A Boosting Approach to Multiple Instance Learning

Peter Auer and Ronald Ortner

University of Leoben, Franz-Josef-Strasse 18,
8700 Leoben, Austria
{auer,rortner}@unileoben.ac.at

Abstract. In this paper we present a boosting approach to multiple instance learning. As weak hypotheses we use balls (with respect to various metrics) centered at instances of positive bags. For the ∞-norm these hypotheses can be modified into hyper-rectangles by a greedy algorithm. Our approach includes a stopping criterion for the algorithm based on estimates for the generalization error. These estimates can also be used to choose a preferable metric and data normalization. Compared to other approaches our algorithm delivers improved or at least competitive results on several multiple instance benchmark data sets.

1 Introduction

Originally arising from a problem in biochemistry [6] multiple instance learning recently got a lot of attraction (see e.g. [1], [9], [24], [22]). Generally, in a multiple instance learning problem one has to classify sets (so-called *bags*) of instances, where each set is classified positively if it contains at least one instance with a certain property. The difficulty for learning this property is that it is unknown which of the instances is responsible for a positive classification of a bag.

Our motivation to investigate a boosting approach for multiple instance learning comes from object recognition in computer vision. For that we want to learn descriptions of object categories in images from labeled images, where the label indicates whether a relevant object is present in the image or not. Position and pose of the objects in the image are not known. Learning of such object categories by a boosting approach has been demonstrated in [16, 10]. In these experiments local feature vectors in $\mathbb{R}^n$ were extracted from the training images, so that each image can be represented by its features. For a a positively classified image it is assumed that this is due to the presence of particular features. According to the training examples weak hypotheses for AdaBoost [7, 20] are calculated that indicate if certain features are present in an image or not. This is done by comparing features with a similarity measure. More precisely, the weak hypotheses were chosen as balls around feature vector templates, such that an image is classified positively if it contains a feature vector sufficiently close to that template. Thereby, the templates are chosen from the feature vectors of the training images.

From a more abstract view this learning problem can be seen as multiple instance problem: Many objects – not only relevant ones – may be present in an image, and a feature may come from a relevant or irrelevant object. An image is classified positively if at least one relevant object is present. In fact, in this situation we have a generalized multiple instance problem since a relevant object may be described not necessarily by a single feature but by a combination of features: An image will be classified positively if a certain combination or number of relevant features can be extracted from it. Which of the feature vectors are relevant is initially unknown and has to be learned, what poses a typical (generalized) multiple instance problem.

In this paper we consider this boosting approach to multiple instance learning with balls as weak hypotheses in greater detail, and apply it to some of the multiple instance benchmark problems. We find this approach very competitive and it outperforms several other approaches to multiple instance learning.

Section 2 gives the formal details of the multiple instance setting, an overview of different approaches and a short introduction to AdaBoost, the boosting algorithm we apply. In Section 3 we describe in greater detail how we obtain our weak hypotheses, displaying how to calculate optimal balls, and also showing how suitable hyper-rectangles can be calculated by a greedy algorithm. In Section 4 we explain which experiments were conducted and how our algorithm performed compared to other approaches.

2 Multiple Instance Learning with Boosting

2.1 Multiple Instance Learning

Formally, the multiple instance learning problem we deal with is defined as follows. Let the instance space X be a set and $\mathcal{B}$ a finite collection of finite subsets of X. The elements of $\mathcal{B}$ are called *bags*. An unknown target function $y : \mathcal{B} \to \{+1, -1\}$ indicates whether a bag in $\mathcal{B}$ is classified as positive or negative. Actually, y is supposed to be an extension of a labeling function $y : X \to \{+1, -1\}$ such that a bag $B \in \mathcal{B}$ is classified positively iff it contains a positive element, that is, $y(B) = +1 \iff \exists x \in B : y(x) = +1$. The aim is to learn y from the training examples given in $\mathcal{B}$.

Of course, the considered problem setting is the simplest one can think about. We have already mentioned the possibility of more complex situations when describing applications to vision in the introduction. Thus in a generalized multiple instance setting a bag B is classified positively if e.g. at least a certain number of positive instances are contained in B. Another generalized multiple instance problem is introduced in [19].

