

A Boosting Approach to Multiple-Instance Learning*

Peter Auer

Ronald Ortner

University of Leoben, Franz-Josef-Strasse 18
8700 Leoben, Austria

AUER@UNILEOBEN.AC.AT

RORTNER@UNILEOBEN.AC.AT

Editor: NN

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 metric induced by the ∞ -norm these hypotheses can be modified into hyper-rectangles by a greedy algorithm. Compared to other approaches our algorithm delivers improved or at least competitive results on several multiple-instance benchmark data sets.

Keywords: multiple-instance learning, boosting

1. Introduction

Multiple-instance learning originally started with a problem arising in biochemistry (Dietterich et al., 1997). 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 original 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 Opelt et al. (2004) and Fussenegger et al. (2004) (there has been similar prior work in face detection by Viola and Jones (2001a,b)). In the 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 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 (Freund and Schapire, 1997; Schapire, 2001) 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. The respective templates are chosen from the feature vectors of the training images.

*. A preliminary version of this paper appeared in *Machine Learning, Proceedings of the 15th European Conference on Machine Learning (ECML 2004)*, pages 63–74, Pisa, Italy, 2004.

From a more abstract point of view this learning task 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, which poses a typical (generalized) multiple-instance problem. Thus, approaches for multiple-instance learning have already been used for problems in computer vision, and there is still a lot of mutual stimulation between the two areas (see e.g. Maron and Ratan, 1998; Andrews et al., 2002; Chen and Wang, 2004; Chen et al., 2006; Viola et al., 2005).

In this paper we consider the mentioned 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.

The paper is organized as follows. 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

In this section we give a formal exposition of the multiple-instance problem setting, followed by an overview of approaches applied to it so far. Finally, we give some details of the boosting algorithm Adaboost and a rough sketch of how we are going to apply it to multiple-instance problems.

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 $y : \mathcal{B} \rightarrow \{+1, -1\}$ 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 $y : X \rightarrow \{+1, -1\}$ on the instance space 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 a subset of training examples $\mathcal{B}' \subseteq \mathcal{B}$.

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 computer 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 is contained in B . Another generalized multiple-instance problem is introduced by Scott

et al. (2003).

Several approaches have been tried to offer solutions to multiple-instance problems. Axis-parallel rectangles as hypotheses were used by Dietterich et al. (1997), Long and Tan (1988), Auer et al. (1998), and Auer (1997). Dietterich et al. try to grow axis-parallel rectangles containing at least one instance from each positive bag and no instance from a negative bag. Long and Tan give PAC bounds for an algorithm under the (rather artificial) assumption that instances are drawn from a product distribution. Later, Auer et al. (1998) gave PAC bounds for an improved algorithm where this assumption is not needed anymore. Finally, this improved algorithm was modified by Auer (1997) for practical purposes. It uses simple statistics of the training data and works quite well under some assumptions on the distribution of the data.

Scott et al. (2003) adapt and improve an approach of Goldman et al. (2001) 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 (Littlestone, 1988). The modification of Scott et al. (2003) allows learning in a generalized multiple-instance setting as well.

A different approach has been tried by Wang and Zucker (2000). The k nearest neighbors of a new bag (with respect to a certain distance measure) are taken and a majority vote decides, which label is assigned to the new bag. Different variants of the k nearest neighbor concept are considered.

The DD algorithm (Maron, 1998; Maron and Lozano-Pérez, 1997) 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 that is close to many positive bags and far away from negative bags. 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. The EM-DD algorithm of Zhang and Goldman (2001) combines this approach with the expectation maximization (EM) algorithm.

There are also approaches to adapt “ordinary” supervised learning algorithms such as using neural networks (Ramona and Raedt, 2000), decision trees (Ruffo, 2000) and kernel methods (Andrews et al., 2002; Gärtner et al., 2002; Chen et al., 2006; Cheung and Kwok, 2006). A survey of these approaches can be found in Ray and Craven (2005). Similarly, Rahmani and Goldman (2006) use an algorithm to transform multiple-instance problems so that graph-based semi-supervised learning algorithms become applicable. Finally, combining several multiple-instance learning algorithms via ensemble learning has been proposed in Zhou and Zhang (2003).

