Online Multiple Classifier Boosting for Object Tracking

Tae-Kyun Kim, Tom Woodley,
Björn Stenger, Roberto Cipolla

CUED/F-INFENG/TR631

19 June 2009

Department of Engineering
University of Cambridge
Trumpington Street
Cambridge CB2 1PZ, UK

{tkk22|tew32|cipolla}@eng.cam.ac.uk

Online Multiple Classifier Boosting for Object Tracking

Tae-Kyun Kim[1], Tom Woodley[1], Björn Stenger[2], Roberto Cipolla[1]

[1] : Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, UK.
[2] : Toshiba Research Europe, Computer Vision Group, Cambridge, CB4 0GZ, UK.

Abstract– This paper presents a new multi-classifier boosting algorithm for automatically learning object appearance models. In many cases the appearance model is multi-modal and therefore a number of strong classifiers are learned to handle different appearances. The algorithm is automatic in that it *jointly* learns the classifiers and a soft partitioning of the input space, defining a distinct area of expertise for each classifier. We show how the new formulation improves the specificity of the strong classifiers leading to a more efficient use of weak classifiers compared to previous learning methods. Motivated by visual tracking applications we propose an on-line scheme to iteratively adapt the classifiers during tracking. Experiments show that the algorithm is able to learn multi-modal appearance models that can be used to detect rapid pose changes in the test sequences.

Keywords– Multi-classifier Boosting, Mixture of Experts, Visual Tracking, On-line Learning, On-line Boosting, Pose Estimation, Clustering

1 Introduction

In object tracking a major challenge is handling appearance changes of the target caused by pose or lighting changes. Recently a class of techniques using discriminative tracking has been shown to give good results by treating tracking in a classification framework [1, 4, 6]. A classifier is iteratively updated using positive and negative training samples extracted from each frame. One inherent difficulty of pure on-line trackers is setting their degree of adaptation, leading to an accuracy vs. drift trade-off [1]. Rapid adaptation allows the tracker to tolerate rapid pose changes, but also increases the risk of incorrectly adapting to background regions. A solution to this problem is to maintain an object model that is learned prior to tracking and which limits the adaptation to regions that do not deviate

too much from the model. For some cases a standard binary boosted classifier is sufficient, e.g. for frontal pose faces [16, 20, 22]. In many cases, however, the target model has to be multi-modal, represented for example by appearance manifolds [14] or a set of classifiers [11, 15]. In the latter case typically distinct appearance clusters are first labeled and a separate classifier is then learned for each [15, 19]. In order to do without the labeling step recent methods for multi-classifier boosting approach the problem by *jointly* clustering and training multiple classifiers for the positive class [2, 13]. These techniques have shown good results on learning multi-pose classifiers for object detection, requiring less weak learners to achieve the same performance. However, the application of these techniques to multi-pose object tracking is not straightforward. The main reason is that in an on-line setting the number of positive and negative samples is not sufficient to ensure a good partitioning of the input space in terms of classifier expertise. As a result the use of weak classifiers is globally sub-optimal. We propose a modification of the multi-classifier boosting algorithm by introducing a weighting function $\mathcal{Q}$ that enforces a soft split of the input space. In addition, we present an on-line version of the algorithm to dynamically update the classifiers. Finally, the method is applied to object tracking where it is used to rapidly learn different appearance clusters in a short calibration stage prior to tracking, thereby solving the drift problem during on-line tracking.

The paper is organized as follows. We first review previous work in multi-classifier learning and object tracking. In section 3 we derive a new algorithm for multi-classifier boosting, MCBQ, using the weighting function $\mathcal{Q}$ in order to learn multi-modal appearance models. Section 4 shows how the method is applied in a tracking setting by using a short calibration sequence to learn the appearance model and subsequently using an on-line update scheme. In the results section 5 we compare the proposed MCBQ with standard boosting and the MCBoost algorithm in [13].

2 Relationship to previous work

This paper is related to the areas of adaptive object tracking as well as multi-classifier learning. We therefore discuss work from both areas here. A number of on-line adaptation schemes have been proposed for object tracking: Jepson et al. proposed to update the appearance model using an on-line EM algorithm in their WSL-tracker [10]. Collins et al. introduced the concept on-line feature selection for tracking, where at each time step the features that best discriminate between foreground and surrounding background are chosen to compute likelihood maps [4]. Avidan proposed Ensemble Tracking, in which a small number of weak classifiers are combined using AdaBoost [1]. Grabner and Bischof introduced an on-line boosting scheme where features are selected from a larger pool of weak classifiers and combined into a strong classifier [6]. On-line schemes without any target model tend to suffer from two drawbacks: Firstly, they cannot handle rapid appearance changes. Secondly, alignment errors in the training samples accumulate and lead to drift. Heuristics to avoid adaptation to outliers have been included in [1, 4], but a more principled approach is to introduce a prior model as a way to guide the tracker.