Several approaches have been tried to offer solutions to multiple instance learning. Axis-parallel rectangles as hypotheses were used in [3] and [6]. Whereas [3] uses a single rectangle, which works quite well under some assumptions on the distribution of the data, in [6] the authors try to grow axis-parallel rectangles containing at least one instance from each positive bag and no instance from a negative bag. Scott et al. [19] adapt and improve an approach of Goldman et al.

[8] to learn d-dimensional patterns. The latter uses as hypotheses weighted combinations of hyper-rectangles over a discretized feature space, where the weights are learned with Winnow [12]. Scott's modification of the original algorithm allows learning in a generalized multiple instance setting as well. A different approach has been tried in [22]. The k nearest neighbors of a new bag (with respect to a certain distance measure) are taken and a majority vote decides, which label the new bag is assigned. Different variants of the k nearest neighbor concept are considered. The EM-DD algorithm of [24] combines the expected maximization (EM) algorithm with the diverse density (DD) algorithm [13, 14]. The DD algorithm tries to find a single instance that is responsible for the positive classification of a bag. More exactly, the key idea is to identify an instance to which many positive bags have close distances and to which instances of negative bags are far away. The diverse density is a measure that captures this concept and corresponds to the likelihood of the corresponding hypothesis, i.e., a high diverse density indicates a good candidate for a "true" hypothesis. Combining this with the idea of the EM algorithm yields the EM-DD algorithm. There are further approaches using neural networks [17], decision trees [18] and kernel methods [9].

2.2 A Boosting Approach

As already mentioned, our approach is to apply a boosting framework, which means given a set of hypotheses $\mathcal{H}$ we use a suitable subset $\mathcal{H}' \subseteq H$ of weak hypotheses that are combined to give a final hypothesis h_f. The AdaBoost [7, 20] algorithm we are going to use, calculates h_f by assigning weights w_B to the training examples $B \in \mathcal{B}$ and calculates weak hypotheses $h \in \mathcal{H}$ which correctly classify a majority (in respect to their weights) of the training examples. According to the error of such an h a weight α_h is calculated for the weak hypothesis. Furthermore, the weights w_B are recalculated such that more weight is put on the bags that were misclassified. The process of finding a suitable weak hypothesis $h \in \mathcal{H}$ and (re)calculating the weights w_B and α_h is repeated and the final hypothesis h_f is then obtained as weighted majority vote of the weak hypotheses, that is,

$$h_f(B) := sgn(\sum_{h \in \mathcal{H}'} \alpha_h h(B)),$$

where $\mathcal{H}' \subseteq \mathcal{H}$ is the set of used weak hypotheses.

Obviously, the weights of the bags can be normalized to sum up to 1 so that we obtain a distribution $w = (w_{B_1}, \ldots, w_{B_{|\mathcal{B}|}})$ over $\mathcal{B}$. Then given a distribution $w = (w_{B_1}, \ldots, w_{B_{|\mathcal{B}|}})$ one can judge the quality of a weak hypothesis via its *distribution accuracy*, which is defined as

$$D(h, w) := \sum_{\substack{B \in \mathcal{B}: \\ h(B) = y(B)}} w_B.$$

We demand from each weak hypothesis used by AdaBoost that its distribution accuracy is $> \frac{1}{2} + \varepsilon$ for some $\varepsilon > 0$. In this case it can be guarantueed that

AdaBoost's final hypothesis approximates the target function y with arbitrary high accuracy (cf. [7]).

In this paper we consider the instance space $X = \mathbb{R}^n$ and our weak hypotheses are balls of arbitrary center and radius with respect to some metric. Actually, our class $\mathcal{H}$ is a bit more restricted. Let $h(x, r)$ denote the open ball with center x and radius r. Then we choose $\mathcal{H}$ to be the set of all balls $h(x, r)$ whose center is contained in a bag $B \in \mathcal{B}$ with $y(B) = +1$. Furthermore, $\mathcal{H}$ shall contain the empty hypothesis. In Section 3 we describe how we determine for each center x the optimal radius r with respect to the current distribution accuracy. The prediction of a ball $h \in \mathcal{H}$ on a bag B is given by

$$h(B) := \begin{cases} +1 & \text{if } h \cap B \neq \emptyset \\ -1 & \text{otherwise.} \end{cases}$$

The following proposition shows that among these balls there is always a weak hypothesis with distribution accuracy larger than $\frac{1}{2} + \varepsilon$ for some fixed ε.