Our algorithm is neither the first nor only boosting approach to multiple-instance learning. Andrews and Hofmann generalized linear programming boosting using results from disjunctive programming?. Viola, Platt, and Zhang use variants of AdaBoost to improve on work on object recognition Viola and Jones (2001a,b), however offer no general approach to multiple-instance learning. Finally, we’d like to mention the work of Xu and Frank (2004) who also applied boosting to multiple-instance learning tasks. However, their approach solves the problem in a somewhat different setting. They assume that the bag labels are generated in a two-stage process. In the first stage, probabilities for the labels of single instances are determined, which in the second stage are combined to give probabilities

for the bag labels. Thereby it is assumed that each instance in a bag contributes equally to a bag label's probability.

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 $\mathcal{H}' \subseteq \mathcal{H}$ of weak hypotheses that are combined to give a final hypothesis h_f . The boosting algorithm we are going to use is AdaBoost (Freund and Schapire, 1997; Schapire, 2001).

AdaBoost maintains a distribution over the training examples, that is, to each bag $B \in \mathcal{B}'$ a weight w_B is assigned, such that all weights sum up to 1. Given such a distribution $w = (w_{B_1}, \dots, w_{B_{|\mathcal{B}'|}})$ one can judge the quality of a weak hypothesis via its *distribution accuracy*, which is defined as

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

Each of the weak hypotheses $h \in \mathcal{H}'$ then is chosen such that it correctly classifies a majority (in respect to the current distribution w) of the training examples, that is, $D(h, w) > \frac{1}{2}$.

According to the error $\epsilon(h, w) := 1 - D(h, w)$ of such a weak hypothesis h , a weight α_h for h is calculated as

$$\alpha_h := \frac{1}{2} \log \left(\frac{1 - \epsilon(h, w)}{\epsilon(h, w)} \right).$$

Then the distribution over the bags is recalculated such that more weight is put on the bags that were misclassified. More precisely, for each bag $B \in \mathcal{B}'$

$$w_B := \frac{\exp(-\alpha_h y(B) h(B)) w_B}{Z},$$

where Z is chosen such that the w_B still sum up to 1.

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 then is obtained as weighted majority vote of the weak hypotheses, that is,

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

where \mathcal{H}' is the list of used weak hypotheses $\in \mathcal{H}$.

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. Let $h(x, r) := \{y \in X \mid d(x, y) < r\}$ denote the open ball with center x and radius r , where $d(x, y)$ is the distance between x and y . Then we choose balls $h(x, r)$ with¹ $r \geq 0$ whose center is contained in a bag $B \in \mathcal{B}'$ with $y(B) = +1$. 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

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

1. For $r = 0$ one obtains the empty hypothesis.

We demand from each weak hypothesis used by AdaBoost that its distribution accuracy is $> \frac{1}{2} + \varepsilon$ for some $\varepsilon > 0$. The following proposition shows that in our setting there is always a suitable ball $h(x, r)$ with distribution accuracy larger than $\frac{1}{2} + \varepsilon$ for some fixed ε . This is sufficient for correct classification of the training examples by the final hypothesis h_f (cf. Freund and Schapire, 1997). However, the generalization error on the test examples will depend on the number of weak hypotheses used.

Proposition 1 *For each weight distribution $w = (w_{B_1}, \dots, w_{B_{|\mathcal{B}'|}})$ 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 $(\bigcup_{B \in \mathcal{B}'} B) \subset h(x, r)$.

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 $h(x, r) \cap (\bigcup_{B \in \mathcal{B}'} B) = \{x\}$ 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. Note that for this last step we need to assume that the training examples really stem from a multiple-instance learning problem as specified in Section 2.1. Otherwise it might be the case that each $x \in B$ with $y(x) = +1$ also occurs in a negative bag. ■

Remark 1 *Obviously a weak hypothesis as in the second part of the proof is of little use, as 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 of Auer et al. (1998). 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. Klivans and Servedio, 2001). 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 first explain in more detail how to obtain a suitable set of balls, which will serve as weak hypotheses for AdaBoost. Then we show how—alternatively—in the case of ∞ -norm balls one may extend these balls using a greedy algorithm and then use the arising hyper-rectangles as weak hypotheses.