The approach employed first by Jebara and Pentland is to verify the tracker output with a classifier trained to detect the target object [9]. More recently, Okuma et al. and Li et al. have used the output of a boosted detector in their likelihood models [16, 18]. Grabner et al. have proposed using a boosted detector (or alternatively a single image taken at the beginning of the sequence) as a prior to their on-line boosting scheme. However, in order to handle multiple appearances of the target, e.g. due to pose changes, a multi-modal object model is required. Standard boosted classifiers are generally not sufficient, and the classical way is to collect labeled data for multiple classes and train a separate classifier for each class [11, 15]. Recently, Babenko et al. and Kim and Cipolla have proposed a multi-classifier boosting method to simultaneously cluster the data and train strong classifiers [2, 13]. These methods have so far been used for detection applications where the full training set is available from the beginning. In section 5 we show how discriminative clustering in these algorithms on sparse data does not guarantee a good separation of the classifiers. Lee and Kriegman modelled a general object appearance manifold by a collection of subspaces, which was subsequently updated during tracking [14]. Our method is close to this work in that we employ a multi-modal appearance model that is updated on-line. However, since we cast the problem in a discriminative framework, significantly fewer samples are required to construct the model allowing us to simply train on a short calibration sequence. Other related work is multiple instance boosting by Viola et al. [21] as well as its recent on-line version [3]. Here a single detector is learned jointly with alignment parameters. In contrast to this work, we train multiple classifiers for the positive class. Our work also builds on the Mixture of Experts model by Jordan and Jacobs [12], which is a tree-structured model for classification that is based on soft clustering of the input space. Similar to their work, we jointly learn a weighting function $\mathcal{Q}$ that weights the contributions of each classifier.

3 Multi-Classifier Boosting with Clustering

This section first briefly reviews multi-classifier boosting as proposed by [2, 13], then details our improvements in the MCBQ algorithm. In both cases the following notation is used: Given is a set of n training samples $\mathbf{x}_i \in \mathcal{X}$, where $\mathcal{X}$ is the input domain (in our case image patches), with corresponding labels $y_i \in \{-1, +1\}$ corresponding to object/non-object. Additionally, each of the object samples can be considered belonging to one of K perceptual groups where the class membership is a priori unknown.

3.1 Multi-Classifier Boosting

In order to discriminate between object and non-object a boosting framework is used to train K strong classifiers H^k, where

$$\mathrm{H}^k(\mathbf{x}) = \sum_t \alpha_t^k \, h_t^k(\mathbf{x}), \qquad k = 1, ..., K, \tag{1}$$

and h_t^k is the t-th weak classifier of the k-th strong classifier weighted by α_t^k. Each weak classifier comprises a simple visual feature and threshold, and each strong classifier $\mathrm{H}^k(\mathbf{x})$ is trained through the rounds of boosting to concentrate its expertise on one of the K groups. The key is the use of a *noisy OR* function to combine the strong classifiers to give the overall classification response. This function classifies a sample as positive if any of the K strong classifiers does so, and negative otherwise:

$$p(\mathbf{x}) = 1 - \prod_k (1 - p^k(\mathbf{x})), \tag{2}$$

where $p^k(\mathbf{x}) = 1/(1 + \exp(-\mathrm{H}^k(\mathbf{x})))$.

Following standard AdaBoost [5], a distribution of weights for the training samples is maintained, one distribution per strong classifier, and at each round the algorithm chooses a new weak classifier h_t^k with associated weight for each strong classifier, and updates the sample weights for the next round. For given weights, the algorithm finds K weak classifiers at the t-th round of boosting, to maximize

$$\sum_i w_i^k \, h_t^k(\mathbf{x}_i), \qquad h_t^k \in \mathcal{H}, \tag{3}$$

where $h_t^k \in \{-1, +1\}$ and $\mathcal{H}$ is a set of weak classifiers. The weak classifier weights $\alpha_t^k, k = 1, ..., K$ are then found by minimizing $\mathcal{L}(\mathrm{H} + \alpha_t^k h_t^k)$ by line search, where $\mathcal{L}$ is a loss function. Applying the AnyBoost method [17], the sample weights are set as the negative gradient of the loss function $\mathcal{L}$ with respect to the classifier score. Choosing $\mathcal{L}$ to be the negative log likelihood, the weight of k-th classifier over i-th sample is updated by

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \, p^k(\mathbf{x}_i). \tag{4}$$

For choosing the number K of strong classifiers it is suggested in [13] to try large values first and select K as the number of visually distinctive clusters.