Proposition 1 *For each weight distribution $w = (w_{B_1}, \ldots, w_{B_{|\mathcal{B}|}})$ there is a ball $h = h(x, r)$ in $\mathcal{H}$ such that $D(h, w) > \frac{1}{2} + \frac{1}{4k+2}$, where k is the number of positive bags in $\mathcal{B}$.*

Proof. Set $\gamma := \frac{1}{4k+2}$ and let W^- and W^+ be the sum of the weights of the negative and the positive bags, respectively. If $W^- > \frac{1}{2} + \gamma$ then it is obviously sufficient to choose the empty hypothesis from $\mathcal{H}$. If $W^+ > \frac{1}{2} + \gamma$ we choose an arbitrary x and a sufficiently large radius r so that $X \subset h(x, r)$.

Thus let us assume that $\frac{1}{2} - \gamma < W^-, W^+ \leq \frac{1}{2} + \gamma$. Clearly, there must be a positive bag B with weight $> \frac{1}{k}(\frac{1}{2} - \gamma)$. Then for a suitable $x \in B$ with $y(x) = +1$ and a sufficiently small radius so that $h(x, r) \cap X = \{x\}$ we have

$$D(h(x,r), w) > W^- + \frac{1}{k}\left(\frac{1}{2} - \gamma\right) > \frac{1}{2} - \gamma + \frac{1}{k}\left(\frac{1}{2} - \gamma\right) = \frac{1}{2} + \gamma$$

by some little calculation. □

Remark 1. Obviously a weak hypothesis as in the second part of the proof is of little use since it learns a single instance by heart. To achieve reasonable generalization accuracy we need to find weak hypotheses which classify a significant number of training examples correctly.

Remark 2. Complementing Proposition 1 we would like to remind the reader of the following hardness result due to Auer, Long, and Srinivasan [4]. Consider the multiple-instance problem where each bag contains r instances $\in \mathbb{R}^n$ and a bag is classified positively if it contains an instance from an unknown hyper-rectangle. If for this problem there is a polynomial algorithm for learning from arbitrary distributions, then there is a polynomial algorithm for learning r-term DNF formulas over n variables as well. However, whether such an algorithm exists is one of the important unsolved problems of learning theory (see e.g. [11]). This result indicates that even in this rather simple setting multiple instance learning is a hard problem that probably cannot be solved efficiently in full generality.

3 The Algorithm

In this section we explain how to obtain a suitable set of balls, which will serve as weak hypotheses for AdaBoost.

3.1 Calculating the Optimal Ball

We compute for each x in a positive bag $B \in \mathcal{B}$ a ball with center x, such that its radius is optimal with respect to the current distribution w. More exactly, the optimal radius r_0 is defined as

$$r_0 = \max\{r' > 0 \,|\, D(h(x, r'), w) = \max_r D(h(x, r), w)\}. \tag{1}$$

Note that we did not make any assumptions concerning the used metric so far. In our experiments we used the metric induced by the 2-norm and the ∞-norm, respectively. In the latter case, our balls are axis-parallel hypercubes.

Computing the optimal radius for each instance is time consuming. To speed up the computation the bags $B \in \mathcal{B}$ can be sorted by their distance to x, which is given by

$$d(x, B) = \min_{x_0 \in B} d(x_0, x)$$

for the chosen metric d. This procedure uses $O(mn + |\mathcal{B}| \log |\mathcal{B}|)$ operations per instance x, where n is the dimension of the instance space X and m is the total number of instances in bags of $\mathcal{B}$, i.e., $m = |\bigcup_{B \in \mathcal{B}} B|$. Taking into account the number of instances in positive bags introduces an additional factor of at most m into the overall complexity, so that the overall time complexity for computing the optimal ball is $O(\ell(mn + |\mathcal{B}| \log |\mathcal{B}|))$, where $\ell = |\bigcup_{B \in \mathcal{B}: y(B) = +1} B|$.