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 precisely, the optimal radius r_0 is defined as

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

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 $\ell \leq m$ 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 performs many 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 (cf. Figure 3.2) 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

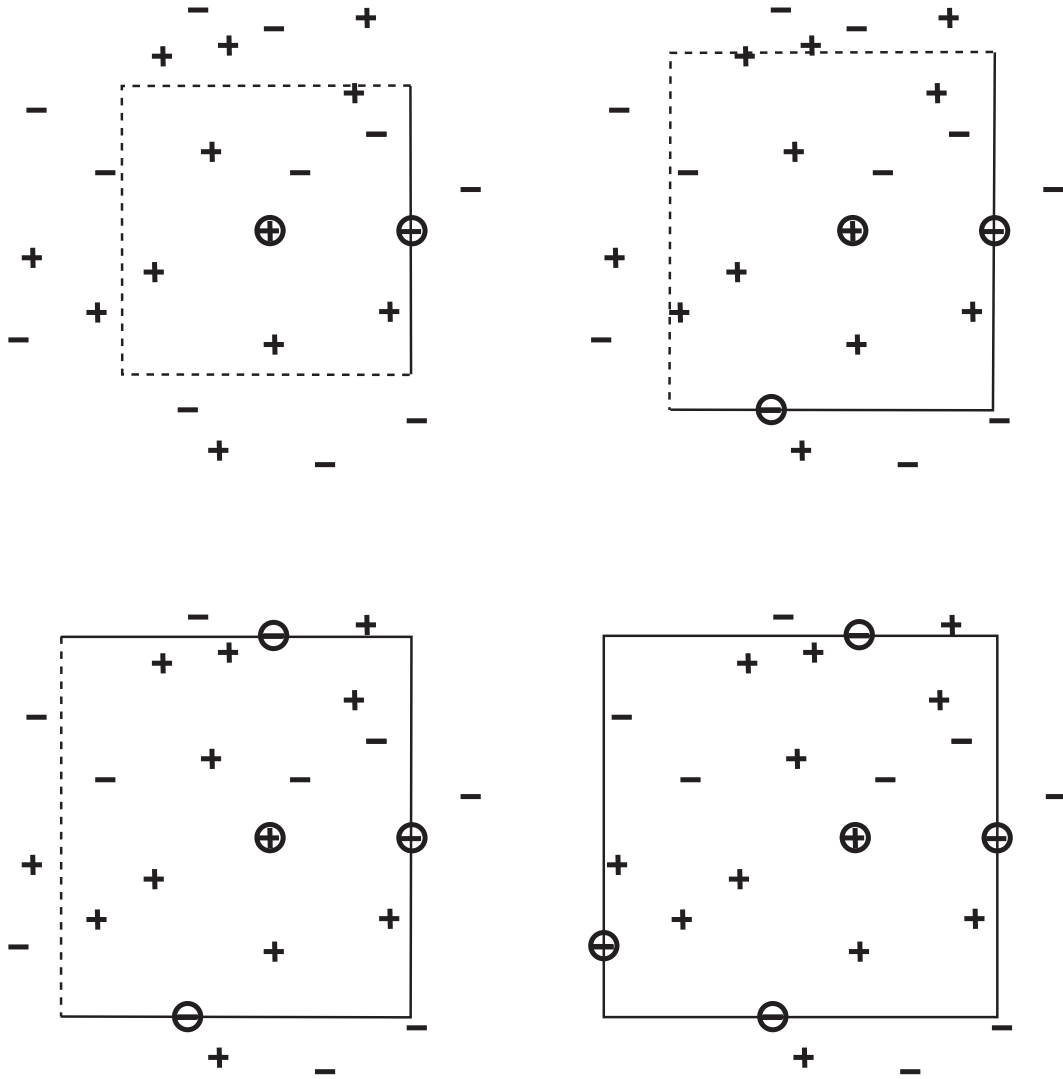


Figure 1: Illustration of the algorithm for extending the optimal hypercube: One starts with the optimal hypercube with center \oplus whose growing is blocked by a bad instance \ominus . The corresponding direction is fixed (continuous line), while all other directions are evenly increased (dashed lines), until another direction is blocked by another bad instance. In the example shown above this process is repeated until every direction is blocked.

the positive or the negative direction of the respective coordinate is fixed. Initially, $F := \emptyset$.

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 $B' \cap R = \emptyset$. 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. To speed things up one sorts the instances within the bags by their distance to the center of the initial optimal hypercube and keeps also in mind the direction(s), in which the distance is assumed. Then in each bag only the instance that is closest to the center needs to be considered. If however the respective direction is blocked before this instance becomes included, one takes the next instance. Thus one pass over all the instances is sufficient. This improvement decreases the running time for the computation of each hypercube-extension 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

In this section we give a short description of the data sets used in the experiments and further details on how the experiments were conducted, including a stopping criterion for AdaBoost. Finally we present our results and compare them to results of competing algorithms.