3.2 Introducing the Partitioning Function $\mathcal{Q}$

Multi-classifier Boosting has the ability to create strong classifiers with non-intersecting areas of expertise. However, its ability to do so relies heavily on the training data set containing negative samples which separate the positive samples into distinct regions in the classifiers' discriminative feature space. This also means that there are no guarantees of pose-wise clustering for poses as we would perceive them by viewing the images. In fact there is no constraint in the algorithm that enforces strong classifiers to focus on a unique area of expertise, and there is no concept of a metric space on which perceived clusters can be formed. We make the classifier assignment explicit by defining a function $\mathcal{Q}^k(\mathbf{x}) : \mathcal{X} \rightarrow [0, 1]$ which weights the influence of strong classifier k on a sample $\mathbf{x}$. By mapping $\mathbf{x}$ into a suitable metric space, we can impose our desired clustering regime on the training set, thus $\mathcal{Q}$ defines a soft partitioning of the input space. The choice of $\mathcal{Q}$ is dependent on the application domain. In principle any function can be used that captures the structure of the input domain, i.e. that maps the samples to meaningful clusters. In this paper $\mathcal{Q}$ is defined by a K-component Gaussian mixture model in the space of the first d principal components of the training data. The k-th GMM mode defines the area of expertise of the k-th strong classifier. The GMM is updated in using EM alongside the weak classifiers in the boosting algorithm as follows:

Algorithm 1 Updating Partitioning Function $\mathcal{Q}$

1. Calculate the likelihood of each of the samples under the k-th strong classifier, $R^k(\mathbf{x})$

2. Set the new probability of the sample being in the k-th GMM component as its current $\mathcal{Q}$ value scaled by the likelihood from the classifier. $p(x) = \mathcal{Q}^k(\mathbf{x})\,\mathcal{R}^k(\mathbf{x})$

3. Update the cluster center as the mean value of the samples under this probability.

The new noisy-OR function in Equation 2 now becomes:

$$p(\mathbf{x}) = 1 - \prod_k (1 - \mathcal{Q}^k(\mathbf{x_i})\, p^k(\mathbf{x})), \tag{5}$$

leading to the new weight update equation:

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)}\, \mathcal{Q}(\mathbf{x_i})\, p^k(\mathbf{x}_i). \tag{6}$$

The full MCBQ algorithm is summarized in Algorithm 2. Note that compared to the original multi-classifier boosting algorithm additional steps 1, 2, and 7 are required and step 6 is modified.

Algorithm 2 Multi-classifier Boosting with Partitioning Function $\mathcal{Q}$(MCBQ)

Input: Data set $(\mathbf{x}_i, y_i)$, set of pre-defined weak learners.
Output: Multiple strong classifiers $H_k(\mathbf{x})$, partitioning function $Q_k(\mathbf{x})$.
1. Initialize Q with a Gaussian mixture model
2. Initialize weights w_i^k to the values of $Q(\mathbf{x}_i)$.
3. Repeat for $t = 1, ..., T$
4. Find weak learners h_t^k maximizing $\sum_i w_i^k \mathrm{h}_t^k(\mathbf{x}_i)$.
5. Compute weights α_t^k maximizing $\mathcal{L}(\mathrm{H}^k + \alpha_t^k \mathrm{h}_t^k)$.
6. Update weights by Equation 6.
7. Update partitioning function Q by Algorithm 1.
8. End

4 Application to Tracking

The goal in the application to tracking is to learn an object-specific appearance model to guide the tracker. In the simplest case, object and non-object samples can be collected from a large number of images. This is a similar setting as the one for object detection described in [2, 13]. Such a detector can the be used in any tracking-by-detection framework, e.g. [8], where the detector response is given by combining the noisy OR function in Equation 2. However, in the tracking setting we would like to use a short calibration sequence only in which the range of appearances is shown to the camera as in [14]. This limits the number of training samples, but is enough to bootstrap the classifier. Subsequently, we would like the tracker to remain flexible to some appearance changes while using the learned model as an anchor. This motivates the following approach. We propose an algorithm to iteratively adapt multiple strong classifiers with MCBQ.

When not all training data is initially available, traditional boosting methods are unable to create a classifier, since they are unable to create and update a distribution of weights over all samples. In order to move MCBQ into an on-line setting we need a mechanism for rapid feature selection for the weak classifiers, and incremental updating as new training samples arrive. The on-line boosting algorithm addresses this issue, allowing for the continuous learning of a strong classifier from training data arriving one sample at a time [6]. The key step is, at each boosting round, to maintain error estimates from samples seen so far, for a pool of weak classifiers. At each round t a *selector* S_t maintains these error estimates for weak classifiers in its pool, and chooses the one with the smallest error to add to the strong classifier.