However, since AdaBoost computes a lot of iterations, one may compute the distance from each instance in a positive bag to each bag $B \in \mathcal{B}$ once and keep the corresponding distance matrix in memory. Thus the time complexity for one boosting iteration decreases to $O(\ell |\mathcal{B}| \log |\mathcal{B}|)$, which is much cheaper. Indeed, one may reduce this even to $O(\ell |\mathcal{B}|)$ by saving for each instance x the order of the bags with increasing distance to x.

3.2 Extension of the Optimal Hypercube

When using the metric induced by the ∞-norm we can extend the optimal hypercube computed according to the previous section. Since the radius of this $||\cdot||_\infty$-ball was chosen to be optimal with respect to the distribution accuracy, the latter surely decreases when the radius is increased. Thus each hypercube $h(x, r)$ has a *bad instance* x' on its boundary that is contained in some negative bag B'. Note that B' should be classified negatively according to $h(x, r)$ as well, because otherwise the inclusion of x' wouldn't make any difference to the prediction of B'. Obviously, the bad instance prevents our hypercube from growing, at least in some directions. However, we may grow the hypercube – which now turns into a hyper-rectangle – evenly in all other directions that are not blocked by x'. That way, one determines analogously to (1) above the optimal "radius" for all unblocked directions. Again, some directions are blocked by a bad instance and are accordingly fixed. This process is repeated until each direction is either fixed or the corresponding boundary is set to infinity, because there are no instances left in the respective direction. More formally, the algorithm can be described as follows:

1. Initially the hyper-rectangle $R = \times_{i=1}^{n}(a_i, b_i)$ we consider is the optimal hypercube $h(x, r_0)$. The set F contains the directions which are already fixed, where each direction is represented by the index of the coordinate and a sign indicating whether the positive or the negative direction of the respective coordinate is fixed. Initially, $F := \varnothing$.
2. Determine the bad instance x', that is, the instance on the boundary of R that is contained in some bag B' for which $y(B') = -1$ and $B' \cap R = \varnothing$. If there is more than one instance with that property, choose an arbitrary one.
3. Update F according to the directions blocked by the bad instance x', that is,
$$\begin{aligned}(i,+) \in F &\iff x'_i = b_i,\\ (i,-) \in F &\iff x'_i = a_i.\end{aligned}$$
4. Determine analogously to (1) the optimal "radius" r with respect to distribution accuracy for all directions that are not fixed yet.
5. Update R as follows:
$$a_i := \begin{cases} a_i & \text{if } (i,-) \in F \\ x_i - r & \text{otherwise,} \end{cases} \quad \text{and} \quad b_i := \begin{cases} b_i & \text{if } (i,+) \in F \\ x_i + r & \text{otherwise.} \end{cases}$$
6. Repeat steps 2–5 until each direction is fixed or r was set to infinity.

Obviously, the computation of hyper-rectangles as described above requires multiple passes over the instances and is therefore quite expensive. However, if one sorts the instances within the bags by their distance to the center of the initial optimal hypercube, one can increment the pointers and one pass over all the instances is sufficient. This improvement decreases the running time for the computation of the hyper-rectangle from a hypercube to $O((\ell + n)|\mathcal{B}| \log |\mathcal{B}|)$, where n is the dimension of the instance space X and $\ell = \sum_{B \in \mathcal{B}: y(B)=+1} |B|$ is the number of positive instances.

4 Experiments

4.1 The Data Sets

The musk data set [6, 5] originates from the already mentioned problem in biochemistry. One observes different conformations of a molecule and wants to know if a "musk-like" conformation exists for this molecule. For each molecule one is given the set of stable conformations, which constitute a bag as in the definition of the multiple instance problem. A bag is labeled positive, if it contains a musk-like conformation.

The data sets musk1 and musk2 consist of 92 and 102 bags which contain a total of 476 and 6598 instances, respectively, each of which is 168-dimensional.

The robot data set [19] consists of 12 learning problems which are constructed from a basis of 28 bags divided into four categories A,B,C, and D of equal size. Each bag consists of (in average about 34) 2-dimensional instances which represent a 360° light intensity measurement of a robot at a certain position in space. The seven bags in each of the groups A,B,C, and D consist of 'similar' positions.

The learning task is to learn concepts which separate each of the four groups from any two of the others. We label the corresponding learning problems by 'AposBCneg','AposBDneg' etc. corresponding to the choice of groups.

