One-Class Multiple Instance Learning and Applications to Target Tracking

Karthik Sankaranarayanan[1] and James W. Davis[2]

[1] IBM Research, India
kartsank@in.ibm.com
[2] Ohio State University, USA
jwdavis@cse.ohio-state.edu

Abstract. Existing work in the field of Multiple Instance Learning (MIL) have only looked at the standard two-class problem assuming both positive and negative bags are available. In this work, we propose the first analysis of the one-class version of MIL problem where one is only provided input data in the form of positive bags. We also propose an SVM-based formulation to solve this problem setting. To make the approach computationally tractable we further develop a iterative heuristic algorithm using instance priors. We demonstrate the validity of our approach with synthetic data and compare it with the two-class approach. While previous work in target tracking using MIL have made certain run-time assumptions (such as motion) to address the problem, we generalize the approach and demonstrate the applicability of our work to this problem domain. We develop a scene prior modeling technique to obtain foreground-background priors to aid our one-class MIL algorithm and demonstrate its performance on standard tracking sequences.

1 Introduction

Multiple Instance Learning (MIL) is machine learning paradigm where the data is presented as collections of instances called "bags" associated with labels. (positive - contains at least one positive instance, negative - every instance in it is negative). While this is a standard two-class MIL problem setting, its one-class variant where the data is available only in the form of positive bags (and no negative bags) has been motivated in previous work [1], but to the best of our knowledge, have not been studied yet. One-class MIL formulations seem to be the appropriate setting for many real world tasks where only the data from the positive class are naturally available - for example, social network data (facebook "likes"), object retrieval, document extraction, biological applications such gene expressions, protein-protein interaction (PPI) networks, etc. Even though the data here exist in a multiple-instance setting, previous work have imposed a supervised one-class approach to them.

Specifically in the domain of target tracking, two-class MIL approaches have been employed in previous work [2,3]. A potential limitation with the existing formulations of the localization and tracking problem as a standard two-class

K.M. Lee et al. (Eds.): ACCV 2012, Part III, LNCS 7726, pp. 126–139, 2013.
© Springer-Verlag Berlin Heidelberg 2013

MIL approach (with positive and negative bags) is that they use certain assumptions about the video to separate out targets from the background, such as motion by assuming that the targets of interest are moving in the scene. To make these applications more general, we explore how far we can go without imposing a restriction (such as motion) on our positive class. So, for example, this would allow cases where the target of interest stops and waits for a few seconds before moving on. In such a case, we would consider potentially all instances from each image as being part of a positive bag, without having any explicit knowledge of the negative bags. The goal of the localization and tracking problem is now to find the most persistent object across these positive bags, which would be our target of interest. This more general formulation is clearly a harder problem, and can be viewed as a Multiple Instance analog of the one-class supervised learning problem [4] where one is only given labeled data from the positive class.

2 Related Work

In this section, we discuss previous work in MIL, related work from the general one-class learning domain, and discuss potential applicability to our problem in the MIL setting.

Since the original work of [5] introducing the MIL paradigm, there have been various algorithms proposed such as Diverse Density (DD) [6] and SVM techniques [7]. One-class MIL has not yet been studied; even the most recent work of [8] in MIL, the focus is on efficient instance selection from positive bags by exploiting the negative instances distribution. While MIL has also been employed for object localization and tracking in [2,9,3], they however make simplifying assumptions to separate the data into positive and negative bags, thus limiting their applicability. The work of [2] uses an online boosting framework [10] to build an additive strong classifier by choosing a succession of weak classifiers using Haar-like features. This approach requires manual initialization to determine what qualifies as the set of instances that form the positive bags and what forms the negative bags. Similarly, while the work of [3] gets around manual labeling, it imposes a motion model by assuming that the targets are always moving, and exploit this assumption to form bags. In our work, we seek to explore the problem without making any such run-time assumptions.

In general, one-class models separate the desired class of examples (referred to as the core or the concept) from the other examples (noise) where the noise model is either unknown or too complex to be explicitly modeled. After the initial density-based clustering work of [11] in this, two complementary approaches emerged which treat these problems as either outside-in (treat it as an outlier detection problem) [12,4] or inside-out (identify a small, coherent subset of the data that captures the concept) [13,14]. Further, there has been work using hyperspherical boundaries [12,13,14] and, more recently, using an information-theoretic approach [15].