To summarize, our tracking algorithm contains two-stages: Firstly, training data is assembled in a supervised learning stage, where the system is given initial samples which span the extent of all appearances to be classified. An initial MCBQ classifier is then built rapidly from this data using a feature pool. Secondly, additional training samples are supplied to update the classifier with new data during tracking.

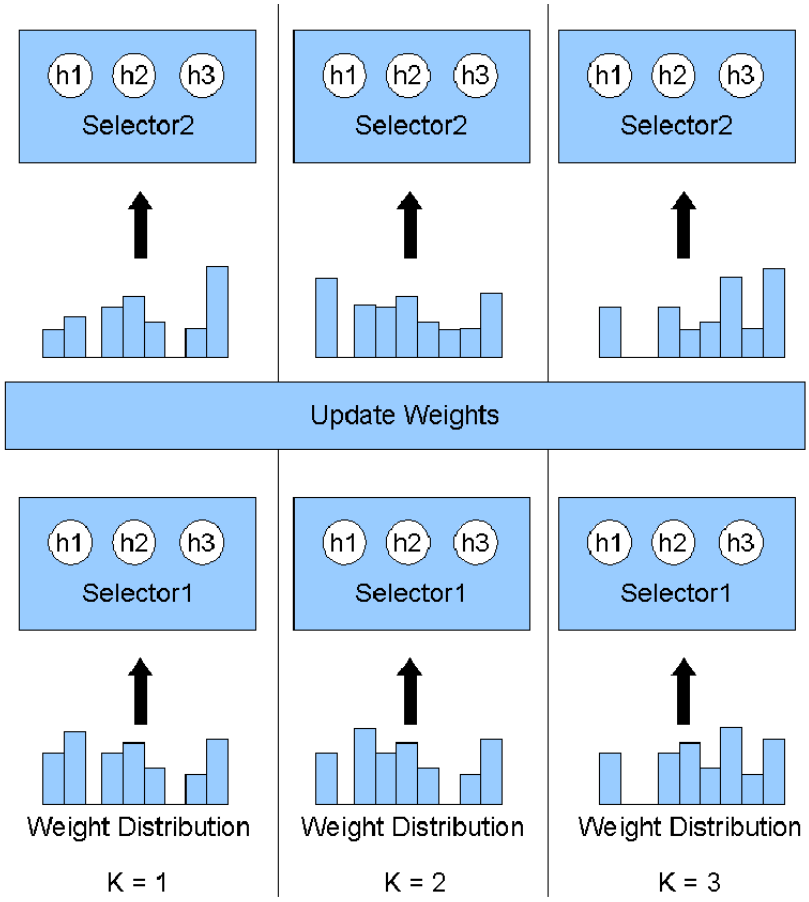


Figure 1: On-line MCBQ schematic. *Shown here is one round of boosting for k strong classifiers H^k. Each strong classifier H^k maintains a distribution of weights over the samples. Selectors choose the best performing weak classifiers on the samples and store them in a pool. At each round of boosting the best-performing weak classifier is added to the strong classifier. The weights are then updated before being passed on to the next row of selectors.*

4.1 Framework

As with off-line MCBQ, our final classifier consists of multiple strong classifiers $\mathrm{H}^k(\mathbf{x})$ made up of weak classifiers $h_t^k(\mathbf{x})$. As with on-line boosting, we need a mechanism for selecting a weak classifier at each boosting round t and strong classifier H^k, based on classification errors for all samples seen so far. We therefore also require *selectors* S_t^k, each with their own pool of weak classifiers to choose from. Figure 1 illustrates the framework.

4.2 Weak Learning and Selection

All our weak classifiers use a single Haar-like feature. For on-line learning of a weak classifier from a feature f and labeled samples $(\mathbf{x}_i, y_i)$ we create a decision threshold θ_m with parity p_m from the mean of feature values seen so far for positive and negative samples, where each feature value is weighted by the corresponding image weight:

$$h_m(\mathbf{x}) = p_m \operatorname{sign}(f(\mathbf{x}) - \theta_m), \tag{7}$$

$$\theta_m = (\mu^+ + \mu^-)/2, \quad p_m = \operatorname{sign}(\mu^+ - \mu^-), \tag{8}$$

$$\mu = \frac{\Sigma_i |w_i| f(\mathbf{x}_i)}{\Sigma_i |w_i|}. \tag{9}$$

The error of the weak classifier is then given as the normalized sum of the weights of mis-classified samples:

$$e = \frac{\sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i^k|}{\sum_i |w_i^k|}. \tag{10}$$

A weak classifier can then be chosen from a pool as the one giving the minimum error.

