i.e. any malicious hostname) just

[9] This way of identifying malicious communications is not so effective in production, since new threats are not on the deny list yet and need to be first discovered.

by analyzing the top two recommended instances from each bag. This ability slightly decreases for the higher number of bags (i.e. 50 and 100), but it is still very useful considering the fact that the largest bags have over 100 instances, and on average only two are labeled as malicious — a needle in a haystack problem. This can be also seen from the right subplot (Recall@TopK) showing that about 50% of all positive instances in bags can be discovered just by verifying the first recommended instance ($K = 1$).

4.2 Public Datasets

To show that the use of ISRT is not limited to network security only, we evaluate the algorithm on 12 other MIL datasets from six different domains[10]. Namely, classification of molecules (Musk1-2), classification of images (Fox, Tiger, Elephant), text categorization (Newsgroups1-3), protein binding site prediction (Protein), breast cancer detection (BreastCancer) and drug activity prediction (Mutagenesis1-2). Their basic meta-descriptions (i.e. counts of positive/negative bags, average bag size and feature dimension) are given in Table 2. For more details, we refer the reader to the survey of MIL datasets [7].

Dataset	Bags +/-	Inst.	Feat.	MI-SVM	NN	BLRT	ISRT ours
Musk1	47/45	5	166	85.9 (1.9)	91.9 (1.5)	**96.8** (1.6)	**97.2** (1.3)
Musk2	39/63	65	166	86.9 (1.5)	**90.3** (2.3)	**91.2** (1.8)	**92.3** (2.6)
Fox	100/100	7	230	55.2 (1.6)	65.9 (1.2)	**73.3** (1.4)	**74.0** (1.8)
Tiger	100/100	6	230	81.7 (2.7)	**90.7** (1.7)	**92.6** (1.0)	**92.5** (0.8)
Elephant	100/100	7	230	84.5 (0.3)	**93.9** (0.8)	**95.8** (0.9)	**95.0** (0.7)
Newsgroups1	50/50	54	200	**82.4** (4.9)	77.0 (3.6)	**78.8** (2.6)	55.4 (2.6)
Newsgroups2	50/50	31	200	**70.2** (3.7)	**63.3** (5.2)	**63.0** (4.0)	**63.8** (2.9)
Newsgroups3	50/50	52	200	54.5 (5.5)	63.9 (4.1)	**76.3** (4.1)	65.0 (2.6)
Protein	25/168	138	9	81.2 (1.7)	75.2 (4.2)	74.9 (2.3)	**85.8** (2.0)
BreastCancer	26/32	35	708	73.1 (2.6)	**76.7** (8.1)	**84.5** (2.5)	79.3 (1.9)
Mutagenesis1	125/63	56	7	53.4 (1.0)	**90.2** (1.0)	**92.1** (1.3)	88.6 (0.8)
Mutagenesis2	13/29	51	7	70.0 (8.2)	66.2 (2.6)	**86.0** (3.5)	70.0 (6.6)

Table 2. Metadata about 12 public datasets. Including the number of positive/negative bags, the average number of instances inside bags and the number of features. Plus evaluation results, measured in AUC $\times$ 100, for individual models. Best results are shown in bold face. Multiple models are highlighted if the difference is not statistically significant (at $\alpha = 0.05$) according to a paired t-test with Holm-Bonferroni correction (for multiple comparisons) [8] computed on the five runs of 10-fold cross-validation.

Each dataset also contains a predefined list of splitting indices for 5-times repeated 10-fold cross-validation. Therefore, we followed this evaluation protocol

[10] Datasets are accessible at `https://doi.org/10.6084/m9.figshare.6633983.v1`.

precisely, similarly, as did the prior works of BLRT [14] and NN [19]. Since the protocol does not specify any approach for hyper-parameter optimization, we used the default values that are known to work well. In particular, to train ISRT, we set the ensemble size to $M = 500$, the number of considered threshold values to $T = 8$ and the number of epochs for training selectors to $E = 1$. The same setting of parameters (i.e. $M = 500$ and $T = 8$) is used for the BLRT model, which corresponds to the setting applied in the original work of BLRT during the evaluation. The NN architecture consists of a single instance layer of size 10 with rectified linear units (ReLU), followed by a mean-max aggregation layer, a single bag layer of size 10 and two output units. The weights are regularized with L1 regularization $\lambda = 10^{-3}$ to decrease overfitting as suggested in [19]. The training minimizes a cross-entropy loss function using ADAM optimizer, mini-batch of size 8, and the maximum number of epochs 1000. Finally, the MI-SVM model is trained with the regularization $\lambda = 10^{-3}$ and 100 epochs.

Table 2 shows the performance of each model on each dataset in terms of the average Area Under the ROC Curve (AUC)[11] ± one standard deviation. It can be seen that the proposed ISRT model significantly outperforms the other three models only on Protein dataset, whereas the prior-art BLRT model significantly wins on two datasets (Newsgroup3 and Mutagenesis2). The average ranks[12] of the models are: BLRT = 1.8, ISRT = 2.0, NN = 3.0 and MISVM = 3.2. According to the non-parametric Friedman-Nemenyi test [8] (comparing all classifiers to each other based on the average ranks), there is no statistically significant difference[13] (at $\alpha = 0.05$) among the models, except for the pair BLRT and MI-SVM, where MI-SVM looses.

