Multiple instance learning with response-optimized random forests

Christoph Straehle, Melih Kandemir, Ullrich Koethe, Fred A. Hamprecht

University of Heidelberg, HCI/IWR
{christoph.straehle, melih.kandemir, ullrich.koethe, fred.hamprecht}@iwr.uni-heidelberg.de

***Abstract*—We introduce a multiple instance learning algorithm based on randomized decision trees. Our model extends an existing algorithm by Blockeel et al. [2] in several ways: 1) We learn a random forest instead of a single tree. 2) We construct the trees by splits based on non-linear boundaries on multiple features at a time. 3) We learn an optimal way of combining the decisions of multiple trees under the multiple instance constraints (i.e. positive bags have at least one positive instance, negative bags have only negative instances). Experiments on the typical benchmark data sets show that this model's prediction performance is clearly better than earlier tree based methods, and is comparable to the global state-of-the-art.**

I. Introduction

Dramatic improvements in instrumentation, and online data collection on an unprecedented scale, result in a tremendous increase in the number of observations. In contrast, the rate with which a human expert can annotate such observations for the purpose of supervised machine learning remains limited. It is hence attractive to trade human effort for computational expense, by learning from *weak supervision*. In brief, learning highly descriptive models from weak supervision appears as a key challenge in next generation data analysis.

Multiple instance learning (MIL) [8] is a supervised learning technique that relies on very weak supervision. As opposed to standard supervised learning settings where a ground-truth label is available for each instance, in the MIL setting, each group of instances, called *bags*, are assigned a label, and the labels of individual instances are latent. A positive bag label indicates that there exists at least one instance in that bag with positive label. In a bag with a negative label, all instances are known to have a negative label.

More formally, in the MIL setting, data can be described as a collection of bags $B^b, b \in \{1, ..., N\}$. Each bag B^b consists of a number of instances $\boldsymbol{x}_i^b \in B^b, i \in \{1, ..., N^b\}$. A bag B^b has a positive label $Y^b = 1$ if $\exists \boldsymbol{x}_i^b \in B^b$ with $y_i^b = 1$. This is called the *positive identifiability* constraint [13]. A bag B^b has a negative label $Y^b = 0$ if $y_i^b = 0 \ \forall \boldsymbol{x}_i^b \in B^b$. This is called the *negative exclusion* constraint [13]. An instance of a positive bag is called a *witness* if its unobserved ground-truth label is positive, and *non-witness* otherwise. The main challenge in MIL is to identify witnesses and non-witnesses within positive bags, and learn a decision boundary that obeys these latent class assignments.

In this paper, we introduce a decision tree based solution to MIL. Our model inherits the desirable properties of decision trees such as small computational footprint, ease of implementation, and high interpretability. In particular, we extend the work of Blockeel et al. on multi-instance decision trees [2] in several aspects. First, we learn multiple randomized decision trees [3]. Secondly, we employ a non-linear split criterion on multiple features at a time, in place of the commonplace axis-orthogonal approach. Thirdly, we increase the robustness of our model against noise by regularizing instance weights. Lastly and importantly, we introduce a means to learn the optimal combination of the decision outputs of all trees in the forest. In a lesion study, we analyze the contribution of each of these extensions to overall performance. The combined model clearly outperforms existing decision tree based methods [12], [2].

II. Related Work

MIL models differ from each other in the heuristic they employ on how to identify positive instances in the positive bags. In [8], Dietterich et al. rely on the heuristic that the positive instances reside in a single axis parallel rectangle (APR). Another seminal method is diverse density modeling [14], [24] which assumes that positive instances come from a Gaussian distribution. The learning task consists of which representative instance to select from each bag for a better fit to a Gaussian.

There also exist extensions of the k-nearest neighbor idea to the MIL setting, such as [22]. It is also possible to apply boosting to MIL, as in Viola et al. [20]. In addition, some deterministic annealing type approaches have been proposed [10], [12] that try to uncover the instance labels in several training iterations.