4.1 The Data Sets

We conducted tests on the following data sets:

The musk data set (Dietterich et al., 1997; Newman et al., 1998) 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 (Scott et al., 2003) 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 visual image of a robot’s current position. The seven bags in each of the groups A,B,C, and D consist of ‘similar’ positions near the same landmark.

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 (Wang et al., 2004) consists of 20 positive and 160 negative bags. In the experiments the latter were split up into 8 equal-sized sets of negative bags (for details see Section 4.4 below). The data itself are 8-dimensional and consist of 7 characteristics of a protein sequence and information about the position of the window on the whole protein sequence in the remaining coordinate.

The Corel image data set (Cheung and Kwok, 2006) consists of 1000 images, each image belonging to one of ten classes (with 100 images in each class). For our experiments we used the data as processed by (Cheung and Kwok, 2006): each image was partitioned into non-overlapping segments using k -means clustering, where segments are 9-dimensional. Then images are regarded as a bags with segments as instances. On average, there are four instances per bag.

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*² we calculated estimates for the generalization error when AdaBoost is run on the *remaining* training examples until the error reached $P\%$, where $P = 99, \dots, 0$. 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. Schapire et al., 1998). 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. Then estimate the generalization error when AdaBoost is run for $M \cdot \alpha^k$ iterations, where $k = 1, 2, \dots, \alpha > 1$, 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 on the training set. 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. For the classification of the test examples we chose the metric and normalization that achieved minimal estimated generalization error (calculated by cross-validation on the training set

2. 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	$\ \cdot\ _2$	$\mathcal{N}(0, 1)$	$\mathcal{N}(\mathbf{0}, I)$	bestnorm
AposBCneg GE	14.29	0	9.52	9.52	9.52	19.05	14.29	9.52
AposBCneg EE	7.90	7.52	8.10	8.10	7.86	7.81	7.67	5.71
AposBDneg GE	14.29	14.29	19.05	4.76	9.52	4.76	4.76	14.29
AposBDneg EE	4.38	4.38	8.71	4.05	4.24	3.81	3.81	3.43
AposCDneg GE	28.57	5.26	14.29	14.29	4.76	19.05	19.05	14.29
AposCDneg EE	8.05	3.10	4.43	4.48	8.10	3.48	3.48	2.48
BposACneg GE	0	0	0	0	0	0	0	0
BposACneg EE	0	0	0	0	0	0	0	0
BposADneg GE	0	0	0	0	0	0	0	0
BposADneg EE	0	0	0	0	0	0	0	0
BposCDneg GE	0	0	0	0	0	0	0	0
BposCDneg EE	0	0	0	0	0	0	0	0
CposABneg GE	0	0	0	0	0	0	0	0
CposABneg EE	0	0	0	0	0	0	0	0
CposADneg GE	28.57	28.57	0	0	38.10	0	0	0
CposADneg EE	3.33	3.33	0	0	2.95	0	0	0
CposBDneg GE	0	0	0	0	0	0	0	0
CposBDneg EE	0	0	0	0	0	0	0	0
DposABneg GE	0	0	0	0	0	0	0	0
DposABneg EE	0	0	0	0	0	0	0	0
DposACneg GE	0	0	19.05	19.05	0	0	0	9.52
DposACneg EE	0	0	3.24	3.24	0	0	0	0
DposBCneg GE	0	0	23.81	0	0	23.81	19.05	0
DposBCneg EE	0	0	4.52	0.24	0	3.29	3.24	0

as described above). In case where the minimum was achieved by several normalizations (this happened e.g. for the rather small robot data) the choice was made according to a majority vote with ties broken arbitrarily.

These are combinations of metric and normalization we have used:

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.

Table 2: Generalization error of the boosting approach and SZB (Scott et al., 2003) for the robot data.

data set	Boosting	SZB
AposBCneg	9.52	0
AposBDneg	14.29	9.52
AposCDneg	14.29	4.76
BposACneg	0	9.52
BposADneg	0	9.52
BposCDneg	0	0
CposABneg	0	9.52
CposADneg	0	0
CposBDneg	0	0
DposABneg	0	4.76
DposACneg	9.52	0
DposBCneg	0	0
Average	3.97	3.97

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.³ For the various combinations of metric and attribute normalization the results show quite a good agreement between the ranking of the generalization error and the ranking of the estimated error.

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

In Table 2 we compare our results with the algorithm of Scott et al. (2003) (cf. Section 2.1). For this we have chosen that combination of metric and normalization with the minimal estimated error from the stopping criterion (the bold entries in Table 1). As can be seen our algorithm outperforms the algorithm of Scott et al. (2003) (bold entries indicate a smaller error). The slightly larger error for the ‘AposCDneg’-set results from choosing the “wrong” normalization-metric-combination by the minimal estimated error, as the 2-norm would have had a smaller error (cf. Table 1).

