MIForests: Multiple-Instance Learning with Randomized Trees[⋆]

Christian Leistner, Amir Saffari, and Horst Bischof

Institute for Computer Graphics and Vision, Graz University of Technology,
Inffeldgasse 16, 8010 Graz, Austria
{leistner,saffari,bischof}@icg.tugraz.at
http://www.icg.tugraz.at

Abstract. Multiple-instance learning (MIL) allows for training classifiers from ambiguously labeled data. In computer vision, this learning paradigm has been recently used in many applications such as object classification, detection and tracking. This paper presents a novel multiple-instance learning algorithm for randomized trees called *MIForests*. Randomized trees are fast, inherently parallel and multi-class and are thus increasingly popular in computer vision. MIForest combine the advantages of these classifiers with the flexibility of multiple instance learning. In order to leverage the randomized trees for MIL, we define the hidden class labels inside target bags as random variables. These random variables are optimized by training random forests and using a fast iterative homotopy method for solving the non-convex optimization problem. Additionally, most previously proposed MIL approaches operate in batch or off-line mode and thus assume access to the entire training set. This limits their applicability in scenarios where the data arrives sequentially and in dynamic environments. We show that MIForests are not limited to off-line problems and present an on-line extension of our approach. In the experiments, we evaluate MIForests on standard visual MIL benchmark datasets where we achieve state-of-the-art results while being faster than previous approaches and being able to inherently solve multi-class problems. The on-line version of MIForests is evaluated on visual object tracking where we outperform the state-of-the-art method based on boosting.

1 Introduction

In recent years, visual object classification and detection has made significant progress. Besides novel methods for image representations, one important factor was the development and application of advanced machine learning methods. Traditional supervised learning algorithms require labeled training data where each instance (*i.e.*, data sample or feature vector) has a given label. In practice, the labels are usually provided by a human labeler. However, especially for

[⋆] This work has been supported by the Austrian FFG project MobiTrick (825840) and Outlier (820923) under the FIT-IT program.

positive classes it is often hard to label the samples so that they can be best exploited by the learning algorithm. For example, in case of object detection bounding boxes are usually cropped around the target object and provided as positive training samples. The decision where exactly to crop the object and at which size is up to the human labeler and it is often not clear if those patches are best suited for the learner. Additionally, it would also ease the labeling effort if the *exact* object location had not to be marked. By contrast, it would be desired to provide the learner only a rough position of the target object and leave it on its own how to incorporate the information in order to deliver best classification results. For standard supervised learning techniques it is hard to resolve such ambiguously labeled data. In contrast, multiple-instance learning (MIL) [1, 2] naturally can perform this task. In particular, in multiple-instance learning, training samples are provided in form of bags, where each bag consists of several instances. Labels are only provided for the bags and not for the instances. The labels of instances in positive bags are unknown whereas all instances in negative bags can be considered as being negative. For positive bags, the only constraint is that at least one of the instances is positive. Recently, multiple instance learning has enjoyed increasing popularity, especially in computer vision, because in practice data is often provided in a similar manner. Applying MIL in the above example, the rough object position would correspond to a bag and patches inside the bag to instances. During training, MIL would find those patches that lead to best classification results and leave out the others.

While multiple-instance learning has been used in many applications such as text-categorization [3], drug activity recognition [2] or computer security problems [4], especially computer vision is one of the most important domains where multiple instance-learning algorithms have been recently applied. For instance, many authors applied MIL to image retrieval [5, 6] or image categorization tasks [7]. Another computer vision application where multiple-instance learning can be used is to tackle the alignment problem when training appearance-based detectors based on boosting [8], speed-up classifier cascades [9] or even action recognition [10] and semantic segmentation [11]. In case of object tracking, it is mostly hard to decide which patches to use for updating the adaptive appearance model. If the tracker location is not precise, errors may accumulate which finally leads to drifting. Recently, Babenko *et al.* [12] demonstrated that using MIL for tracking leads to much more stable results. For most of these vision tasks SVM variants or boosting have been used.