A large group of existing MIL methods are extensions of kernelized classifiers [1], [10], [6]. These models are based on extensions of the support vector machine (SVM) optimization problem with the positive identifiability and negative exclusion constraints. MI-SVM [1] adds one constraint per one instance in a positive bag that has the largest discriminant value. mi-SVM [1] treats the instance labels in positive bags as latent variables to be learned from data. An alternative kernel-based MIL approach is GPMIL [11], which extends Gaussian process classification to the MIL setting by modifying the sigmoid likelihood from instance level to bag level.

Another alternative approach is decision tree based MIL. Blockeel et al. [2] adapted decision trees to the MIL problem by introducing a priority queue into the tree construction

process. Leistner et al. [12] propose a deterministic annealing procedure for uncovering the hidden instance labels in several forest training iterations. In this paper, we introduce another MIL algorithm on this track, which is an extended version of of the work of Blockeel et al. [2].

Recent examplary applications of MIL include diabetic retinopathy screening [17], cancer detection from tissue images [23], and content-based object detection and tracking [19].

III. Decision trees

Let $\mathcal{D} = \{(\boldsymbol{x}_1, y_i), (\boldsymbol{x}_2, y_2) \cdots, (\boldsymbol{x}_N, y_N)\}$ be a data set of N instances where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]$ are D-dimensional vectors of observed instances and y_i are associated labels. Suppose that $M(\boldsymbol{y}_s)$ is a scalar measure based on the labels of an instance set s, denoted by $\boldsymbol{y}_s$. A decision tree is built by the following steps:

- For each observed value x_{ij} of each feature j, group the instances into two groups according to the split rule $f_j > x_{ij}$ where f_j is a threshold for feature j. Calculate a goodness measure for the split $\theta_{ij} = M(\boldsymbol{y}_{f_j > x_{ij}}) + M(\boldsymbol{y}_{f_j \leq x_{ij}})$.
- Create a node for the feature $\hat{j}$ which gives the highest θ_{ij}, and two child nodes, and assign all instances with $f_j > x_{ij}$ to one node, and the rest to the other node.
- For each child node, repeat this process on their assigned set of instances recursively until all nodes have instances belonging to a single class.

Let f_c denote the proportion of instances in $\boldsymbol{y}_s$ that belong to class c. Two widely used examples of goodness measures are *Gini Impurity*:

$$I_G = 1 - \sum_{c=1}^{C} f_c^2,$$

and *Information Gain*:

$$I_E = -\sum_{c=1}^{C} f_c \log f_c.$$

Gini impurity is widely used by the CART (Classification And Regression Tree) algorithm [4], and the information gain is more often preferred with the C4.5 algorithm [18]. The outlined axis-orthogonal splitting procedure can be replaced by more complex splitting tests over multiple features at a time, see for example [15].

IV. Random forests

Decision tree learning has several desirable properties such as being computationally very fast, and being very interpretable. However, the main drawback of this algorithm is the suboptimal prediction performance it provides. One reason for the low accuracy of decision trees is that features are handled one-by-one in the decision process, hence complex correlations between features are not captured sufficiently. Another reason is that the goodness measures are calculated over the entire set of available instances, which makes the algorithm prone to overfitting.

Breiman shows that a simple extension to the decision tree algorithm, called the *random forest*, indeed brings a significant improvement in prediction performance, while keeping the other good properties of decision trees [3]. The random forest takes a decision tree as a weak classifier, and performs ensemble learning together with a bagging procedure. In particular, the algorithm learns multiple decision trees on randomly chosen features based on randomly chosen sets of training instances. The decision for a newly seen instance is made by combining the outcomes of the learned trees in the forest.

V. Multi-instance tree learning

As a basis for our method we rely on the *multi-instance tree learning* (MITI) algorithm presented in [2]. This greedy decision tree algorithm initially assumes that each instance in a positive bag has a positive label. Importantly, the algorithm from [2] does not follow the depth first recursive partitioning scheme outlined in Section III. Instead, the authors propose a priority queue based node expansion. The algorithm maintains a priority queue of split nodes, and iteratively takes the node with the highest priority from the queue. The node that was taken from the queue is split according to the goodness measure, and the resulting two child nodes are inserted into the priority queue. The algorithm uses the number of positive instances in a node as the priority for the queue.