The extension of any of these approaches to the MIL paradigm is essentially non-trivial. This is because while the positive bags have a small number of

positive instances put together, they also contain a lot of negative instances. Therefore the one-class MIL can either be approached as an unsupervised learning problem with some additional restrictions (MIL constraints) or as a supervised learning problem where you don't directly have the instance labels but have additional knowledge of how the labels relate to each other. We propose an SVM approach based on the latter idea where the main constraint is that, while learning the core concept (the one-class), we impose the additional restriction that it must contain at least one instance from each of the positive bags.

3 Support Vector Machine One-Class MIL Framework

Support Vector Machines (SVM) are one of the most popularly used supervised learning methods used for classification, regression, ranking, and other such learning tasks. While there can be many hyperplanes that separate the space between the examples from the two classes of data, the model that the SVM training algorithm builds is a representation so that the examples of the two classes are divided by a hyperplane with as large a margin between them as possible (called the maximum-margin hyperplane).

In the one-class Multiple Instance setting, the training data is available only in the form of positive bags. Formally, this can be represented as $D_{oc} = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$ where each bag $X_i = \{x_{i1}, x_{i2}, ..., x_{im_i}\}$ is a positive bag (containing m_i instances) and therefore has a bag label $Y_i = 1$. This means that for each bag $\mathbf{B_i}$, there exists at least one instance $x_{ij} \in \mathbf{B_i}$ that is positive so that $y_{ij} = 1$, and the other instances could be positive or negative $y_{ij} \in \{-1, 1\}$. Note that the instance labels y_{ij} are not actually given but it is only the bag labels Y_i that are available. The relation between the instance labels and their bag labels is captured using the max operator $Y_i = \max_{j \in m_i} y_{ij}$, and can be formulated in terms of a linear constraint as given in Eqn. 1.

$$\sum_{j \in m_i} \frac{y_{ij} + 1}{2} \geq 1 \quad \forall i \text{ s.t. } Y_i = 1 \tag{1}$$

Since what we wish to learn is an instance-level classifier (so that at test time we can classify each *instance* as positive or negative), in terms of the instance labels y_{ij}, the generalized soft-margin SVM formulation employing this constraint can be written as given in Eqn. 2

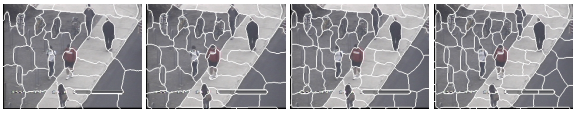
$$\begin{aligned} \min_{y_{ij}, \mathbf{w}, b, \xi} \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C \textstyle\sum_{i \in N} \sum_{j \in m_i} \xi_{ij} \\ \text{subject to} \quad & y_{ij}\left(\mathbf{w}^T \cdot \phi(x_{ij}) + b\right) \geq 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0, y_{ij} \in \{-1, 1\}, \textstyle\sum_{j \in m_i} \frac{y_{ij}+1}{2} \geq 1 \end{aligned} \tag{2}$$

In this formulation, the function $\phi(x)$ maps the instance x to a higher-dimensional kernel space, $C > 0$ is the regularization parameter, and the variables ξ_{ij} are the slack variables to allow for classification errors in the training data.

For our target localization and tracking application, we treat each image in the video sequence as a positive bag. Each image is segmented at different levels using the segmentation algorithm of [16] and these segments are then employed as instances. An example of such a multi-level segmentation of a single image from a sequence is shown in Fig. 1. Each segment instance x_{ij} is modeled using a vector of covariance features derived from its covariance matrix representation C_R which is calculated as

$$C_R = \frac{1}{n} \sum_{k \in R} (f_k - \mu_R)(f_k - \mu_R)^T \tag{3}$$

where $f_k = [x\ y\ r\ g\ b\ I_x\ I_y]$ is a 7 dimensional feature vector using a combination of position, color, and gradient values at each pixel location in the segment R of size n pixels, and μ_R is the mean feature vector within the segment.