In this paper, we present a novel multiple-instance learning algorithm based on random forests (RF) [13] [1]. The motivation for developing such an algorithm has several reasons: RFs have demonstrated to be better or at least comparable to other state-of-the-art methods in both classification [13] and clustering [14]. Caruana *et al.* [15] showed that RFs outperform most state-of-the-art learners on high dimensional data problems. Especially, the speed in both training and evaluation is one of their main appealing properties. Additionally, RFs can

[1] Note that we consider "random forests" and "randomized trees" to be the same and use the term interchangeably throughout the paper.

easily be parallelized, which makes them interesting for multi-core and GPU implementations [16]. RFs are inherently multi-class, therefore it is not necessary to build several binary classifiers for solving multi-class problems. Finally, compared to boosting and other ensemble methods, RFs are more robust against label noise [13]. These advantages of random forests have also led to increased interest in the computer vision domain. For instance, recently Gall and Lempinsky [17] presented an efficient object detection framework based on random forests. Shotton *et al.* [18] presented a real-time algorithm for semantic segmentation based on randomized trees. Bosch and Zisserman used RFs for object categorization [19]. Randomized trees have also successfully applied to visual tracking, either in batch mode using keypoints [20] or on-line using tracking-by-detection [21].

The main contribution of this work is an algorithm that extends random forests to multiple-instance learning. We thus call the method *MIForests*. MIForests bring the advantages of random forests to multiple-instance learning, where usually different methods have been applied. In turn, extending random forests in order to allow for multiple-instance learning allows vision tasks where RFs are typically applied to benefit from the flexibility of MIL. MIForests are very similar to conventional random forests. However, since the training data is provided in form of bags, during learning the real class labels of instances inside bags are unknown. In order to find the hidden class labels, we consider them as random variables defined over a space of probability distributions. We disambiguate the instance labels by iteratively searching for distributions that minimize the overall learning objective. Since this is a non-convex optimization problem, we adopt an approach based on deterministic annealing, which provides a fast solution and thus preserves the speed of random forests during training. The evaluation speed of MIForests is identical to standard random forests.

Although there have been proposed numerous approaches to the MIL problem, most of them operate in off-line or batch mode. Off-line methods assume having access to the entire training data which eases optimization and typically yields good classifiers. In practice, however, learners often have limited access to the problem domain due to dynamic environments or streaming data sources. In computer vision, this is *e.g.* the case in robot navigation or object tracking. For such problems off-line learning does not work anymore and on-line methods have to be applied. In this paper, we take this into account and show how MIForests can be extended to on-line learning.

In the experimental section, we compare MIForests with other popular MIL algorithms both on benchmark data sets and on multi-class image classification problems, where we show that MIForests can achieve state-of-the-art results without splitting multi-class problems into several binary classifiers. We evaluate the on-line extension of MIForests on object tracking and compare it to the state-of-the-art methods.

In Section 2, we present a brief overview on previous multiple-instance learning methods and RFs. In Section 3, we derive our new multiple-instance learning algorithm for random forests and present an on-line extension. Experimental results on standard visual MIL datasets, comparisons to other MIL approaches and tracking results of our approach are presented in Section 4. Finally, in Section 5, we give some conclusions and ideas for future work.

2 Related work

In traditional supervised learning training data is provided in form of $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is an instance and, in the binary case, $y_i \in \{-1, +1\}$ the corresponding label. In multiple instance learning training samples are given in bags $B_i, i = 1, \ldots, n$, where each bag may consist of an arbitrary number of instances $B_i = \{x_i^1, x_i^2, \ldots, x_i^{n_i}\}$. Negative bags B_i^- consist of only negative instances. Ambiguity is introduced into learning by the constraint that for positive bags B_i^+, it is only guaranteed that there exist at least one positive instance (also called *witness* of the bag). There is no information about other instances in the bag. In fact, they might not even belong to the negative class. The task is to learn either a bag classifier $f : B \to \{-1, 1\}$ or an instance classifier $f : \mathbb{R}^d \to \{-1, 1\}$. However, bag classification can be obtained automatically from instance classification, *e.g.*, by using the *max* operator $p_i = \max_j \{p_{ij}\}$ over posterior probability estimates p_{ij} for the j-th instance of the i-th bag.

There exists a vast amount of literature and many different approaches on how to solve the MIL problem. Here, we briefly review some of the most popular ones. The most naïve approach is to simply ignore the MIL setting and train a supervised classifier on all instances with the bag label. Blum and Kalai [22], for instance, showed that one can achieve reasonable results when training an instance classifier that is robust to class label noise. As we will show later in the experimental part, RFs are also promising candidates for such a naïve approach. Many MIL methods work by adapting supervised learners to the MIL constraints, mostly using SVM-type learners. For example, Andrews *et al.* [3] proposed two different types of SVM-MIL approaches mi-SVM and MI-SVM. They differ basically on their assumptions, *i.e.*, the first method tries to identify the labels of all instances in a bag while the latter one finds only the witness and ignores all others. Another SVM-based approach MICA [23] tries to find the witness using linear programming. There also exist some boosting-based methods, *e.g.*, [8]. Wang and Zucker [24] trained a nearest neighbor algorithm using Hausdorff distance. Other popular approaches are based on the diverse-density assumption, for example [25, 26], which more directly tries to address the MIL problem via finding a more appropriate feature representation for bags. In MILES, Chen *et al.* [7, 27] trained a supervised SVM on data mapped into a new feature space based on bag similarities. There exist also approaches for training decision trees in a MIL fashion, *e.g.*, [28].