4.3 Supervised Learning

During the supervised learning stage, we have a set of weighted samples, and a global feature pool $\mathcal{F}$. Weight distributions for each k are initialized to randomly assign positive samples to a strong classifier k, and at each round t and strong classifier k the equations 7, 8, 9, 10 are used to initialize and select a weak classifier based on exact errors. In order to facilitate selection at the incremental update stage, we store in each selector S_t^k, for the positive and negative samples (1) for each feature value, the sum of weights of samples with that value, and (2) the sum of image weights.

To improve speed, each selector keeps only the best N performing weak classifiers for use in the incremental update stage. After each round of boosting, image weights are updated as in Equation 4, and voting weights calculated based on the error of the chosen weak classifier. Algorithm 3 details the process.

4.4 Incremental Update

Once the initial classifier has been created, it can be updated with new samples. Weights for positive samples are initialized based on their classification responses from each of the component strong classifiers in the MCBQ classifier, and the sample is passed through the boosting framework. The summations stored in each selector can be updated from the new sample, and thus the new classification thresholds for the weak classifiers calculated using equations 7, 8, 9. The error values from Equation 10 are used to choose the best

Algorithm 3 On-line MCBQ – Supervised Learning

Require: Labeled data set $(\mathbf{x}_i, y_i)$, $y \in \{-1, +1\}$.
Require: Feature pool $\mathcal{F}$ (initialized randomly).
Require: Initial weight distributions $\{w_i^k\}$, $k = 1, \ldots, K$, one per strong classifier.
  Pre-calculate all feature values for all samples
  // For each round of boosting
  for $t = 1, \ldots, T$ **do**
    // For each strong classifier, initialize selector $\mathbf{S}_t^k$
    for $k = 1, \ldots, K$ **do**
      // Find the best M weak classifiers from the feature values, weights and image labels
      $h_{t,m}^k = \text{Train}(\mathcal{F}, \{y_i\}, \{w_i^k\})$

      Cache weight sums for each feature value f, for positive and negative samples
      // Calculate errors for each weak classifier
      $e_{t,m} = \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i^k|$
      // Choose the weak classifier with the lowest error
      $h_t^k = \text{argmin}_m\,(e_{t,m})$
      // Calculate voting weight
      $\alpha_t^k = \frac{1}{1+\exp\{-\ln\left(\frac{1-e_t}{e_t}\right)\}}$
    end for

    // Update $\mathcal{Q}$ function
    // Update importance weights, then re-normalize
    $w_i^k = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \mathcal{Q}^k(\mathbf{x}) p^k(\mathbf{x}_i)$
  end for

weak classifier to add to the strong classifier. Finally, the worst-performing weak classifier is replaced with a new randomly-generated one. Note that in the case of $\mathcal{Q}$being defined by a Gaussian mixture in the PCA space, we update the PCA space by the algorithm of Hall et al. [7] before updating $\mathcal{Q}$. The incremental update algorithm is summarized in Algorithm 4.

Note our selector structure is equivalent to the storing of the sums of weights of correctly and incorrectly classified samples in the selectors in the on-line boosting algorithm. However, those error estimates do not reflect the changing nature of a weak classifier as it is updated from new samples. By our choice of feature and weak classifier formation we are able to report exact errors for the current threshold/parity of a weak classifier.

Algorithm 4 On-line MCBQ – Incremental Update

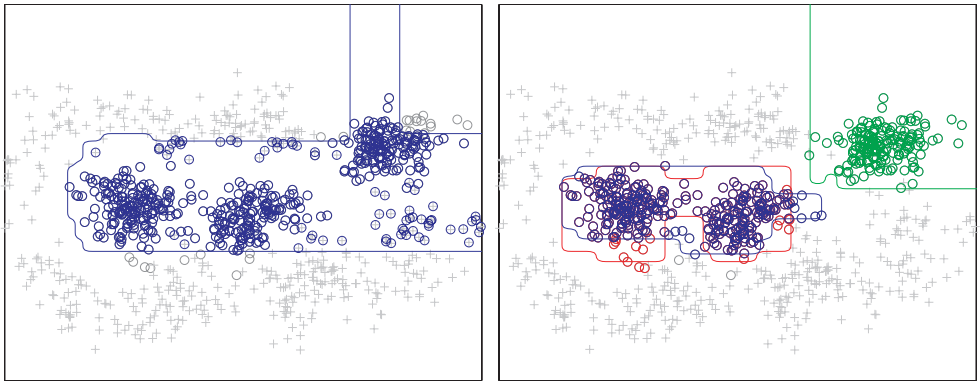
Require: Labeled training image $(\mathbf{x}, y)$, $y \in \{-1, +1\}$.
Require: MCBQ classifier $\mathrm{H}(\mathbf{x})$.
// Initialize sample weight
$w^k = \mathcal{Q}^k(\mathbf{x}) / \sum_k \mathcal{Q}^k(\mathbf{x})$