$n = 40$ $n = 60$ $n = 80$ $n = 100$

Fig. 1. Hierarchy of segmentations on the same image (# of segments n from 30 to 100 in steps of 20)

We express the optimization problem in dual form so that the maximum-margin hyperplane, and consequently, the decision function is only a function of the support vectors (the training data that lie on the margin). Also, in the dual-form, we get around actually having to specify the transformation function $\phi(x)$ and rather only need to define the kernel function $K(\cdot, \cdot)$ that seeks to compute the inner-product in that higher-dimensional space. This formulation is given in Eqn. 4

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\alpha^T \mathbf{Q} \alpha - \mathbf{e}^T \alpha \\ \text{subject to} \quad & \mathbf{y}^T \alpha = 0, 0 \leq \alpha_i \leq C, \textstyle\sum_{j \in m_i} \frac{y_{ij}+1}{2} \geq 1 \end{aligned} \tag{4}$$

where $\mathbf{e} = [1, .., 1]$ is a vector of all ones, $\mathbf{Q}$ is a positive semi-definite matrix such that $\mathbf{Q}_{mn} = y_m y_n K(x_m, x_n)$, and $K(x_m, x_n) = \phi(x_m)^T \phi(x_n)$ is the kernel function that computes the inner product in kernel space. By solving this dual formulation and using the primal-dual relationship, the final $\hat{\mathbf{w}}$ of the hyperplane can be written in terms of the support vectors as given in Eqn. 5

$$\hat{\mathbf{w}} = \sum_{i \in \mathrm{SV}} y_i \alpha_i \phi(x_i) \tag{5}$$

where all x_i corresponding to $\alpha_i > 0$ are the support vectors. The decision function to perform the classification will now be given as

$$sgn\left(\mathbf{w}^T \phi(x) + b\right) = sgn\left(\sum_{i \in \mathrm{SV}} y_i \alpha_i K(x_i, x) + b\right) \tag{6}$$

Once this classification model is learned, the decision hyperplane parameters $\mathbf{w}, b$ can be used to classify each segment instance. Therefore, when presented with a new image, the probability that an image segment within it (with feature vector x_{ij}) contains the learned target can be calculated from Eqn.6 using $\hat{\mathbf{w}}, b$.

While in the standard supervised setting, the labels y_{ij} of each of the examples are known, in the present case, they are unknown integer variables only related by the constraint in Eqn. 1. Therefore, this leads to a mixed-integer programming problem where we are trying to maximize the soft-margin criterion over all possible label assignments and all possible hyperplanes. In essence, the program seeks to find a hyperplane such that there is at least one instance from each bag on the positive side of the space and at the same time, maximizes the margin with respect to the labeling that the hyperplane assigns to the instances in the bags.

This formulation poses a computational challenge since it is a typical mixed integer programming problem that is well known to be NP-hard and cannot be solved efficiently using existing solvers. Therefore, we develop an iterative heuristic approach using prior labeling information that we obtain from the scene modeling technique described in the next section.

4 Scene Modeling for Instance Priors

In this section, we develop a technique to build statistical scene environment models using the Expectation-Maximization algorithm that enables us to assign prior probabilities to parts of the scene as belonging to the background or foreground. Note that this is a one-time modeling technique to assign probabilities using *a priori* scene information as opposed to employing run-time assumptions such as manual initializations or motion models performed by previous work in tracking.

We first move the camera to a set of random pan-tilt locations in the scene (≈ 25) and run the segmentation algorithm on each of the collected images (see Fig. 2(a)-(b)). Since it can be seen from these segmentation results that most

Fig. 2. (a)-(b) Segmentation across the scene showing mostly background segments. (c)-(d) Nearest neighbor classification of segments using the GMM model. Color denotes cluster assigned and intensity of color corresponds to probability value of belonging to that cluster. (e) Variance map showing the intensity variance value in each segment of (a). Here we see that "noisy segments" show high variance. This is used in the weighting scheme for the foreground-background probability assignment.

of the segments belong to parts of the background, we then run a clustering algorithm on these segments to obtain clusters that are representative of scene background entities.

Our next step is to build gaussian probability distributions corresponding to each of the clusters so that we can then map any individual segment from a given image to one of these clusters, and then using its probability density function, assign it a probability of belonging to that cluster. To achieve this, we use the Expectation-Maximization (EM) algorithm to obtain a GMM distribution with the scene segmentation data using a color-based feature representation (mean YCbCr). Instead of selecting an arbitrary value for the number of clusters in the distribution, we automatically select from different models, the model that maximizes the Bayesian Information Criterion (BIC) [17]. By analyzing the segments in each cluster, we observed that the main four clusters emerging from the scene were the ones corresponding to environmental features such as streets (shown in yellow), walkways (shown in blue), buildings (red), and vegetation/grass (green).