2.1 Random Forests

Random Forests (RFs) were originally proposed by Amit *et al.* [29], extended by Breiman [13] and consist of ensembles of M independent decision trees $f_m(\mathbf{x})$:

$\mathcal{X} \to \mathcal{Y} = \{1, \dots, K\}$. For a forest $\mathcal{F} = \{f_1, \cdots, f_M\}$ the predictive confidence can be defined as $F_k(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} p_m(k|\mathbf{x})$, where $p_m(k|\mathbf{x})$ is the estimated density of class labels of the leaf of the m-th tree, where sample $\mathbf{x}$ resides. A decision is made by simply taking the maximum over all individual probabilities of the trees for a class k with $C(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x})$. [13] showed that the generalization error of random forests is upper bounded by $GE \leq \bar{\rho} \frac{1-s^2}{s^2}$, where $\bar{\rho}$ is the mean correlation between pairs of trees in the forest and s is the strength of the ensemble (*i.e.*, the expected value of the margin over the entire distribution). In order to decrease the correlation of the trees, each tree is provided with a slightly different subset of training data by subsampling with replacement from the entire training set, *a.k.a* bagging. Trees are trained recursively, where each split node randomly selects binary tests from the feature vector and selects the best according to an impurity measurement such as the entropy $H(I) = -\sum_{i=1}^{K} p_i^j \log(p_i^j)$, where p_i^j is the label density of class i in node j. The recursive training continues until a maximum depth is reached or no further information gain is possible.

3 Multiple-Instance Random Forests

In the following, we introduce a novel multiple instance learning algorithm using randomized trees called *MIForests*. MIForests deliver multi-class instance classifiers in form of $F(\mathbf{x}) : \mathcal{X} \to \mathcal{Y} = \{1, \dots, K\}$. Hence, during learning for each bag there is guaranteed that it has at least one instance from the target class but it may also consist of instances of some or all other classes $\{1, \dots, K\}$. This makes MIForests different to most previous MIL algorithms that only yield binary classifiers and require to handle a multi-class problem by a sequence of binary ones. One obvious way to design RFs capable of solving MIL tasks is to adopt MIL versions for single decision trees [28]. However, strategies developed for common decision trees are hard to apply for RFs due to the random split nature of their trees. For example, improper regularization of trees of a RF on the node level can decrease the diversity $\bar{\rho}$ among trees and thus increase the overall generalization error [13]. Thus, in order to perform multiple instance learning with random forests one has to find an optimization strategy that preserves the diversity among the trees.

We formulate multiple instance learning as an optimization procedure where the labels of the instances become the optimization variables. Therefore, the algorithm tries to uncover the true labels of the instances in an iterative manner. Given such labels, one can train a supervised classifier which then can be used to classify both instances and bags. Let $B_i, i = 1, \dots, n$ denote the i-th bag in the training set with label y_i. Each bag consists of n_i instances: $\{\mathbf{x}_i^1, \dots, \mathbf{x}_i^{n_i}\}$.

We write the objective function to optimize as

$$\begin{aligned}(\{y_i^j\}^*, F^*) =& \arg\min_{\{y_i^j\}, F(\cdot)} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \ell(F_{y_i^j}(\mathbf{x}_i^j)) \\ &\text{s.t. } \forall i : \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x}_i^j)) \geq 1.\end{aligned} \quad (1)$$

The objective in this optimization procedure is to minimize a loss function $\ell(\cdot)$ which is defined over the entire set of instances by considering the condition that at least one instance in each bag has to be from the target class. Note that $\mathbb{I}(\cdot)$ is an indicator function and $F_k(\mathbf{x})$ is the confidence of the classifier for the k-th class, *i.e.*, $F_k(\mathbf{x}) = p(k|\mathbf{x}) - \frac{1}{K}$. Often the loss function depends on the classification margin of an instance. In the case of Random Forests, the margin can be written as [13]

$$m(\mathbf{x}, y) = p(y|\mathbf{x}) - \max_{\substack{k \in \mathcal{Y} \\ k \neq y}} p(k|\mathbf{x}) = F_y(\mathbf{x}) - \max_{\substack{k \in \mathcal{Y} \\ k \neq y}} F_k(\mathbf{x}). \quad (2)$$