// For each round of boosting
for $t = 1, \ldots, T$ **do**
  // For each strong classifier, update selector $\mathbf{S}_t^k$
  for $k = 1, \ldots, K$ **do**
    // Update the selector's weak classifiers
    for $m = 1, 2, \ldots, M$ **do**
      // Update cached weight sums from sample's feature value, for positive and negative samples
      // Update classification threshold and parity $\mathrm{Update}\left(h_{t,m}^k, (\mathbf{x}, y), w^k\right)$
      // Calculate new error $e_{t,m}$
      $e_{t,m} = \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i^k|$
    end for
    // Choose the weak classifier with the lowest error
    $m^* = \mathrm{argmin}_m \left(e_{t,m}\right)$
    $h_t^{k*} = h_{t,m^*}^k$
    $e^* = e_{t,m*}$
    // Calculate voting weight
    $\alpha_t^k = \frac{1}{1+\exp\{-\ln\left(\frac{1-e^*}{e^*}\right)\}}$
    // Replace the weak classifier with the highest error
    $m^- = \mathrm{argmax}_m \left(e_{t,m}\right)$
    Replace h_{t,m^-}^k
  end for

  // Update $\mathcal{Q}$ function
  // Update importance weights, then re-normalize
  $w_i^k = \frac{y_i - p(\mathbf{x}_i)}{P(\mathbf{x}_i)}\, \mathcal{Q}^k(\mathbf{x})\, p^k(\mathbf{x}_i)$
end for

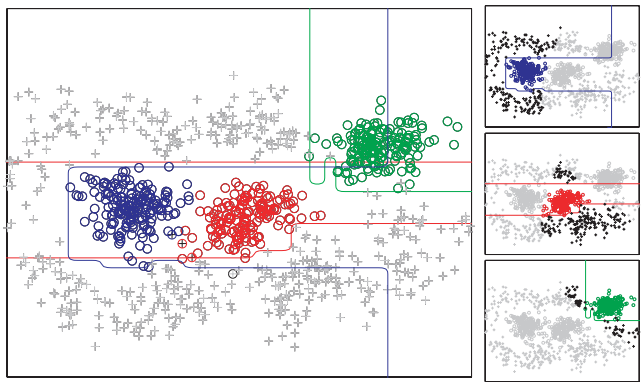
5 Results

In this section we first present results on a toy dataset to demonstrate the behavior of different boosting algorithms and then show results on image sequences.



(a) Standard AdaBoost [5]

(b) Multi-classifier Boosting (MCBoost) [13]



(c) Proposed MCBQ

Figure 2: Learning cluster-specific classifiers on toy data. *The positive class (circles) exhibits three distinct clusters and is surrounded by data from the negative class (crosses). (a) The classification result using standard two-class boosting shows errors due to the XOR configuration of one of the positive data clusters (blue circles denote classification as positive class). (b) The Multi-classifier boosting algorithm of [13] is not able to capture the clusters that are not well separated by negative data, leading to duplicated use of weak classifiers. (c) The classification result of the proposed MCBQ algorithm shows improved clustering leading to more efficient use of weak classifiers.*

5.1 Comparison on Toy Dataset

We first show an illustrative result on a toy dataset, see Figure 2, where we compare standard AdaBoost [5], the multi-classifier boosting algorithm MCBoost [13] and the proposed MCBQ algorithm. The positive class exhibits three clusters. Standard boosting shows poor separation of the classes because it is unable to resolve XOR configurations of the positive class. The MCBoost algorithm shows overlapping areas of expertise for the clusters that are not separated well by negative data points, leading to inefficient use of weak classifiers. The MCBQ algorithm shows improved partitioning of the input space using the Q function, which in this case is simply defined by a Gaussian mixture in $\mathbf{R}^2$.

Figure 3: Tracking over different object appearances. Top row: *frames from the initial calibration sequence (about 200 frames) for real-time training of a target object which undergoes appearance changes.* **Below:** *The classifier learned with MCBQ (with K=2 strong classifiers) allows successful tracking during rapid appearance changes. Note that on-line trackers without prior model are generally unable to handle these cases.*

5.2 Rapid appearance changes

Our initial calibration stage followed by tracking provides a good compromise between the rigidity of off-line training and the over-adaptivity of purely on-line trackers: we show the tracker the exact object to be tracked, and the various appearances it should expect of it. We then track by simply searching for the maximum classifier response in the vicinity of the last result. Figure 3 shows typical results for tracking an object through learned appearance changes. In this case K is set to two because we expect two different appearances.