After this, each segment in a new image is classified using a nearest neighbor scheme in YCbCr space using Mahalanobis distance. Further, the Gaussian probability distribution function (mean and variance) corresponding to that cluster assignment is used to obtain the corresponding generative probability as shown in Eqn. 7.

$$p_i^{gaussian} = \exp\left(\frac{-(x_{seg_i} - \mu_k)^2}{2\sigma_k^2}\right) \tag{7}$$

where x_{seg_i} is the feature vector corresponding to the ith segment seg_i, μ_k is the mean of its closest cluster k, and σ_k is its corresponding standard deviation.

Figure. 2(c)-(d) shows results for test images where each segment is assigned to its closest background cluster and a corresponding probability value is assigned.

The nature of any segmentation algorithm is that it essentially forms segment borders where there are discontinuities in color, and hence any segment that basically contains a large variance in color values is most likely to be a result of incorrect segmentation around that area. As we can see in the variance map in Fig. 2(e), the segments from the original image which have a greater mix of multiple objects exhibit higher variance in intensity (lighter shades of gray) whereas background segments which exhibit a more uniform color have lower variance (darker shades). We exploit this intuition by employing a Laplacian-based weighting formulation in the scene model such that we assign lower weights to segments having a higher variance, in its contribution to the probability of that segment being a background segment (as given in Eqn.8)

$$\beta_i = \exp\left(\frac{-\sigma_{seg_i}^2}{\sigma_n}\right) \tag{8}$$

where β_i is the variance-based weight assigned to segment i whose intensity variance is $\sigma_{seg_i}^2$, and σ_n is the standard scaling parameter for the Laplacian-based weighting scheme that is manually picked.

These variance-based weights β_i are used along with gaussian-based probability values $p_i^{gaussian}$ to obtain a final weighted-probability for each segment, which indicates the probability that the segment belongs to the scene background.

$$p_i^{bg} = \beta_i \cdot p_i^{gaussian} \tag{9}$$

Some results of this two step probabilistic scene modeling from a tracking sequence are shown in Fig. 3. As seen in Fig. 3(b), the gaussian probability calculation step results in low prior assignments for the foreground target as compared to most of the background segments (which receive higher values). But after the variance weighting (Fig. 3(c)), this contrast is further enhanced with the foreground target receiving very low values whereas most of the background segments retain their high values from the previous step (see Fig. 3(d)), barring a few segments which have relatively higher variance due to inaccurate segmentation.

(a) Segmented Image (b) Gaussian probability (c) Variance weighting (d) Final probability

Fig. 3. Two step weighted-probability assignment for foreground-background separation. The background probability values before (b) and after variance weighting (d) are shown by heatmaps to illustrate the change in absolute probabilities.

Once these foreground-background prior probabilities are obtained for each segment, these are then employed in the mixed-integer SVM formulation to relax the problem using an iterative heuristic approach and make it tractable as explained in the next section.

5 One-Class MIL Algorithm

The iterative approach to solve the One-class MIL problem formulation is based on the idea that, given the integer label variables, the mixed-integer programming problem reduces to a quadratic programming problem that can be solved efficiently. Therefore, we use the background prior probabilities for each of the segment instances, and using a threshold of 0.5, convert them to instance labels for initialization.[1] Once these initial values for the integer variables are obtained, we solve the corresponding QP problem to obtain the hyperplane parameters $\mathbf{w}$ and b.

[1] Likewise, for other applications, one could employ simple techniques (e.g. supervised approach) or domain information to achieve priors.

Now using this hyperplane, we re-classify each of the instances to obtain new labels for each of them. Following this, we impose the MIL constraint that there should be atleast one positive instance in each of the bags. To do this, we go through each bag, and if it is found that a bag has no instance classified as positive according to the new hyperplane, then we re-label its *most positive* instance (the one closest to the hyperplane) as being positive. Once this re-labeling is performed across all bags, we re-initialize the QP problem and solve for a more optimal hyperplane, and again check for the MIL constraint. This iterative procedure is repeated till the algorithm converges on the labeling so that the instance labels do not change anymore.

Once the algorithm converges, the hyperplane obtained $\mathbf{w}^*, b^*$ represents the classifier that maximizes the soft-margin criterion as well as satisfies the Multiple Instance constraint across all positive bags. This procedure is summarized in Algorithm 1.