Note that for a correct classification $m(\mathbf{x}, y) > 0$ should hold. Overall, it can easy be seen that Eq. (1) is a non-convex optimization problem because a random forest has to be trained and simultaneously a suitable set of labels y_i^j has to be found. Due to the integer values of the labels y_i^j, this problem is a type of integer programming and is usually difficult to solve. In order to solve this non-convex optimization problem without loosing too much of the training speed of random forests, we use a fast iterative optimization procedure based on deterministic annealing (DA).

3.1 Optimization

DA [30] is a homotopy method which is able to fast minimize non-convex combinatorial optimization problems. The main idea is to extend a difficult optimization problem with an easier one by adding a convex entropy term and solve this first. In particular, one tries to minimize the entropy $\mathcal{H}$ of the distribution p in form of

$$p^* = \arg\min_{p \in \mathcal{P}} E_p(\mathcal{F}(y)) - T\mathcal{H}(p), \quad (3)$$

where $\mathcal{P}$ is a space of probability distributions and $\mathcal{F}(y)$ is our objective function. The optimization problem is than gradually deformed to its original form using a cooling parameter T, *i.e.*, $T_0 > T_1 > \ldots > T_\infty = 0$. Due to its speed and simplicity, DA-based optimization has been applied to many problems, among them also multiple-instance learning though in context with SVMs, *i.e.*, see [31]. Furthermore, due to the induced randomness in deterministic annealing, it fits to the nature of randomized trees and was recently also used for solving semi-supervised learning problems [32]. For further details on DA we refer the reader to [30].

In order to optimize our MIL objective function (Eq. (1)), we propose the following iterative strategy: In the first iteration, we train a naïve RF that ignores the MIL constraint and uses the corresponding bag labels for instances inside that bag. Then, after the first iteration, we treat the instance labels in target bags as binary variables. These random variables are defined over a space of probability distributions $\mathcal{P}$. We now search a distribution $\hat{\mathbf{p}} \in \mathcal{P}$ for each bag which solves our optimization problem in Eq. (1). Based on $\hat{\mathbf{p}}$ each tree randomly selects the instance labels for training. Hence, based on the optimization of $\hat{\mathbf{p}}$ we try to identify the real but hidden labels of instances.

We reformulate the objective function given in Eq. (1) so that it is suitable for DA optimization

$$\mathcal{L}_{DA}(F, \hat{\mathbf{p}}) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \ell(F_k(\mathbf{x}_i^j)) + T \sum_{i=1}^{n} H(\hat{\mathbf{p}}_i), \tag{4}$$

where T is the temperature parameter and

$$H(\hat{\mathbf{p}}_i) = -\sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \log(\hat{p}(k|\mathbf{x}_i^j)) \tag{5}$$

is the entropy over the predicted distribution inside a bag. It can be seen that the parameter T steers the importance between the original objective function and the entropy. If T is high, the entropy dominates the loss function and the problem can be easier solved due to the convexity. If $T = 0$ the original loss dominates (Eq. (1)). Hence, DA first solves the easy task of entropy minimization and then by continuously decreasing T from high values to zero gradually solves the original optimization problem, *i.e.*, finding the real but hidden instance labels y and simultaneously training an instance classifier.

In more detail, for a given temperature level, the learning problem can be written as

$$\begin{aligned}(F^*, \hat{\mathbf{p}}^*) =& \arg\min_{\hat{\mathbf{p}}, F(\cdot)} \mathcal{L}_{DA}(F, \hat{\mathbf{p}}) \\ \text{s.t. } \forall i : & \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x}_i^j)) \geq 1.\end{aligned} \tag{6}$$

We split this optimization problem up into a two-step convex optimization problem analog to an alternating coordinate descent approach. In the first step, the objective function $\mathcal{F}$ is optimized by fixing the distribution $\hat{\mathbf{p}}$ and optimizing the learning model. In the second step, the distribution p^* over the bags according to the current entropy level is adjusted. Note that both individual steps are convex optimization problems. For a given distribution over the bag samples, we randomly choose a label according to $\hat{\mathbf{p}}$. We repeat this process independently for every tree f in the forest. Hence, in the limit, we will exactly maintain the same distribution over the unlabeled samples as given by $\hat{\mathbf{p}}$. Let