5.3 Pose clustering

For this experiment we captured a short calibration sequence (100 frames) of a face, while the head is rotating from left to right. We train classifiers using the MCBoost algorithm of [13] and the proposed MCBQ algorithm. In both cases the number of strong classifiers K is set to 3. The Q function is defined by a 3-component Gaussian mixture on the first d=30 principal components. The graph in Figure 4 shows the contribution of each strong classifier on the test sequence, which contains face images in similar poses as the training sequence. The MCBoost algorithm shows no clear pose-specific response, while MCBQ has successfully captured three distinct pose clusters, left, right, and center, as shown by the switches in classifier weights.

In order to measure the separation of classifiers we collected a dataset of 500 frames with a face in different poses. We compute a score that measures the output of the classifier

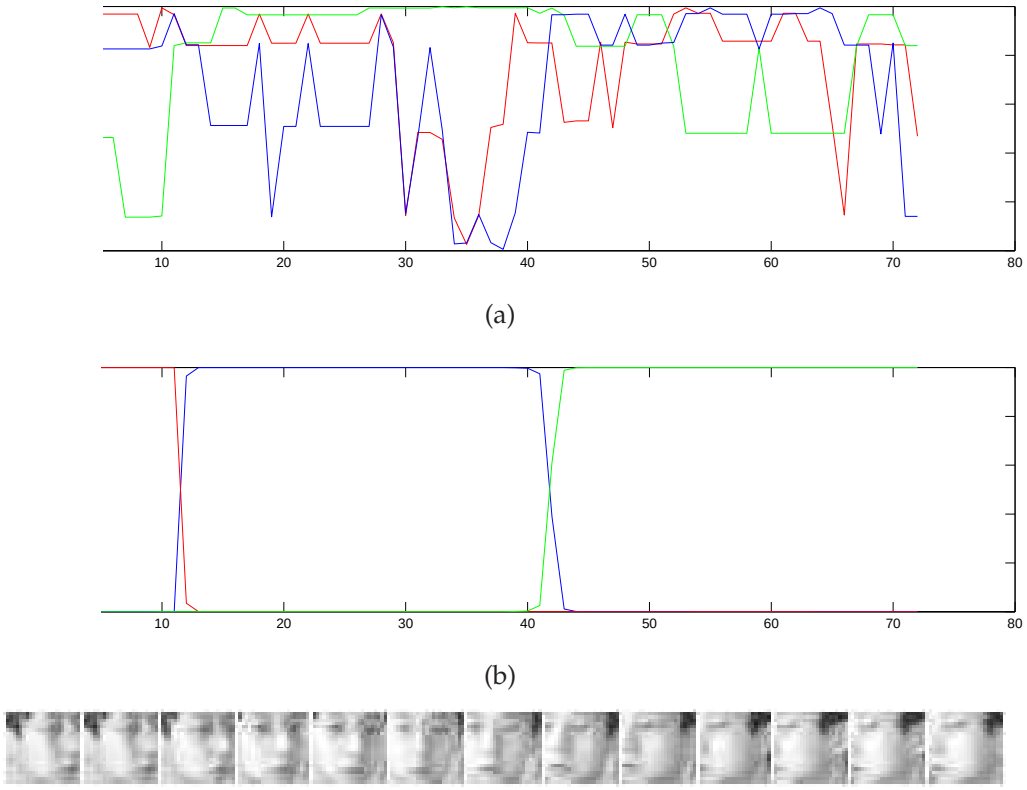


Figure 4: Improved pose expertise: *Plots of the contributions of three strong classifiers given the image input (bottom row).* **(a)** *the discriminative multi-classifier boosting algorithm of [13] shows no clear separation of expertise over different poses.* **(b)** *MCBQ has learned pose-specific classifiers for left, center and right view of the face.*

with the highest response versus the other classifiers at each frame t as

$$\eta(t) = \sum_{p_j(t) \neq p\text{max}(t)} p\text{max}(t) - p_j(t). \tag{11}$$

The following table shows the mean values of $\eta(t)$ on the test data for the MCBoost algorithm in [13] and MCBQ.