Furthermore, the authors assign to each instance $\mathbf{x}_i^b$ a weight w_i^b. At the beginning this weight is set to $\frac{1}{|B_b|}$, the inverse of the size of the bag b to which instance i belongs. This ensures that the weights of all the instances in a bag sum up to 1 - otherwise large bags would cause a bias. The instance weight is used throughout the entire tree construction process and is also considered during the evaluation of the goodness of a split. Thus, a weighted Gini-impurity measure for a set $\mathcal{S}$ of instances $\mathbf{x}_i^b$ is

$$G(\mathcal{S}) = \left(\sum_{\mathbf{x}_i^b \in \mathcal{S}} w_i^b\right)^2 - \left(\sum_{\mathbf{x}_i^b \in \mathcal{S}+} w_i^b\right)^2 - \left(\sum_{\mathbf{x}_i^b \in \mathcal{S}-} w_i^b\right)^2,$$

where $\mathcal{S}+$ denotes the subset of instances in $\mathcal{S}$ with positive labels, and $\mathcal{S}-$ denotes the ones with negative labels.

In addition the authors propose to change the assigned weight of the instances once a purely positive leaf has been discovered. In this case the other instances of the bags that are covered by the leaf will be discounted by setting their weight to 0. More formally, once a positive leaf Π_k has been found during tree construction we set $w_i^b = 0, \mathbf{x}_i^b \in B^b, i \neq j, \mathbf{x}_j^b \in \Pi_k$. The intuition here is that once a positive bag is explained by one or more positive instances in a purely positive leaf node, the decision tree does not need to find another positive instance to explain the positive label of the associated bag, thus the weight w_i^b of the other instances of the bags that are covered by the leaf node can be set to 0. This essentially excludes these instances from the further tree growing process, since these presumed negative instances have no influence on the weighted Gini impurity during split evaluation.

The overall process amounts to building a decision tree in a special order, where the largest positive subset in the decision tree is expanded first. This prioritization induces the tree learner to focus on finding pure positive subsets. In other words, it enforces the rule that it is legal for non-witness positive instances to end up in negative nodes, but it is not

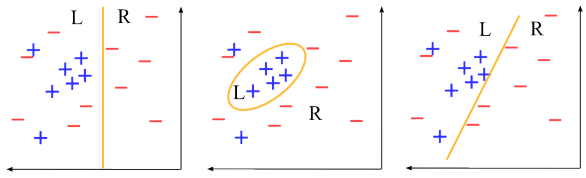

Fig. 1: Overview of the different decision tree split types. The standard axis-orthogonal split type (left) is less discriminative then the oblique hyperplane split (right) and the ellipsoid split (middle). The ellipsoid split is defined by the contour line of a normal distribution fitted to the positive instances.

desirable for negative instances to end up in positive nodes. Thus trying to first find pure positive subsets best expresses the multiple instance constraint. In [2] the authors show that the resulting tree growing procedure outperforms other decision tree based algorithms in MIL problems.

In the following sections we extend the MITI algorithm in several ways and show that this approach yields an algorithm that outperforms the existing decision tree based MIL algorithms, and gives results similar to other existing algorithms. Our extensions to MITI are:

- learning a random forest instead of a single deterministic decision tree,
- using a regularization and a weight redistribution to increase robustness to noise (as detailed in V-A),
- using a non-linear splitting method on pairs of features, instead of line search in individual features (as detailed in V-B),
- learning how to combine the decision outputs of the trees in the forest from data, subject to a multiple instance constraint (as detailed in V-D).