Algorithm 1 MIForests

```
Require: Bags $\{\mathcal{B}_i\}$
Require: The size of the forest: $M$
Require: A starting heat parameter $T_0$
Require: An ending parameter $T_{min}$
Require: A cooling function $c(T, m)$
 1: Set: $\forall i : \hat{y}_i^j = y_i$
 2: Train the RF: $\mathcal{F} \leftarrow$ trainRF($\{\hat{y}_i^j\}$).
 3: Init epochs: $m = 0$.
 4: while $T_{m+1} \geq T_{min}$ do
 5:   Get the temperature: $T_{m+1} \leftarrow c(T_m, m)$.
 6:   Set $m \leftarrow m + 1$.
 7:   $\forall \mathbf{x}_i^j \in \mathcal{B}_i, k \in \mathcal{Y}$ : Compute $p^*(k|\mathbf{x}_i^j)$
 8:   for $t$ from 1 to $M$ do
 9:     $\forall \mathbf{x}_i^j \in \mathcal{B}_i$ : Select random label, $\hat{y}_i^j$ according to $p^*(\cdot|\mathbf{x}_i^j)$
10:     Set the label for instance with highest $p^*(\cdot|\mathbf{x}_i^j)$ equal to bag label
11:     Re-train the tree:
12:     $f_m \leftarrow$ trainTree($\{\hat{y}_i^j\}$).
13:   end for
14: end while
15: Output the forest $\mathcal{F}$.
```

$\{\hat{y}_{ij}\}$ be the randomly drawn labels according to the distribution $\hat{\mathbf{p}}$ for m-th tree. The optimization problem for the m-th tree becomes

$$\begin{aligned} f_m^* =& \arg\min_f \sum_{i=1}^{n} \sum_{j=1}^{n_i} \ell(f_{\hat{y}_i^j}(\mathbf{x}_i^j)) \\ & \text{s.t. } \forall i : \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} f_k(\mathbf{x}_i^j)) \geq 1. \end{aligned} \tag{7}$$

Since the margin maximizing loss function is convex, this loss function is also convex. In order to not violate the MIL constraint, after having randomly selected instance labels for a bag, we always set the instance with the highest probability according to $\hat{\mathbf{p}}$ equal to the bag label. At this stage we train all the trees in the forest by the formulation given above.

After we trained the random forest, we enter the second stage where we find the optimal distribution according to

$$\hat{\mathbf{p}}^* = \arg\min_{\hat{\mathbf{p}}} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \ell(F_k(\mathbf{x}_i^j)) + T \sum_{i=1}^{n} H(\hat{\mathbf{p}}_i). \tag{8}$$

The optimal distribution is found by taking the derivative *w.r.t* p and setting it to zero. We depict all detailed steps of the method in Algorithm 1.

3.2 On-line MIForests

MIForests as introduced above are trained off-line using a two-step optimization procedure as given in Eq. (4), where in one step the objective function $\mathcal{F}$ is optimized and in the second step the distribution $\hat{\mathbf{p}}$ over the bags, respectively. In order to modify the algorithm so that it is suitable for on-line learning, *i.e.*, the bags B_i arrive sequentially, one has to change both optimization steps to operate in on-line mode. In the following, we show how to train the randomized trees on-line in order to optimize $\mathcal{F}$ and also how $\hat{\mathbf{p}}$ can be optimized on-line to disambiguate the class labels inside positive bags.

Bagging, necessary to build the tree ensemble, can be easily done on-line by modeling the sequentially arriving samples with a Poisson distribution initialized with a constant value λ [33]. On-line learning of the decision trees is less trivial due to their recursive split nature. However, as we recently showed [21] the pure recursive training of the trees can be circumvented by using a tree-growing procedure similar to evolving-trees [34]. In more detail, the algorithm starts with trees consisting only of root nodes and randomly selected node tests f_i and thresholds θ_i. Each node estimates an impurity measure based on the Gini index ($G_i = \sum_{i=1}^{K} p_i^j (1 - p_i^j)$) on-line, where p_i^j is the label density of class i in node K. Then, after each on-line update the possible information gain ΔG during a potential node split is measured. If ΔG exceeds a given threshold β, the node becomes a split node; *i.e.*, it is not updated any more and generates two child leaf nodes. The growing proceeds until a maximum depth is reached. Even when the tree has grown to its full size, all leaf nodes are further on-line updated. The method is simple to implement and has shown to converge fast to its off-line counterpart. For further details we refer the reader to [21].