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	$\ \cdot\ _2$	$\mathcal{N}(0, 1)$	$\mathcal{N}(\mathbf{0}, I)$	bestnorm
musk1 GE	15.11	8.04	22.07	17.07	11.96	10.22	36.52	8.04
musk1 EE	5.15	1.31	10.05	5.82	4.56	5.82	24.62	1.24
musk2 GE	14.02	5.78	18.43	13.43	12.65	12.26	18.73	11.67
musk2 EE	8.24	5.49	8.70	4.92	4.03	3.75	12.29	3.31

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 Andrews et al. (2002) 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

Table 4: Generalization error of the boosting approach, IAPR (Dietterich et al., 1997), and DD (Maron and Lozano-Pérez, 1997) for the musk data.

data set	Boosting	IAPR	DD
musk1	8.04	7.6	12
musk2	11.67	10.8	16

approaches. Again, as can be seen from Table 3, for the musk2-data the rectangles achieved an even smaller error than the minmax-normalization reported in Table 4.

Table 5: Generalization error of the boosting approach and k NN (Wang et al., 2004) for the protein data.

Protein	cubes	rect.	median	minmax	$\ \cdot\ _2$	$\mathcal{N}(0, 1)$	$\mathcal{N}(\mathbf{0}, I)$	k NN
Pos. bags	10	10	10	10	10	5	10	25
Neg. bags	0	0	0	0	0	0	0	25

Table 6: Generalization error (GE) and estimated error (EE) in % for various combinations of metric and normalization for the ten classes of the Corel image data set.

data set	cubes	rect.	median	minmax	$\ \cdot\ _2$	$\mathcal{N}(0, 1)$	$\mathcal{N}(\mathbf{0}, I)$	bestnorm
Class 0 GE	12.64	10.72	12.48	13.36	11.44	13.12	12.32	11.12
Class 1 GE	16.56	15.76	15.36	15.44	16.16	14.96	11.44	12.00
Class 2 GE	18.48	18.72	15.60	17.04	17.04	14.88	13.76	14.32
Class 3 GE	4.16	3.68	3.92	3.92	4.00	3.76	0.16	4.00
Class 4 GE	0.96	0.48	0.32	0.32	0.72	0.40	1.84	0.72
Class 5 GE	17.20	16.40	17.04	16.96	16.80	16.56	15.76	15.76
Class 6 GE	5.68	3.92	8.88	6.40	5.28	7.12	6.88	3.76
Class 7 GE	6.80	6.24	3.92	7.92	6.24	6.08	6.08	4.48
Class 8 GE	19.68	18.48	17.84	19.20	19.36	16.96	13.36	14.80
Class 9 GE	9.52	8.72	10.00	8.96	9.04	10.40	7.84	7.76
avg GE	11.17	10.31	10.54	10.95	10.61	10.42	8.94	8.87
avg ECE	3.64	3.48	3.62	3.96	3.86	3.64	3.26	2.70

To obtain comparable results with Wang et al. (2004) for the protein data the tests were performed in the following way: For each of the 8 sets of negative bags and each 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 60 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.

Finally, for the Corel image data set, we followed the setup of Cheung and Kwok (2006) to obtain comparable results. The data are randomly partitioned into a training set and a test set, such that both sets consist of exactly 50 images of each class. Each class is learned against all other classes. The experiment is repeated five times. In Table 6 we report the average accuracy for each single class as well as average estimated generalization error. As can be seen from the table, even though real and estimated errors for different metrics and normalizations are quite close to each other, the estimated error is still a good indicator for ordering of the real errors.

In Table 7 the average accuracy over all classes is compared to results of other approaches. Our approach is by far the best.

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.

Table 7: Generalization error of the boosting approach, DD-SVM Chen and Wang (2004), Hist-SVM Chapelle et al. (1999), MI-SVM Andrews et al. (2002), SVM with MI kernel Cheung and Kwok (2006), and the improved SVM with MI kernel Cheung and Kwok (2006) for the Corel image data in % together with 95% confidence interval.

	Boosting	DD-SVM	Hist-SVM	MI-SVM	SVM (MI kernel)	SVM+(MI kernel)
avg GE	8.87	18.5	33.3	25.3	15.9	15.6
conf. interv.	[7.54, 9.42]	[15.5, 21.5]	[31.1, 35.5]	[24.7, 25.9]	[15, 16.8]	[14.22, 16.98]

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.