Input: Training data: positive bags $D_{oc} = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$, background priors $p_{ij}^{bg}, i \in N, j \in m_i$ for each instance

Initialize labels $y_{ij}^0 = 1 \; \forall j \in \{j, p_{ij}^{bg} > 0.5\}$ and $y_{ij}^0 = -1 \; \forall j \in \{j, p_{ij}^{bg} \leq 0.5\}$

repeat iterate k

Solve quadratic program (QP) in Eqn. 4 using labels y_{ij}^k to obtain hyperplane $\mathbf{w}^k, b^k$

Compute output values $f_{ij} = \mathbf{w}^T \phi(x) + b \; \forall i, j$

Assign labels $y_{ij} = sgn(f_{ij})$

forall the $i \in N$ **do** iterate i

`/*Check if all instances in the bag are negative*/`

if $\sum_{j \in m_i} \frac{y_{ij}+1}{2} == 0$ **then**

`/*Re-label the most positive instance to 1*/`

$\hat{j} = \arg\max_j f_{ij}$

$y_{i\hat{j}} = 1$

end

end

until $y_{ij}^k == y_{ij}^{k-1} \; \forall i, j$;

$\mathbf{w}^* = w^k, b^* = b^k$

Output: Maximum margin hyperplane $\mathbf{w}^*, b^*$

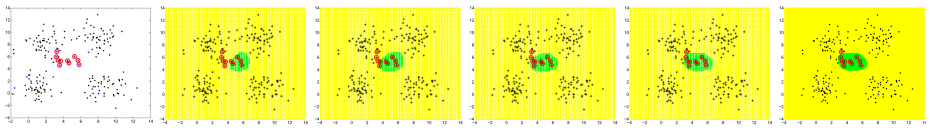
Algorithm 1. Iterative SVM algorithm for one-class MIL

The iterative heuristic employed here is a well-known technique to reduce a Mixed-Integer programming problem to an efficient Quadratic problem and empirical convergence has been shown in many domains/datasets such as Bioinformatics (MUSK datasets), Text (TREC9, MEDLINE), Images (COREL) etc [7]. We also demonstrate empirical convergence in our datasets in the following

experiments. Theoretical proof of convergence is a more general problem out of the scope of this paper.

6 Experiments

In this section, we demonstrate the validity and the performance of our one-class MIL algorithm on synthetic data and real tracking sequences.



(a) Original (b) Step 1 (c) Step 2 (d) Step 3 (e) Step 4 (f) Final

Fig. 4. Different steps of running Algorithm 1 on synthetic one-class MIL data. Blue asterisks - positively labeled, Black dots - negatively labeled, Red circles - ground truth positive class.

6.1 Synthetic Data

We first performed experiments with synthetically constructed positive class and noise data explaining how the construction of data simulates the typical one-class MIL problem setting, and then validate the performance of our algorithm on that data.

For the positive class, we sample from a Gaussian distribution of known mean and variance (points marked with a red circle in Fig. 4(a)). Since in the real world, noise is essentially constituted by samples from a number of different sources, to simulate noise, we sample from a number of different Gaussian distributions which are present in different parts of the space (represented as black dots in the figure). To construct the bags (every bag being positive), we pick one sample from the positive class and remaining samples (around 20) from the noise class. To simulate the presence of priors, we associate labels for each of the sample points based on their source class, but with the introduction of randomly assigned false positives (blue asterisks in the noise region of the space) and false negatives (black dots marked with red circles in the positive class region of the space).

The figure in Step 1 is the first step classifier that is learned by solving the quadratic program (QP) in Eqn. 4 using the initial priors. As we can see, the positive space (in green) does not fully cover all the initial positive examples and is inhibited from doing so by the strong presence of noise all around. This is essentially the typical problem with the Multiple Instance setting, where the noise instances truly dominate the given data and with very sparse positive data being actually present in the positive bags. The iterative process of MIL constraint checking, instance re-labeling, and updating the classifier is run until

the labels do not change any more (final classification space shown in Fig. 4(f)). As we can see, the classification boundary at the end of every iteration is pulled closer to the true boundary with the reinforcement of the MIL constraint from Eqn. 1. Thus, overall we conclude that, at the end, the one-class MIL algorithm learns a classification boundary that covers the positive class data very well even in the presence of a large amount of noise surrounding the data. This experiment provides a clear understanding of the running of our one-class MIL algorithm, validating its ability to learn the positive class concept in the presence of high percentage of noisy data in each of the input bags.