Besides on-line training of the randomized trees, we also have to perform the deterministic annealing on-line. This means we have to estimate $\hat{p}$ on-line by examining the sequentially arriving samples. Therefore, if a new bag B_i arrives, we initialize a new distribution $\hat{p_i}$ over its instances using the current confidence output of $\mathcal{F}_t$. Then, we iteratively apply the optimization of $\mathcal{F}_t$ and $\hat{p_i}$ only for the current bag B_i following the same two-step procedure and annealing schedule as in the off-line case (Eq. (7),Eq. (8)). Afterwards, B_i is discarded and the training proceeds with the next bag B_{i+1}. We skip the algorithm box due to lack of space.

4 Experiments

The purpose of this section is to evaluate the proposed algorithms on standard MIL machine learning benchmark datasets and to demonstrate their performance on typical computer vision problems such as object tracking. Note that, in general, we abstain from any data set or feature engineering procedures, since the main purpose is to compare the different learning methods.

4.1 Benchmark Datasets

We first evaluate our proposed MIForests on popular benchmark datasets used in most studies of multiple-instance learning algorithms, *i.e.*, the *Musk1 and Musk2* drug activity datasets proposed by Dietterich [2] and the *Tiger, Elephant and Fox* image datasets proposed by Andrews *et al.* [3][2]. For sanity check we also tested common random forests [13], *i.e.*, ignoring the MIL constraint. For all learners we used 50 trees with a maximum depth of 20. As cooling schedule we used a simple exponential function in form of $T_t = e^{-t \cdot C}$, where t is the current iteration and the constant $C = \frac{1}{2}$. We determined these settings empirically and kept them fixed over all experiments.

As can be observed, the performance of the individual approaches varies highly depending on the data set. The experiments show that MIForests achieve state-of-the-art performance and are even outperforming several SVM-based approaches and those based on boosting. Especially for the vision problems, we are always among the best. Also the naïve RF approach yields surprisingly good performance, especially on *Fox* and *Musk1*; however, it cannot take pace with the performance of its MIForest counterpart. One explanation for this might be that RFs are less susceptible to noise compared to other learning methods, which is necessary for the naïve approach [22]. Compared to its most similar SVM variant (AL-SVM), MIForest outperforms it on two datasets, draws on one and performs worse on two. Finally, it has to be mentioned that especially for [31] and [35] better results can be achieved by incorporating prior knowledge into the learners, *e.g.*, how many “real” positives exist inside bags; which however also holds for MIForests.

Method	Elephant	Fox	Tiger	Musk1	Musk2
RandomForest [13]	74	60	77	85	78
MIForest	**84**	**64**	82	85	82
MI-Kernel [3]	**84**	60	**84**	**88**	**89**
MI-SVM [36]	81	59	**84**	78	84
mi-SVM [36]	82	58	79	87	84
MILES [7]	81	62	80	**88**	83
SIL-SVM	**85**	53	77	**88**	**87**
AW-SVM [31]	82	**64**	83	86	84
AL-SVM [31]	79	63	78	86	83
EM-DD [26]	78	56	72	85	85
MILBoost-NOR [8]	73	58	56	71	61

Table 1. Results and comparisons in terms of percent classification accuracy on popular MIL benchmark datasets. We report the average over 5 runs. Best methods with the error margin are marked in bold face.

[2] Sample C++ code is available at `http://www.ymer.org/amir/software/milforests`

4.2 Corel Dataset

Here, we evaluate our proposed methods on the Corel-1000 and Corel-2000 image dataset for region-based image classification. The data set consists of 2000 images with 20 different categories. Each image is a bag consisting of instances obtained via oversegmentation. It is thus a typical MIL problem. In order to allow for fair comparison we used the same data settings and features as proposed by Chen *et al.* [7]. For the results we used the same settings as in our previous experiments. In contrast to most other approaches, we did not train 20 1-vs.-all classifiers, but trained one multi-class forest, which is usually a more difficult task. We compare MIForests with MILES, the original algorithm proposed on this data set [7]. Since MILES is a binary algorithm we trained 20 1-vs.-all MILES classifiers and depict the results in Table 2. As can be seen, MIForests achieve completive results for multi-class scenarios, however, being much faster. We measured the average time on a standard Core Duo machine with 2.4 Ghz.