A. Tree regularization and weight redistribution

As opposed to [2] who grow a decision tree until impurity, we use a minimum leaf node size as the stopping criterion for the decision tree growing process. This early stopping acts as a regularizer, since it allows leaf nodes to contain positive and negative instances and the positive instances in such impure leaves are assumed to be false positives. In this case we redistribute the weights of the positive instances in an impure leaf node to the other instances of the corresponding bag which are not yet assigned to a leaf node. More formally, for an impure leaf node Π_k that is below a size threshold we set $w_i^b = w_i^b + \frac{1}{|B^b|}, \mathbf{x}_i^b \in B^b, Y^b = 1, i \neq j, \mathbf{x}_j^b \in \Pi_k$ where $|B^b|$ is the size of bag B^b. Thus, these remaining positive instances of a bag B^b contribute more strongly to the further decision tree building process and the reweighted positive instances are more likely to end up in a positive leaf node.

B. Inside/outside split concepts

In typical multiple instance learning tasks, positive instances are often distributed around few cluster centers while the negative class is distributed more evenly. This fact is exploited in many multiple instance learning algorithms, such as [14], [24], [6]. The orthogonal axis splits used in traditional decision trees, such as in [2], appear as a bottleneck in model performance since they assume an axis-orthogonal boundary between classes.

As a solution to this problem, we strenghten the modeling power of our decision trees with a more complex split scheme. Instead of the line-search based splits on single features used in [2], we employ two types of split criteria on multiple features at a time (see Figure 1):

- Ellipsoid splits
- Hyperplane splits

These complex split criteria are inspired from the oblique random forest idea studied in [16], and the Gaussian density estimation forests in [7]. In the case of *ellipsoid splits*, we pick a random subset of the feature dimensions and calculate the weighted mean and weighted sample covariance matrix of the positive samples that are assigned to that split node N:

$$\boldsymbol{\mu} = \sum_{\mathbf{x}_i^b \in \mathbf{X}_+} \frac{w_i^b \mathbf{x}_i^b}{W},$$

where $W = \sum_{\mathbf{x}_i^b \in \mathbf{X}_+} w_i^b$, and

$$\Sigma_{kl} = \left[\frac{\sum_{\mathbf{x}_i^b \in \mathbf{X}_+} w_i^b}{\left(\sum_{\mathbf{x}_i^b \in \mathbf{X}_+} w_i^b\right)^2 - \sum_{\mathbf{x}_i^b \in \mathbf{X}_+} (w_i^b)^2} \right] \times \sum_{\mathbf{x}_i^b \in \mathbf{X}_+} w_{ij} (x_{ik}^b - \mu_k)(x_{il}^b - \mu_l)$$

where $\mathbf{X}_+$ is the set of positive instances, $\mathbf{x}_i^b$ is the instance i of bag b, and x_{ik}^b is the kth feature of the same instance. We then sort the instances with respect to their Mahalanobis distance

$$\lambda = \sqrt{(\mathbf{x}_i^b - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i^b - \boldsymbol{\mu})}$$

and search for the optimal split threshold $\lambda > \tau$ with respect to the Gini impurity of the induced partitioning into a left and right subset.

In the case of *hyperplane splits*, we choose a random subset of the feature dimensions and assign a random weight to each dimension, afterwards we search for the optimal linear intercept with respect to the Gini impurity. This split type allows to discriminate better on correlated features than the axis orthogonal split type used in [2].

C. Forest vote bias correction

Our model learns multiple decision trees, each on a random subset of training instances and features. Each of these trees have their own decision output for a newly seen instance. Hence, a mechanism is required to combine these decisions to obtain the final outcome for that instance.

In the traditional random forest formulation, this issue is solved simply by majority voting. In the case of multiple instance decision trees this can have detrimental effects: since each tree deactivates all positive instances of a positive bag when it discovered a positive witness for that bag the decision