The protein data set [23] consists of 20 positive bags paired with 8 different sets of 20 resp. 28 different negative bags. The data itself is 8-dimensional and consists of 7 characteristics of a protein sequence and information about the position of the window on the whole protein sequence in the remaining coordinate.

4.2 A Stopping Criterion for AdaBoost

For our tests we used the following stopping criterion for the number of boosting iterations. By a leave-one-out cross-validation on the *training examples*[1] we calculated estimates for the generalization error when AdaBoost is run on the *remaining* training examples until the error reached $P\%$, $P = 99, \ldots, 0$. The P^* with the best estimated generalization error was chosen and AdaBoost was run on all training examples until the training error reached $P^*\%$, giving the final hypothesis.

For some data sets it might be advisable to continue boosting even when the training error has reached 0% (cf. [21]). We did not explore this for the data sets we used. However, in this situation a similar approach can be taken to choose the number of boosting iterations: Let M be the number of iterations until training error 0 is reached. Estimate the generalization error when AdaBoost is run for $M \cdot \alpha^k$ iterations, $k = 1, 2, \ldots$, $\alpha > 1$, and choose the best k^*.

4.3 Various Combinations of Metric and Attribute Normalization

We performed experiments with various combinations of metric and attribute normalization. In this respect an interesting question is, if the estimated generalization error from the stopping criterion is a good indicator for which metric and attribute normalization should be chosen. The results seem to answer this question affirmatively. We used the following combinations:

cubes: The weak hypotheses for AdaBoost are balls with respect to the ∞-norm, i.e. axis-parallel hypercubes. No normalization is used.

rectangles: The weak hypotheses for AdaBoost are axis-parallel hyper-rectangles. The hypothesis is grown from the optimal hypercube by the greedy algorithm described in Section 3.2.

[1] This is not to be confused with the cross-validation for estimating the test error. No test examples were used by the stopping criterion.

Table 1. Generalization error (GE) and estimated error (EE) for various combinations of metric and normalization for the robot data set.

data set	cubes	rect.	median	minmax	$\|\| \cdot \|\|_2$	$\mathcal{N}(0,1)$	$\mathcal{N}(\mathbf{0},I)$
AposBCneg GE	0.1429	0	0.1905	0.0476	0.0952	0.0952	0.0952
AposBCneg EE	0.079	0.0752	0.0833	0.0786	0.0786	0.0781	0.0767
AposBDneg GE	0.1429	0.1429	0.1905	0.0476	0.0952	0.0476	0.0476
AposBDneg EE	0.0438	0.0438	0.0919	0.0381	0.0424	0.0381	0.0381
AposCDneg GE	0.2857	0.0526	0.1429	0.1429	0.0476	0.1905	0.1905
AposCDneg EE	0.0805	0.031	0.0448	0.0448	0.081	0.0348	0.0348
BposACneg GE	0	0	0	0	0	0	0
BposACneg EE	0	0	0	0	0	0	0
BposADneg GE	0	0	0	0	0	0	0
BposADneg EE	0	0	0	0	0	0	0
BposCDneg GE	0	0	0	0	0	0	0
BposCDneg EE	0	0	0	0	0	0	0
CposABneg GE	0	0	0	0	0	0	0
CposABneg EE	0	0	0	0	0	0	0
CposADneg GE	0.2857	0.2857	0	0	0.381	0	0
CposADneg EE	0.0333	0.0333	0	0	0.0295	0	0
CposBDneg GE	0	0	0	0	0	0	0
CposBDneg EE	0	0	0	0	0	0	0
DposABneg GE	0	0	0	0	0	0	0
DposABneg EE	0	0	0	0	0	0	0
DposACneg GE	0	0	0.1905	0.1905	0	0	0
DposACneg EE	0	0	0.0324	0.0324	0	0	0
DposBCneg GE	0	0	0.381	0	0	0.2381	0.1905
DposBCneg EE	0	0	0.0443	0	0	0.0329	0.0324

median: The data are preprocessed by linearly transforming each coordinate separately, such that the range between the two quartiles is mapped to the interval $[-1, 1]$. If the two quartiles are equal, the interval is only centered at the quartiles. Hypercubes are used as weak hypotheses.

minmax: The data are preprocessed by linearly transforming each coordinate, such that the range $[min, max]$ of each coordinate is mapped to the interval $[-1, 1]$. Hypercubes are used as weak hypotheses.