Method	Corel-1000	Corel-2000	1000 Images$_{[sec.]}$	2000 Images$_{[sec.]}$
MIForest	59	66	4.6	22.0
MILES	58	67	180	960

Table 2. Results and comparisons on the COREL image categorization benchmark. Additionally, we depict the training times in seconds.

4.3 Object Tracking

A recent dominating trend in tracking called "tracking by detection" has shown to deliver excellent results at real-time speeds. In these methods, usually an appearance-based classifier is trained with a marked object at the first frame versus its local background [37]. The object is then tracked by performing re-detection in the succeeding frames. In order to handle rapid appearance and illumination changes, recent works, *e.g.*, [38], use on-line self-updating of the classifiers. However, during this process it is not clear where to select the positive and negative updates necessary for self-updating. If the samples are selected wrongly, slight errors can accumulate over time (*a.k.a* label jitter) and cause drifting. Recently, Babenko *et al.* [12] demonstrated that label jitter can be handled by formulating the update process using an on-line MIL boosting algorithm. Using MIL, the allowed positive update area around the current tracker can be increased and the classifier resolves the ambiguities by itself, yielding more robust tracking results. See [12] for a more detailed discussion about the usefulness of MIL for tracking. In the following, we demonstrate that on-line MIForests can also give excellent tracking results, outperforming the state-of-the-art tracker based on boosting.

We focus on tracking arbitrary objects; so there is no prior knowledge about the object class available except its initial position. We use eight publicly available sequences including variations in illumination, pose, scale, rotation and appearance, and partial occlusions. The sequences *Sylvester* and *David* are taken from [39] and *Face Occlusion 1* is taken from [40], respectively. *Face occlusion 2*, *Girl*, *Tiger1,Tiger2* and *Coke* are taken from [12]. All video frames are gray scale and of size 320×240. To show the real accuracy of the compared tracking methods, we use the overlap-criterion of the VOC Challenge [41], which is defined as $A_{overlap} = R_T \cap R_{GT} / R_T \cup R_{GT}$, where R_T is the tracking rectangle and R_{GT} the groundtruth. Since we are interested in the alignment accuracy of our tracker and the tracked object, rather than just computing the raw distance we measure the accuracy of a tracker by computing the average detection score for the entire video. Note that values between 0.5 and 0.7 are usually acceptable results, values larger than 0.7 can be considered as almost perfect.

The main purpose of the tracking experiments is the comparison of the influence of the different on-line learning methods. Hence, we use simple Haar-like features for representation, did not implement any rotation or scale search and avoid any other engineering methods, although these things would definitely improve the overall results. For MIForests, we used 50 trees with depth 10 and the same annealing schedule as in the ML experiments. Overall, we generate 500 features randomly. As [12] for all boosting methods, we used 50 selectors with each 250 weak classifiers which results in a featurepool of size 12500.

In Table 3 we depict detailed results for all tracking sequences compared to MILBoost [12], SemiBoost (OSB) [42], on-line AdaBoost (OAB)[38] and on-line random forests (ORF) [21]. As can be seen, MIForests perform best on seven tracking sequences. Remarkably, we are able to outperform MILBoost, which is currently known to be amongst the best tracking methods, on 6 out of 8 sequences, draw on 1 and are slightly worse on 1. The resulting tracking videos can be found in the supplementary material.

Method	*sylv*	*david*	*faceocc2*	*tiger1*	*tiger2*	*coke*	*faceocc1*	*girl*
MIForest	0.59	**0.72**	**0.77**	**0.55**	**0.53**	**0.35**	**0.77**	**0.71**
MILBoost	**0.60**	0.57	0.65	0.49	**0.53**	0.33	0.60	0.53
OSB	0.46	0.31	0.63	0.17	0.08	0.08	0.71	0.69
OAB	0.50	0.32	0.64	0.27	0.25	0.25	0.47	0.38
ORF	0.53	0.69	0.72	0.38	0.43	**0.35**	0.71	0.70

Table 3. Tracking results on the benchmark sequences measured as average detection window and ground truth overlap over 5 runs per sequence. Best performing method is marked in bold face.

5 Conclusion

In this paper, we presented a new multiple-instance learning method based on randomized trees (MILForest). We define the labels of instances inside positive bags as random variables and use a deterministic-annealing style procedure in order to find the true but hidden labels of the samples. In order to account for the increasing number of data and leverage the usage of our method in streaming data scenarios, we also showed how to extend MILForests for on-line learning. We demonstrated that MILForests are competitive to other methods on standard visual MIL benchmark datasets while being faster and inherently multi-class. We demonstrated the usability of the on-line extension on the task of visual object tracking where we outperformed state-of-the-art methods. In future work, we plan to test our algorithm on other vision applications such as object detection and categorization.