As the last experiment, we investigate the influence of individual model components/parameters on the final performance. Figure 3 shows results from this ablation study as a series of eight pair-wise comparisons. Each subplot compares two different variants (horizontal and vertical axis) of the proposed ISRT algorithm on the 12 datasets (dots on the scatter plot). X and Y coordinates of each dot are determined by the achieved AUCs of the corresponding variants on that particular dataset. Therefore, if a dot lies above the main diagonal, the variant associated with the vertical axis outperforms the other one associated with the horizontal axis and vice versa.

The first two **Subplots (A-B)** illustrate the effect of the ensemble size. While it is almost always better to build 100 trees than 5, building 500 trees usually does not bring any additional performance compared to 100. In **Subplot (C)**, we examine the model stability with respect to different random seeds (1234 vs. 42) and as can be seen, there is almost no difference. **Subplot (D)** shows a slight improvement that can be achieved by considering more thresholds for splitting ($T = 8$) than one as it is characteristic for Extremely Randomized Trees [11].

[11] AUC is agnostic to class imbalance and classifier's decision threshold value.

[12] The best model is assigned the lowest rank (i.e. one).

[13] The performance of any two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference, which is (for 12 datasets, four methods and $\alpha = 0.05$) approximately 1.35.

It is also worth experimenting with the sparse vs. dense selectors because, as can be observed from **Subplot (E)**, this option has different effects on different datasets. **Subplot (F)** then supports the idea that strongly randomized trees, unlike Breiman's Random Forests, do not have to be trained on the bootstrapped datasets. In **Subplot (G)** we analyze whether the search of splitting parameters $\varPhi = (f, v, \mathbf{w})$ over multiple selectors with different regularization values $\lambda \in \{10^{-4}, 10^{-3} \ldots, 1\}$ instead of one $\lambda = 1$ can help. Finally, the last **Subplot (H)** indicates that there is no need to train selectors with a large number of epochs.

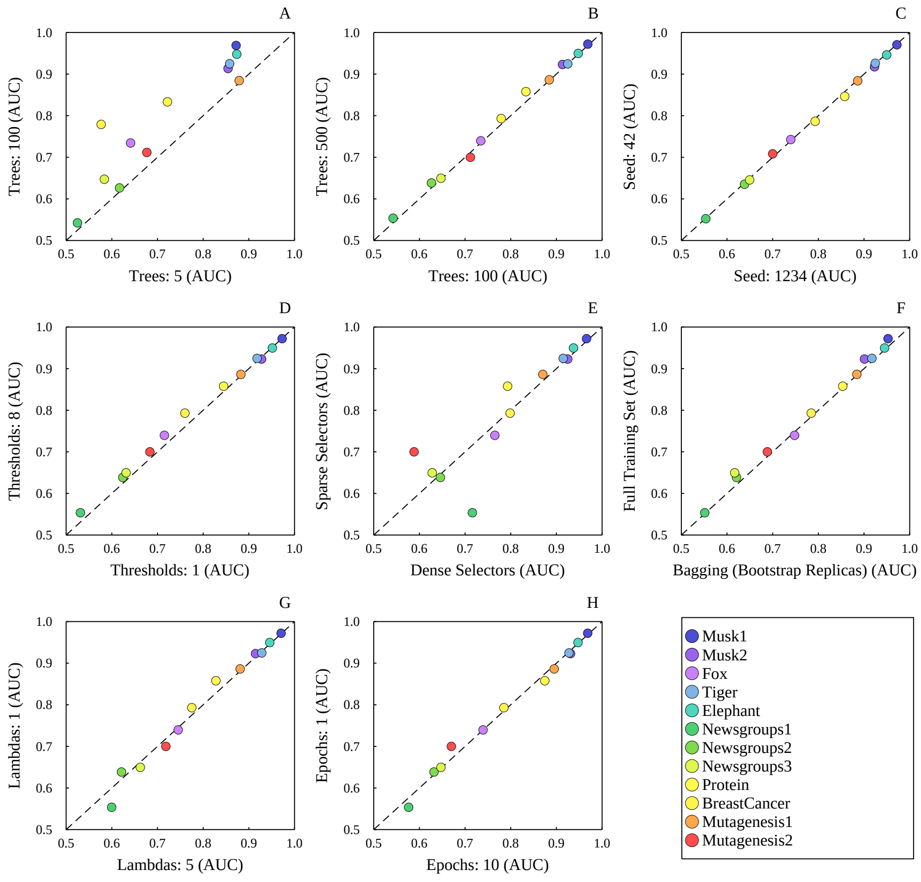

Fig. 3. Ablation study assessing influence of individual model components/parameters. It illustrates the effect of the ensemble size (A-B), the stability wrt. random seed (C), the slight improvement caused by considering more thresholds for splitting (D), the impact of sparse vs. dense selectors (E), the no need for using bagging (F), multiple regularization values (G) nor a large number of epochs for training selectors (H).