$|| \cdot ||_2$: 2-norm balls are used as weak hypotheses. Data are not normalized.

$\mathcal{N}(0, 1)$: Each coordinate is linearly transformed for mean 0 and variance 1. 2-norm balls are then used as weak hypotheses.

$\mathcal{N}(\mathbf{0}, I)$: A linear transformation of all instances is chosen such that the mean of all attributes is 0 and the covariance matrix is the identity. Then 2-norm balls are used as weak hypotheses.

4.4 The Results

The results for the robot data set are shown in Table 1, which displays the generalization error (GE) calculated by leave-one-out cross-validation, and the best estimated generalization error (EE) calculated by the stopping criterion[2]. For the various combinations of metric and attribute normalization the results show a quite good agreement between the ranking of the generalization error and the ranking of the estimated error.

Table 2. Generalization error of the boosting approach and [19] for the robot data.

data set	Boosting	[19]
AposBCneg	0	0
AposBDneg	0.0476	0.0952
AposCDneg	0.0476	0.0476
BposACneg	0	0.0952
BposADneg	0	0.0952
BposCDneg	0	0
CposABneg	0	0.0952
CposADneg	0	0
CposBDneg	0	0
DposABneg	0	0.0476
DposACneg	0	0
DposBCneg	0	0
Average	0.0079	0.0396

In Table 2 we compare our results with the algorithm by Scott et al. [19] (cf. Section 2.1). For this we have chosen that combination of metric and normalization with the minimal estimated error from the stopping criterion. As can be seen our algorithm outperforms the algorithm of [19].

Table 3. Generalization error (GE) and estimated error (EE) for various combinations of metric and normalization for the musk data sets.

data set	cubes	rect.	median	minmax	$\|\|\cdot\|\|_2$	$\mathcal{N}(0,1)$	$\mathcal{N}(\mathbf{0},I)$
musk1 GE	0.1511	0.0804	0.2293	0.1554	0.1196	0.088	0.3717
musk1 EE	0.0515	0.0131	0.111	0.058	0.0456	0.0708	0.2469
musk2 GE	0.1402	0.0578	0.1912	0.1431	0.1265	0.1294	0.1902
musk2 EE	0.0824	0.0549	0.0737	0.0515	0.0403	0.0309	0.1215

The results for the musk data as summarized in Table 3 are based on a 10-fold cross-validation, where the values are averaged over ten runs. A comparison of the different approaches was taken from [2] and is reported in Table 4. Again, the numbers for the boosting approach are those that have the minimal estimated error from the stopping criterion. We find that our approach is quite competitive with these other approaches.

To obtain comparable results with [23] for the protein data sets the tests were performed in the following way: For each of the 8 sets of negative bags and each

[2] The estimated error (EE) is averaged over all runs of the leave-one-out cross-validation for (GE).

Table 4. Generalization error of the boosting approach and [6, 15] for the musk data.

data set	Boosting	IAPR[6]	DD[15]
musk1	0.08	0.076	0.12
musk2	0.129	0.108	0.16

of the 20 positive bags, a classifier is trained where the set of negative bags and the remaining 19 positive bags are used as training examples, resulting in 160 classifiers. Then each bag is assigned to the majority class among all classifiers for which it was not used as training example. The error of this classification by majority is reported as the generalization error for the positive and negative bags. The results are summarized in Table 5. With the exception of the $\mathcal{N}(\mathbf{0}, I)$-normalization our boosting approach again outperforms the competing approach.

Table 5. Generalization error of the boosting approach and [23] for the protein data.

Protein	cubes	rect.	median	minmax	$\|\|\cdot\|\|_2$	$\mathcal{N}(0,1)$	$\mathcal{N}(\mathbf{0},I)$	[23]
Pos. bags	0	0	0	0	0	0	1	0.25
Neg. bags	0.036	0.095	0.036	0.036	0.036	0.036	0	0.25

5 Conclusion

We presented a boosting approach to multiple instance learning that was inspired by problems stemming from generic object recognition. In this context the application of the AdaBoost algorithm with suitable weak hypotheses worked consistently well. Our approach with balls as weak hypotheses improves over or is at least competitive to other approaches. We think that the results strongly benefit from the introduced stopping criterion. The estimated generalization error used by the stopping criterion also turns out to be a good indicator for choosing a distance metric and an attribute normalization.