References

1. Keeler, J., Rumelhart, D., Leow, W.: Integrated segmentation and recognition of hand-printed numerals. In: NIPS. (1990)
2. Dietterich, T., Lathrop, R., Lozano-Perez, T.: Solving the multiple-instance problem with axis-paralle rectangles. In: Artifical Intelligence. (1997)
3. Andrews, S., Tsochandaridis, I., Hofman, T.: Support vector machines for multiple-instance learning. Adv. in Neural. Inf. Process. Syst. **15** (2003) 561–568
4. Ruffo, G.: Learning single and multiple instance decision trees for computer security applications. PhD thesis (2000)
5. Zhang, M.L., Goldman, S.: Em-dd: An improved multi-instance learning technique. In: NIPS. (2002)
6. Vijayanarasimhan S. Grauman, K.: Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In: CVPR. (2008)
7. Chen, Y., Bi, J., Wang, J.: Miles: Multiple-instance learning via embedded instance selection. In: IEEE PAMI. (2006)
8. Viola, P., Platt, J., Zhang, C.: Multiple instance boosting for object detection. In: NIPS. (2006)
9. Zhang, C., Viola, P.: Multiple-instance pruning for learning efficient cascade detectors. In: NIPS. (2008)
10. : Activity recognition from sparsely labeled data using multi-instance learning. In: LoCA. (2009)
11. Vezhnevets, A., Buhmann, J.: Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In: CVPR. (2010)
12. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR. (2009)
13. Breiman, L.: Random forests. Machine Learning (2001)
14. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS. (2006) 985–992
15. Caruana, R., Karampatziakis, N., Yessenalina, A.: An empirical evaluation of supervised learning in high dimensions. In: ICML. (2008)
16. Sharp, T.: Implementing decision trees and forests on a gpu. In: ECCV. (2008)

17. Gall, J., Lempinsky, V.: Class-specific hough forests for object detection. In: CVPR. (2009)
18. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image catergorization and segmentation. In: CVPR. (2008)
19. Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: ICCV. (2007)
20. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. In: CVPR. (2006)
21. Saffari, A., Leistner, C., Godec, M., Santner, J., Bischof, H.: On-line random forests. In: OLCV. (2009)
22. Blum, A., Kalai, A.: A note on learning from multiple instance examples. Machine Learning (1998) 23–29
23. Mangasarian, O., Wild, E.: Multiple-instance learning via successive linear programming. Technical report (2005)
24. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML. (2000)
25. Maron, O., Lozano-Perez, T.: A framework for multiple-instance learning. In: NIPS. (1997)
26. Zhang, Q., Goldman, S.: Em-dd: An improved multiple instance learning technique. In: NIPS. (2001)
27. Foulds, J., Frank, E.: Revisiting multi-instance learning via embedded instance selection. Lecture Notes in Computer Science (2008)
28. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: ICML. (2005)
29. Geman, Y.A.D.: Shape quantization and recognition with randomized trees. Neural Computation (1996)
30. Rose, K.: Deterministic annealing, constrained clustering, and optimization. In: IJCNN. (1998)
31. Gehler, P., Chapelle, O.: Deterministic annealing for multiple-instance learning. In: AISTATS. (2007)
32. Leistner, C., Saffari, A., Santner, J., Bischof, H.: Semi-supervised random forests. In: ICCV. (2009)
33. Oza, N., Russell, S.: Online bagging and boosting. In: Proceedings Artificial Intelligence and Statistics. (2001) 105–112
34. Pakkanen, J., Iivarinen, J., Oja, E.: The evolving tree—a novel self-organizing network for data analysis. Neural Process. Lett. **20** (2004) 199–211
35. Bunescu, R., Mooney, R.: Multiple instance learning for sparse positive bags. In: ICML. (2007)
36. Zhou, Z.H., Sun, Y.Y., Li, Y.F.: Multi-instance learning by treating instances as non-i.i.d. samples. In: ICML. (2009)
37. Avidan, S.: Ensemble tracking. In: CVPR. Volume 2. (2005) 494–501
38. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR. (2006)
39. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. IJCV (2008)
40. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR. (2006)
41. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object class challenge 2007. In: VOC. (2007)
42. Grabner, H., Leistner, C., Bischof, H.: On-line semi-supervised boosting for robust tracking. In: ECCV. (2008)