5 Conclusion

In this paper, we have proposed a new tree-based algorithm called Instance Selection Randomized Trees (ISRT)[14] for solving binary classification MIL problems. The algorithm naturally extends the traditional randomized trees, since bags of size one are processed in the standard way by evaluating single feature value conditions at each node. When bags contain multiple instances, every node selects one instance from the bag, upon which the decision whether to send the bag to the left or right branch is made. Making decisions upon deliberately selected instances at each step is essential for the algorithm as it enables to extract (even multivariate) discriminative information from both ends of the spectrum, the local instance-level and the global bag-level.

We demonstrated on the task of detecting infected users in computer networks that this capability may not only lead to superior performance when compared with three state-of-the-art methods, but it also may greatly help threat analysts with the post-alert analysis. This is because the positive bag predictions can be explained on the level of instances by their ranking according to how many times have been selected. The model also achieved competitive results on 12 publicly available datasets from six other domains. Finally, we conducted ablation experiments to understand contributions of individual algorithm parts.

References

1. Amores, J.: Multiple instance classification: Review, taxonomy and comparative study. Artif. Intell. **201**, 81–105 (Aug 2013), `http://dx.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. pp. 577–584. NIPS'02, MIT Press, Cambridge, MA, USA (2002), `http://dl.acm.org/citation.cfm?id=2968618.2968690`
3. Brabec, J., Komárek, T., Franc, V., Machlica, L.: On model evaluation under non-constant class imbalance. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science – ICCS 2020. pp. 74–87. Springer International Publishing, Cham (2020)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (Oct 2001), `http://dx.doi.org/10.1023/A:1010933404324`
5. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
6. Carbonneau, M.A., Cheplygina, V., Granger, E., Gagnon, G.: Multiple instance learning: A survey of problem characteristics and applications. Pattern Recognition **77**, 329–353 (2018), `https://www.sciencedirect.com/science/article/pii/S0031320317304065`
7. Cheplygina, V., Tax, D.M.J.: Characterizing multiple instance datasets. In: Feragen, A., Pelillo, M., Loog, M. (eds.) Similarity-Based Pattern Recognition. pp. 15–27. Springer International Publishing, Cham (2015)

[14] Source codes are available at `https://github.com/komartom/ISRT.jl`

8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (Dec 2006), `http://dl.acm.org/citation.cfm?id=1248547.1248548`
9. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence **89**(1), 31 – 71 (1997), `http://www.sciencedirect.com/science/article/pii/S0004370296000343`
10. Franc, V., Sofka, M., Bartos, K.: Learning detector of malicious network traffic from weak labels. In: Bifet, A., May, M., Zadrozny, B., Gavalda, R., Pedreschi, D., Bonchi, F., Cardoso, J., Spiliopoulou, M. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 85–99. Springer International Publishing, Cham (2015)
11. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **63**(1), 3–42 (Apr 2006), `https://doi.org/10.1007/s10994-006-6226-1`
12. Ho, T.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**, 832–844 (1998)
13. 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), `http://www.sciencedirect.com/science/article/pii/S0957417418300794`
14. Komárek, T., Somol, P.: Multiple instance learning with bag-level randomized trees. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 259–272. Springer International Publishing, Cham (2019)
15. Li, K., Chen, R., Gu, L., Liu, C., Yin, J.: A method based on statistical characteristics for detection malware requests in network traffic. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). pp. 527–532 (2018), `https://doi.org/10.1109/DSC.2018.00084`
16. Machlica, L., Bartos, K., Sofka, M.: Learning detectors of malicious web requests for intrusion detection in network traffic (2017)
17. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: TESSERACT: Eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 729–746. USENIX Association, Santa Clara, CA (Aug 2019), `https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury`
18. Pevny, T., Somol, P.: Discriminative models for multi-instance problems with tree structure. In: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. p. 83–91. AISec '16, Association for Computing Machinery, New York, NY, USA (2016), `https://doi.org/10.1145/2996758.2996761`
19. Pevný, T., Somol, P.: Using neural network formalism to solve multiple-instance problems. In: Cong, F., Leung, A., Wei, Q. (eds.) Advances in Neural Networks - ISNN 2017. pp. 135–142. Springer International Publishing, Cham (2017)
20. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (Mar 1986), `http://dx.doi.org/10.1023/A:1022643204877`
21. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: Proceedings of the 24th International Conference on Machine Learning. p. 807–814. ICML '07, Association for Computing Machinery, New York, NY, USA (2007), `https://doi.org/10.1145/1273496.1273598`
22. Stiborek, J., Pevný, T., Rehák, M.: Multiple instance learning for malware classification. Expert Systems with Applications **93**, 346 – 357 (2018), `http://www.sciencedirect.com/science/article/pii/S0957417417307170`