Having learned a non-linear classifier (the maximum-margin hyperplane $\mathbf{w}^*, b^*$ from Algorithm 1), we next quantify its testing accuracy on different synthetic datasets by comparing the classifier's labeling with the ground truth labels of the examples, and then compare its performance with a standard supervised version of SVM. The classification accuracy for different datasets is shown in Table. 1. From this we observed that for each of the datasets, our one-class MIL algorithm performed better than its supervised counterpart. This was as expected primarily because the supervised learner fails to employ the MIL constraint which is the defining characteristic of this problem domain, thus demonstrating the need for a one-class MIL algorithm for such problems.

Table 1. Comparison of classification accuracy of the classifier learned from Algorithm 1 on different datasets (using Polynomial and RBF kernel functions)

Dataset	#InstPerBag / #Bags / #Dims	One-Class MIL (%)		Supervised (%)	
		Poly	RBF	Poly	RBF
DS_{small}	20 / 10 / 2	98.9	99.4	75.2	74.3
DS_{medium}	250 / 50 / 75	96.5	97.7	68.5	67.2
DS_{large}	5,000 / 200 / 1,000	94.3	96.1	59.4	61.2

6.2 Comparison of One-Class and Two-Class MIL

The goal of this experiment was to explore and compare the performance of the one-class approach with the well-studied two-class approach as we changed the prior threshold values.[2]

We first created bags containing synthetic data from the positive and negative class (positive bags) similar to the previous experiment. To simulate scene priors, we assign prior probabilities by sampling from Gaussians with (μ, σ) of $(0.75, 0.1)$ - positive class and $(0.25, 0.1)$ - negative class. Picking a prior threshold value t from $[0, 1]$ for initialization, we then ran the one-class MIL algorithm. For the two-class algorithm, we used t to split the data into positive bags ($\geq t$) and negative bags ($< t$). After both algorithms learned their target spaces (classification boundaries), we then measured the true positive (TP) and false positive (FP) rates and calculated the *F1 score* for each of them. This experiment was

[2] We compared our performance with the state-of-the-art two-class MIL algorithm of [3], but the discussion holds for any other two-class MIL approach.

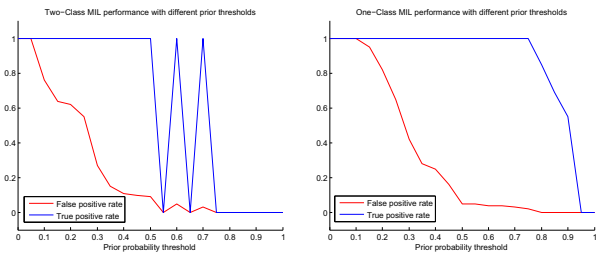
repeated for t between 0 and 1 in steps of 0.1, and above mentioned scores were calculated in each case.

Table 2 plots show the false positive and true positive rate results for the Two-Class and the One-Class MIL algorithm for different values of prior thresholds. From these, the performance of the two algorithms can be analyzed as follows.

For the two-class MIL algorithm, for low values of the t, all the positive instances along with most of the negative instances are put in the positive bags and the remaining few negative instances are put in the negative bags. Consequently the algorithm learns multiple concepts across the entire space since they satisfy the MIL criterion of persistence across positive bags. This results in a high FP rate (and also a high TP rate since the true positive is also learned). However, as t is increased, the negative class concepts are not learned anymore since one or more of these instances end up in the negative bags. Consequently the false positive rate drops quickly.

Since the prior probabilities are randomly assigned from gaussians, it could so happen that in one (or more) bags the true positive instances have a prior lower than t. Therefore, as t is further increased (beyond 0.5), we see that this results in such an instance ending up in the negative bag. Due to the this, no target concept is learned by the algorithm since no concept is consistently found across positive bags. This causes the TP rate to drop to 0 as seen in the figure. However, since the prior assignment is performed at random for a threshold value, if it happens to be the case that there are true positives in all positive bags, then the target concept is learned and a high TP rate is obtained. However, this behavior is unstable as seen in the figure (see blue line beyond 0.5) and is completely dependent on the initial prior value assigned. This can also be observed in Table 2, where we see that for many of the threshold values beyond 0.5, the target concept is not learned and the behavior is unstable. Therefore, this suggests that the two-class MIL algorithm is sensitive to the threshold picked to create the pairs of positive and negative bags from the prior probabilities.