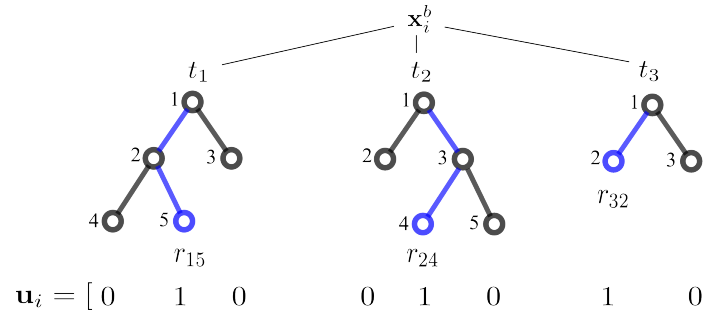

Fig. 2: The classifier response $f(\boldsymbol{x_i^b}) = \boldsymbol{u}_i^T \boldsymbol{r}$ for the instances is calculated as the scalar product between leaf weights $\boldsymbol{r}$ and the indicator vector $\boldsymbol{u}_i$ for instance $\boldsymbol{x}_i^b$. The indicator vector has 1 entries for the leaf nodes in which that instance $\mathbf{x}_i^b$ ends up when predicting the instance with the individual trees t_k. The leaf node weights $\boldsymbol{r}$ are then optimized in a post processing step subject to positive identifiability and negative exclusion constraints.

boundary is biased in favor of the negative instances since instances of the negative bags are never deactivated.

Another undesirable situation occurs when the true positive instances are arranged in several distinct clusters. Due to the deactivation mechanism, each tree can only identify a single positive cluster per positive bag. The other clusters have a certain probability of being misclassified by a single tree. Once again, true positive samples may not receive sufficiently many positive votes to reach the majority vote threshold. Consequently, a simple majority vote combination of the individual trees will lead to a strong negative bias of the forest.

We propose to counter the negative class bias of the forest in a simple way by optimizing a threshold with regard to the MI-constraint. We count the number of positive tree votes for each instance, and the instance that receives the most positive votes from each bag is recorded. For these most positive instances the vote count threshold that best separates the positive from the negative bags is determined by a line search with respect to the induced bag accuracy. We experimentally show that countering the negative bias of the forest is crucial for good performance (see section VI).

D. Forest response optimization

Threshold optimization counters the class imbalance bias (which arises from deactivation of positive training samples) by lowering the *number* of positive votes an instance must receive in order to be assigned to the positive class. This strategy does not alter the votes themselves, but only their interpretation. Alternatively (or in addition), we can improve voting performance by optimizing the *response* of all leaf nodes. This optimization is only beneficial if it is conducted *jointly* over all trees in the ensemble, because the standard leaf response (the majority label of the instances assigned to each leaf) is clearly optimal as long as each tree is considered in isolation. Since we want to retain the property that the ensemble response is computed as a linear combination of tree responses, this leads us to a constrained linear optimization problem similar to the ones discussed in [9], [15]. These works introduce a sparsity constraint in order to prune as many nodes as possible without degrading performance. In contrast, we attempt to optimize leaf responses under the multiple instance constraints.

To define the linear system, we introduce the *indicator vector* $\boldsymbol{u}_i$ of an instance i. It is a binary vector whose length equals the total number of leaf nodes in the forest, and whose elements represent the membership of i to the different leaves. That is, an element of $\boldsymbol{u}_i$ takes a value of 1 precisely when the instance i is assigned to the corresponding leaf. For example, the indicator vector for the ensemble in Figure 2 has length 8. When the feature vector $\boldsymbol{x}_i^b$ is propagated down each tree, the leaves (1,5), (2,4), and (3,2) marked in blue are reached, and the three corresponding entries in the indicator vector are set to 1. In addition, we define the leaf response vector by $\boldsymbol{r} = [r_{11}, r_{12}, \cdots, r_{TL}]$ (where r_{tl} is the response of leaf tl). Then, the total response of the ensemble can be written as the scalar product