Since – as already mentioned in the introduction – computer vision seems to be an obvious area of application for multiple instance learning, and due to the encouraging results so far, we will incorporate our findings in future work on object recognition problems.

Acknowlegements

This work was supported by the European project LAVA (IST-2001-34405). We would like to thank Thomas Korimort for implementing the algorithms and conducting some of the tests.

References

1. Amar, R.A., Dooly, D.R., Goldman, S.A., Zhang, Q.: Multiple-instance learning of real-valued data. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Morgan Kaufmann (2001) 3–10
2. Andrews, S., Tsochantaridis, I., Hofmann T.: Support vector machines for multiple instance learning. In: Advances in Neural Information Processing Systems 15. Papers from Neural Information Processing Systems (NIPS 2002), MIT Press (2002)
3. Auer, P.: On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In: Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Morgan Kaufmann (1997) 21–29
4. Auer, P., Long, P.M., Srinivasan, A.: Approximating hyper-rectangles: Learning and pseudorandom sets, *J. Comput. Syst. Sci.* **57**(3): 376-388 (1998)
5. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998)
6. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. In: *Artif. Intell.* **89** (1-2): 31–71 (1997)
7. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *J. Comput. Syst. Sci.* **55**(1): 119–139 (1997)
8. Goldman, S.A., Kwek, S.K., Scott, S.D.: Agnostic learning of geometric patterns. In: *J. Comput. Syst. Sci.* **62**(1): 123–151 (2001)
9. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: Machine Learning. Proceedings of the Nineteenth International Conference (ICML 2002), Morgan Kaufmann (2002) 179-186
10. Fussenegger, M., Opelt, A., Pinz, A., Auer, P.: Object recognition using segmentation for feature detection. Accepted for ICPR 2004 (2004)
11. Klivans, A., Servedio, R.A.: Learning DNF in time $2^{\tilde{O}(n^{1/3})}$ In: Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC 2001), ACM (2001) 258-265
12. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. In: *Machine Learning* **2**(4): 285-318 (1988)
13. Maron, O.: Learning from ambiguity. Technical Report AITR-1639 (1998)
14. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: Advances in Neural Information Processing Systems 10 (NIPS 1997), MIT Press (1998)
15. Maron, O., Ratan, A.L.: Multiple-instance learning for natural scene classification. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Morgan Kaufmann (1998) 341-349
16. Opelt, A., Fussenegger, M., Pinz, A., Auer, P.: Weak hypotheses and boosting for generic object detection and recognition. In: Computer Vision - ECCV 2004. 8th European Conference on Computer Vision. Proceedings, Part II. Lecture Notes in Computer Science 3022, Springer (2004) 71-84
17. Ramon, J., Raedt, L.D.: Multi instance neural networks. In: Proceedings of IMCL-2000 workshop on Attribute-Value and Relational Learning (2000)
18. Ruffo, G.: Learning single and multiple instance decision trees for computer security applications. Doctoral dissertation. Department of Computer Science, University of Turin, Torino, Italy (2000)

19. Scott, S.D., Zhang, J., Brown, J.: On generalized multiple-instance learning. Technical report UNL-CSE-2003-5, University of Nebraska (2003)
20. Schapire, R.: The boosting approach to machine learning: An overview. In: MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, Mar. (2001)
21. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. In: *Ann. Statist.* **25**(5): 1651–1686 (1998)
22. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Morgan Kaufmann, San Francisco, CA (2000) 1119–1126
23. Wang, C., Scott, S.D., Zhang, J., Tao, Q., Fomenko, D.E., Gladyshev, V.N. : A study in modeling low-conservation protein superfamilies. Technical report UNL-CSE-2004-0003, University of Nebraska (2004)
24. Zhang, Q., Goldman, S.: EM-DD: An improved multiple-instance learning technique. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), MIT Press (2001) 1073–1080