Table 2. Plots show performance of the Two- and One-Class MIL algorithm for different values of prior thresholds. Table shows F1 scores, × indicates no model was learned.



Threshold	2-Class	1-Class
0.10	0.724	0.667
0.20	0.763	0.709
0.30	0.881	0.826
0.40	0.947	0.888
0.50	0.956	0.970
0.60	×	0.975
0.70	0.985	0.985
0.80	×	0.850
0.90	×	0.550
1.00	×	×

In the one-class approach, for small values of t, all positive and most negative instances were assigned positive labels and very few negative instances obtained negative labels. Solving the QP with this labeling would result in a classification space with high FP rate and full TP rate as seen in Table 2. Now, as t is increased, we observe a smooth decrease in the FP rate since more instances would likely be labeled negative (TP rate stays high). However, as t approaches and crosses 0.75 (mean for the positive gaussian), the likelihood that *every* bag contains an instance from the positive class reduces and therefore the most positive instance is relabeled for each bag. When the algorithm converges, the classification boundary need not necessarily cover the entire true positive space and the TP rate drops a bit as seen in the figure. Further, for very high t (beyond 0.95), since it is very unlikely to have reasonable number of positively initialized instances, the QP from the first step itself fails to learn a classification boundary, and no target concept is learned (as seen in Table 2).

Therefore, overall from the F1 scores and graphs, we conclude that the two-class MIL approach is sensitive to the method of creation of positive and negative bags (since that imposes a hard constraint), especially if it is done using a threshold on the priors. The one-class MIL approach, on the other hand, is more robust to this procedure since the threshold only initializes the instance labels (a soft constraint) and the iterative MIL approach ensures that the true classification boundary is subsequently approached and learned even in this case.

6.3 One-Class MIL with Tracking Data

In this section, we demonstrate how important applications such as object tracking can significantly benefit from this novel paradigm in machine learning of One-class MIL. For this, we performed experiments with tracking sequences from standard surveillance cameras and further evaluated the one-class MIL approach.[3]

For each image in the training sequence, we perform a multi-level segmentation to create positive bags, used the scene modeling scheme described in Sect. 4 to initialize the instance labels and run the first iteration of the QP in Eqn. 4. Similar to the synthetic data case, the Algorithm 1 was run until the labels did not change anymore. Fig. 5-left side shows the classifier score from Eqn. 6 (normalized to $[0, 1]$) for each of the segments overlaid on the test image at different iterations of Algorithm 1. As we can see, during the initial iterations, the classifier is not very accurate and exhibits a few false positives. At the same time, the target of interest also only receives a low score (implying the target instance is as far away from the classification boundary as some of the other instances). But as the algorithm progresses, the classifier boundary is refined, gradually improving the score for the target instance and simultaneously reducing the false positive probabilities, until the algorithm converges.

Figure 5 shows the results for running the one-class MIL algorithm on different video sequences where a heatmap of the final classifier score is overlaid on the

[3] We do not claim that object tracking is purely a One-class MIL problem. In fact, a complete robust tracking system would certainly need other components as well.

segmented testing images. Note that, the image is tested at multiple segmentation levels and scores are obtained for the segments at each of these levels, but the heatmap shown here is the one corresponding to the level containing the highest scoring segment. For each of the sequences, we can see that the algorithm is able to localize and track the target of interest. The approach works in real-time (at 10fps) since tracking is performed online with active/moving PTZ cameras.

These results demonstrate the applicability of employing the one-class MIL approach for localizing the target of interest without having to make any motion assumptions on the part of the targets. However, a background prior model is required to obtain prior instance probabilities to bootstrap the one-class learning algorithm. While this is a tradeoff in assumptions, from a practical standpoint we believe that building a scene model is more feasible, since it is a one-time task performed beforehand and affords an added flexibility in applications by obviating the need to make run-time assumptions such as motion models or manual initializations.