$$f(\boldsymbol{x_i^b}) = \boldsymbol{u}_i^T \boldsymbol{r}$$

To optimize the weights $\boldsymbol{r}$, we take the indicator vectors of all training examples and stack them next to each other into the indicator matrix $\boldsymbol{U}$. The training response is thus $\boldsymbol{f} = \boldsymbol{U}^T \boldsymbol{r}$. Note that all instances are assigned to all trees during global optimization, irrespective of whether they where out-of-bag or de-activated in the tree construction phase. We now seek the vector $\boldsymbol{r}^*$ that minimizes the training loss under the multiple instance constraints. That is, f_i should take the value -1 for all instances from negative bags, and 1 for the witness of each positive bag, whereas the response for non-witness members of positive bags is ignored. Under our linear model, the witness is defined as the positive bag member that receives the maximum response. This leads to the following formal definition of the loss

$$L_{\boldsymbol{r}}^b(\mathbf{X}^b, Y^b) = \begin{cases} (\max_{\mathbf{x}_i^b \in \mathbf{X}^b}[f(\mathbf{x}_i^b)] - 1)^2 & , Y^b = 1, \\ \sum_{\mathbf{x}_i^b \in \mathbf{X}^b}(f(\mathbf{x}_i^b) + 1)^2 & , Y^b = 0. \end{cases}$$

This loss is combined with a quadratic regularizer to obtain the global optimization problem

$$\boldsymbol{r}^* = \text{argmin}_{\boldsymbol{r}}\, \boldsymbol{r}^T \boldsymbol{r} + \sum_{b \in \mathcal{B}} L_{\boldsymbol{r}}^b(\mathbf{X}^b, Y^b).$$

While this is a non-convex optimization problem (the arguments of the square functions in the loss can have arbitrary sign), we found in practice that good local optima can be found using gradient descent by initializing the leaf weights $\boldsymbol{r}$ with

the fraction of positive instances assigned to them:

$$r_l = \frac{\sum_{\boldsymbol{x}_i^b \in \Pi_l, y_i^b = 1} 1}{\sum_{\boldsymbol{x}_i^b \in \Pi_l} 1}$$

The resulting optimization algorithm is efficient and takes only a small fraction of the total training time.

VI. Experiments

We evaluated the generalization performance of our algorithm on the following five public MIL benchmarking data sets: MUSK 1, MUSK2, Elephant, Tiger, and Fox. In MUSK 1 and MUSK 2, the task is to find a configuration of a molecule that binds its target. Each molecule is formulated as a bag, and its each configuration an instance. The MUSK1 data set consists of 476 instances of 166 dimensions grouped in 47 positive and 45 negative bags. The MUSK 2 data sets has 6598 instances of 166 dimensions, and 39 positive and 63 negative bags. In Elephant, Fox, and Tiger data sets, the task is image classification, the animal in the scene determining the image class. Each image is considered a bag and each patch of an image is treated as an instance. All three data sets consist of 230-dimensional feature vectors grouped into 100 positive and 100 negative bags. The Elephant, Fox, and Tiger data sets include 1391, 1320, and 1220 instances, respectively.

The randomized decision tree parameter *mtry* which determines how many different splits are tested in each split node during tree construction was set to the standard square root of number of feature dimensions proposed by Breiman [3]. The number of dimensions for the hyperplane splits was set to 5 and the number of dimensions for the Gaussian ellipsoid split was set to 2. These choices ensure that both split types have the same number of free parameters (a 2D Gaussian has 3 free covariance matrix entries and 2 free mean vector entries). Additionally, we set the minimum leaf node size regularization parameter to the number of free parameters in the split nodes, i.e. 5. All performance scores reported below are obtained by averaging 5 runs of 10-fold cross-validation.

A. Influence of proposed extensions

As seen in Table I, all extensions except constraining the minimum leaf node size contribute to prediction accuracy, and the highest performance is achieved when all the extensions are employed together. Most pronounced is the influence of combining multiple decision trees into a forest, which is indicated by the low accuracy of the single tree model. The difference of our single tree accuracy in comparison to the MITI method can be explained by the randomized training procedure that we employ to ensure uncorrelated trees. To investigate the influence of the forest reponse optimization, we replaced this part of our model with the simpler forest vote bias correction step on the instance predictions of the forest. We found that the response optimization is an essential part of the proposed model, as can be seen in Table I. Deactivating also the forest vote bias correction from Section V-C leads to predictions at the chance level. Less pronounced but still positive is the contribution of the Ellipsoid split that we propose. The leaf node regularization via a minimum leaf node size and a redistribution of the positive weights does not have a visible positive effect. The Musk2 data set benefits from this extension while the effect on the Musk1 data set is detrimental.