Acknowledgements. This work was supported in part by the European project LAVA (IST-2001-34405), the Austrian Science Fund FWF (S9104-N04 SP4) and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. We would like to thank Thomas Korimort for implementing the algorithms and conducting some of the tests.

References

- Stuart Andrews and Thomas Hofmann. Multiple instance learning via disjunctive programming boosting. In *Advances in Neural Information Processing Systems 16. Papers from Neural Information Processing Systems (NIPS 2003)*. MIT Press, 2004.
- Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple instance learning. In *Advances in Neural Information Processing Systems 15. Papers from Neural Information Processing Systems (NIPS 2002)*, pages 561–568, 2002.
- Peter Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*, pages 21–29, 1997.
- Peter Auer, Philip M. Long, and Aravind Srinivasan. Approximating hyper-rectangles: Learning and pseudorandom sets. *Journal of Computer and System Sciences*, 57(3): 376–388, 1998.
- Olivier Chapelle, Patrick Haffner, and Vladimir Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10:1055–1064, 1999.

- Yixin Chen, Jinbo Bi, and James Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. to appear.
- Yixin Chen and James Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:919–939, 2004.
- Pak-Ming Cheung and James T. Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, 2006.
- Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.
- Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *17th International Conference on Pattern Recognition (ICPR 2004)*, Vol. III, pages 41–44, 2004.
- Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In *Machine Learning. Proceedings of the Nineteenth International Conference (ICML 2002)*, pages 179–186, 2002.
- Sally A. Goldman, Stephen K. Wek, and Stephen D. Scott. Agnostic learning of geometric patterns. *Journal of Computer and System Sciences*, 62(1):123–151, 2001.
- Adam Klivans and Rocco A. Servedio. Learning dnf in time $2^{\tilde{O}(n^{1/3})}$. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 258–265, 2001.
- Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
- Philip M. Long and Lei Tan. Pac learning axis-aligned rectangles with respect to product distributions from multiple-instance examples. *Machine Learning*, 30(1):285–318, 1988.
- Oded Maron. Learning from ambiguity. Technical Report Technical Report AITR-1639, MIT AI Lab, Cambridge, MA, 1998.
- Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10 (NIPS 1997)*, pages 570–576, 1997.
- Oded Maron and Aparna Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pages 341–349, 1998.

- D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. URL <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Computer Vision - ECCV 2004. Proceedings of the 8th European Conference on Computer Vision, Volume II*, pages 71–84, 2004.
- Rouhollah Rahmani and Sally A. Goldman. Missl: Multiple-instance semi-supervised learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 705–712, 2006.
- Jan Ramona and Luc De Raedt. Multi instance neural networks. In *Proceedings of IMCL-2000 workshop on Attribute-Value and Relational Learning*, 2000.
- Soumya Ray and Mark Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 697–704, 2005.
- Giancarlo Ruffo. *Learning single and multiple instance decision trees for computer security applications*. PhD thesis, Department of Computer Science, University of Turin, Torino, Italy, 2000. Doctoral dissertation.
- Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.
- Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 25(5): 1651–1686, 1998.
- Stephen D. Scott, Jun Zhang, and Joshua Brown. On generalized multiple-instance learning. Technical Report Technical Report UNL-CSE-2003-5, University of Nebraska, 2003.
- Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, 2005.
- Paul A. Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 1311–1318, 2001a.
- Paul A. Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 511–518, 2001b.
- Chang Wang, Stephen D. Scott, Jun Zhang, Qingping Tao, Dmitri E. Fomenko, and Vadim N. Gladyshev. A study in modeling low-conservation protein superfamilies. Technical Report Technical Report UNL-CSE-2004-0003, University of Nebraska, 2004.
- Jun Wang and Jean-Daniel Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 1119–1126, 2000.

- Xin Xu and Eibe Frank. Logistic regression and boosting for labeled bags of instances. In *Advances in Knowledge Discovery and Data Mining. Proceedings of the 8th Pacific-Asia Conference (PAKDD 2004)*, pages 272–281, 2004.
- Qi Zhang and Sally A. Goldman. Em-dd: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 1073–1080, 2001.
- Zhi-Hua Zhou and Min-Ling Zhang. Ensembles of multi-instance learners. In *492-502*, pages Machine Learning: ECML 2003, 14th European Conference on Machine Learning, LNCS 2837. Springer, 2003.