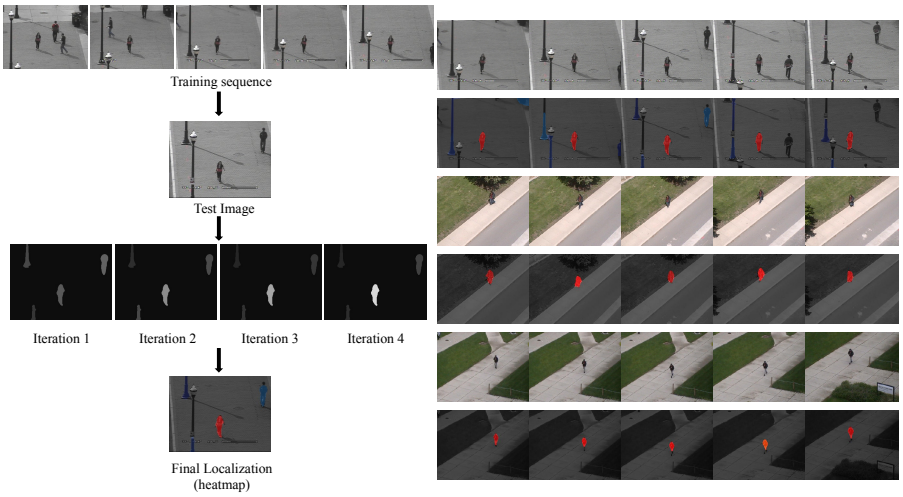


Fig. 5. Left - Step-wise illustration of running the one-class MIL Algorithm 1 on a tracking sequence. Right - Target Localization on different tracking video sequences. Top row shows test images for each sequence, and bottom row the corresponding localization results.

7 Summary and Future Work

We presented a novel analysis of the one-class version of Multiple Instance Learning problem (where one is only provided input data in the form of positive bags), and developed an SVM-based formulation to solve this problem setting. We addressed the target tracking problem by building positive bags with video sequences, segmenting each image in the sequence to obtain instances, so that

the intended target model is learned by exploiting the idea of "commonality" across all input frames. We further developed a scene prior modeling technique to obtain foreground-background instance priors. Exploiting these priors we developed an iterative heuristic approach to solve the SVM formulation. We showed the validity of our approach with synthetic data, compared it with the two-class approach, and then demonstrated its performance on standard tracking sequences. In future work, we are working on other interesting applications for our theoretical one-class MIL framework.

References

1. Chen, Y., Bi, J., Wang, J.: Miles: Multiple instance learning via embedded instance selection. IEEE TPAMI (2006)
2. Babenko, B., Yang, M., Belongie, S.: Visual tracking with online multiple instance learning. In: Proc. IEEE CVPR (2009)
3. Sankaranarayanan, K., Davis, J.: Object association across ptz cameras using logistic mil. In: Proc. IEEE CVPR (2011)
4. Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13, 2001 (1999)
5. Dietterich, T., Lozano-Perez, T.: Solving the multiple-instance problem with axis-parallel rectangles. In: Artificial Intelligence (1997)
6. Maron, O., Lozano-Perez, T.: A framework for multiple instance learning. In: Proc. Neural Information Processing Systems, NIPS (1998)
7. Andrews, S., Hofmann, T.: Support vector machines for multiple instance learning. In: Proc. Neural Information Processing Systems, NIPS (2003)
8. Fu, Z., Robles-Kelly, A., Zhou, J.: Milis: Multiple instance learning with instance selection. In: IEEE TPAMI, pp. 958–977 (2011)
9. Mu, L., Kwok, J., Bao-Liang, L.: Online multiple instance learning with no regret. In: Proc. IEEE CVPR (2010)
10. Viola, P., Platt, J., Zhang, C.: Multiple instance boosting for object detection. In: Proc. Neural Information Processing Systems, NIPS (2005)
11. Ester, M., Kriegel, H., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: ACM Conference on Knowledge Discovery in Databases SIGKDD (1996)
12. Tax, D.M.J., Duin., R.P.W.: Outliers and data descriptions. In: Proceedings of the 7th Annual Conference of the Advanced School for Computing and Imaging (2001)
13. Crammer, K., Chechik, G.: A needle in a haystack: local one-class optimization. In: Proc. ICML (2004)
14. Gupta, G., Ghosh., J.: Robust one-class clustering using hybrid global and local search. In: Proc. ICML (2005)
15. Crammer, K., Talukdar, P., Pereira, F.: A rate-distortion one-class model and its applications to clustering. In: Proc. ICML (2008)
16. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. International Journal of Computer Vision (IJCV), 7–27 (2001)
17. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics, 461–464 (1978)