B. Discussion

Table II shows the accuracy of the models in comparison. The top-most group contains decision tree based algorithms including our proposed model (MIOForest). MIOForest performs better than all the three models, except that it is marginally worse than MITI [2] in MUSK 2. When classification accuracies are averaged over five data sets, MIOForest clearly improves the state-of-the-art in decision tree based multiple instance learning by 3%.

Among the kernel-based MIL algorithms in the second group, MIOForest ranks second after GPMIL [11] by 1%. It is noteworthy that GPMIL is a computationally expensive algorithm that involves inversion of an $N \times N$ kernel matrix, N being the number of training instances, and tuning of kernel hyperparameters.

The third group from the top consists of two variants of a special type of kernel function that calculates the dissimilarity between two bags. Due to the fact that each bag is represented as a single instance during classification, these algorithms are very fast. In addition their accuracy is very competitive. However, the problem of these algorithms is that they are able to make predictions only at the bag level, as opposed to all the other algorithms in comparison, including MIOForest. mi-Graph is on average marginally better than MIOForest.

The methods in the last group include methods depending on various other approaches. MILES [6] and SIL-SVM [5] basically solve the MIL problem using single-instance learning, and EM-DD [24] uses a diverse density based approach combined with expectation maximization. MIOForest is clearly better than all algorithms in this group.

To summarize, MIOForest ranks third among all models in comparison only marginally ($< 1\%$) behind mi-Graph and GPMIL. Note that MIOForest gives the globally best performance for the Fox data set.

VII. Conclusion

We have extended the multi instance tree learning algorithm proposed in [2] with a regularization and weight redistribution scheme, inside/outside split concepts, randomization and a leaf node response optimization. We have shown that our method exhibits superior performance to other decision tree based algorithms on a number of benchmark data sets, and that its accuracy is only slightly worse than the global state-of-the-art.

There exist several interesting future extensions of the model we propose. For instance, exploring additional ways to control the witness/non-witness co-occurrences in positive bags is a promising strategy for future work. In addition, the idea of treating the instance from the bags as non i.i.d. samples as in [25] is another attractive direction of research that might be incorporated in the response optimization stage. Furthermore the instance-to-bag principle presented in [21] could be beneficial also for the decision tree based approach used here.

References

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems*, 15:561–568, 2002.

TABLE I: Analysis on the influence of the proposed extensions. We analyze the impact of the extensions by comparing the accuracies between including and excluding each extension. The results illustrate that all extensions except constraining the minimum leaf node size have a positive influence on model accuracy, and the highest influence comes from response optimization. The best performer version of the algorithm on each data set is shown in bold.

	Drug Activity		Image Classification			
	Musk1	**Musk2**	**Elephant**	**Fox**	**Tiger**	**Average**
MIOForest	**89**	87	**86**	**68**	**83**	**83**
MIOForest w.o. forest (single randomized tree)	79	75	79	62	74	74
MIOForest w.o. Ellipsoid Split	87	86	86	67	82	82
MIOForest w.o. response optimization	82	79	81	65	81	78
MIForest w.o response optimization and bias correction	51	50	50	50	38	48
MIOForest with min. leaf size	88	**88**	85	**68**	**83**	82

TABLE II: Prediction accuracies of our model (MIOForest) and the existing methods on five benchmark data sets. The best performer among decision tree based models is shown in bold, and the best performer among all models is shown in bold and italics. Our model gives the highest accuracies in four of the five data sets among decision tree based models, and its performance is slightly worse than the global state-of-the-art.

		Drug Activity		Image Classification			
Category	**Method**	**Musk1**	**Musk2**	**Elephant**	**Fox**	**Tiger**	**Average**
Decision tree based methods	MIOForest (our method)	**89**	87	**86**	***68***	**83**	**83**
	MIForest [12]	85	82	84	64	82	79
	MITI [2]	84	**88**	N/A	N/A	N/A	N/A
	RandomForest [3]	85	78	74	60	77	75
Kernel based methods	MI-Kernel [1]	88	89	84	60	84	81
	MI-SVM [1]	78	84	81	59	84	77
	mi-SVM [1]	87	84	82	58	79	78
	AW-SVM [10]	86	84	82	64	83	80
	AL-SVM [10]	86	83	79	63	78	78
	MILBoost-Nor [20]	71	61	73	58	56	63
	GPMIL [11]	89	87	84	66	***88***	83
Graph-based methods	mi-Graph [25]	***90***	***90***	***87***	62	86	83
	MI-Graph [25]	89	***90***	85	61	82	81
Other methods	MILES [6]	88	83	81	62	80	79
	EM-DD [24]	85	85	78	56	72	75
	SIL-SVM [5]	88	87	85	53	77	78

[2] H. Blockeel, D. Page, and A. Srinivasan. Multi-instance tree learning. In *Proceedings of International Conference on Machine Learning*, pages 57–64, New York, NY, USA, 2005. ACM.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[5] R. C. Bunescu and R. J. Mooney. Multiple instance learning for sparse positive bags. In *Proceedings of International Conference on Machine Learning*, pages 105–112, New York, NY, USA, 2007. ACM.

[6] Y. Chen, J. Bi, and J. Wang. Miles: Multiple-instance learning via embedded instance selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1931–1947, 2006.

[7] A. Criminisi. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, 2011.

[8] T. Dietterich, R. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[9] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):pp. 916–954, 2008.

[10] P. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. 2006.

[11] M. Kim and F. De La Torre. Gaussian process multiple instance learning. In *Proceedings of International Conference on Machine Learning*, pages 535–542, New York, NY, USA, 2010. ACM.

[12] C. Leistner, A. Saffari, and H. Bischof. MIForests: multiple-instance learning with randomized trees. *Computer Vision–ECCV 2010*, pages 29–42, 2010.

[13] F. Li and C. Sminchisescu. Convex multiple-instance learning by estimating likelihood ratio. *Advances in Neural Information Processing Systems*, pages 1360–1368, 2010.

[14] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, pages 570–576, 1998.

[15] N. Meinshausen. Forest garrote. *Electronic Journal of Statistics*, 3:1288–1304, 2009.

[16] B. Menze, B. Kelm, D. Splitthoff, U. Koethe, and F. Hamprecht. On oblique random forests. *Machine Learning and Knowledge Discovery in Databases*, pages 453–469, 2011.

[17] G. Quellec, M. Lamard, M. D. Abràmoff, E. Decencière, B. Lay, A. Erginay, B. Cochener, and G. Cazuguel. A multiple-instance learning framework for diabetic retinopathy screening. *Medical Image Analysis*, 2012.

[18] J. R. Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[19] P. Sharma, C. Huang, and R. Nevatia. Unsupervised incremental learning for improved object detection in a video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3298–3305. IEEE, 2012.

[20] P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. *Advances in Neural Information Processing Systems*, 2006.

[21] H.-Y. Wang, Q. Yang, and H. Zha. Adaptive p-posterior mixture-model kernels for multiple instance learning. In *Proceedings of International Conference on Machine Learning*, pages 1136–1143, New York, NY, USA, 2008. ACM.

[22] J. Wang and J. Zucker. Solving multiple-instance problem: A lazy learning approach. 2000.

[23] Y. Xu, J. Zhu, E. Chang, and Z. Tu. Multiple clustered instance learning for histopathology cancer image segmentation, classification and clustering. CVPR, 2012.

[24] Q. Zhang, S. Goldman, et al. Em-dd: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems*, 14:1073–1080, 2001.

[25] Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li. Multi-instance learning by treating instances as non-i.i.d. samples. In *Proceedings of International Conference on Machine Learning*, pages 1249–1256, New York, NY, USA, 2009. ACM.