Algorithm	Mean specificity score (Equation 11)
MCBoost [13]	0.26
MCBQ	0.97

Figure 5 shows a comparison of the recognition rate of MCBoost [13] and MCBQ for different numbers of weak classifiers. The better space partitioning of MCBQ allows it to achieve significantly higher recognition rates using few weak learners. For this experiment we have used 160 positive and 1300 negative training samples. The positive examples are

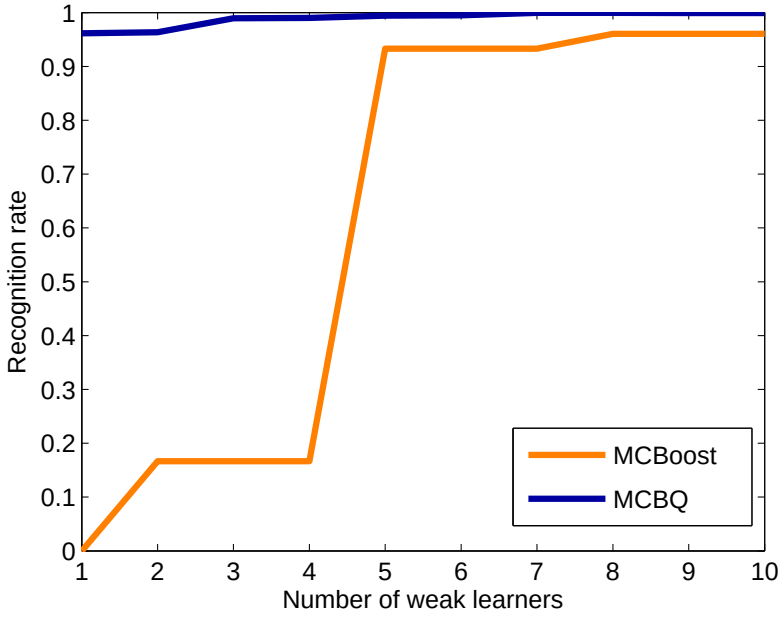


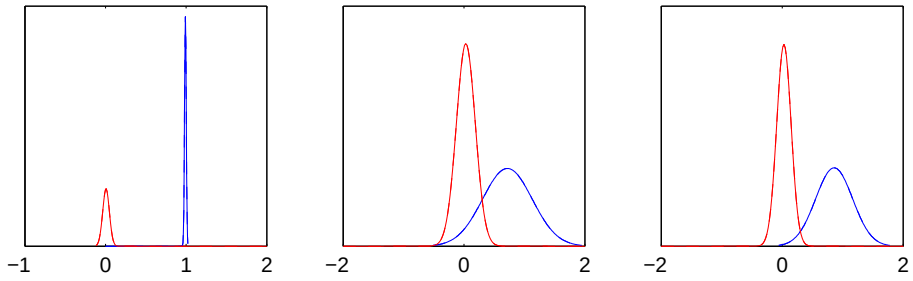
Figure 5: ***Varying number of weak classifiers.*** *This plot shows a comparison of the recognition rate of MCBoost [13] and MCBQ for different numbers of weak classifiers. The better space partitioning of MCBQ allows it to achieve significantly higher recognition rates using few weak learners.*

manually cropped face images and the negative samples are randomly drawn from the background regions. The number of strong classifiers is set as three for both MCBoost and MCBQ. When doing classification, classifier selection was done by $\mathcal{Q}$in MCBQ, i.e. by activating classifiers that have $\mathcal{Q}(x) > 0.1$. As it has very distinctive classifiers, it typically activates just one classifier for each sample thus saving weak-learners.

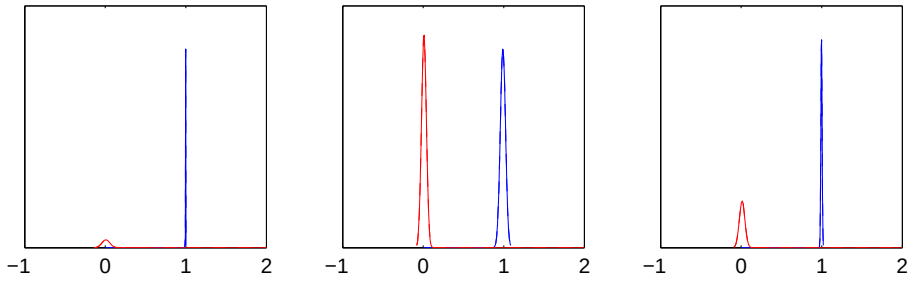
5.4 Improvement by On-line Updating

In order to demonstrate the benefit of on-line MCBQ over off-line MCBQ we have first trained a classifier with 3 strong classifiers on a training sequence with changing head pose (the same as in the previous experiment). We then compared the performance on a test sequence using the learned model with and without adaptation. Positive data is obtained by manual labeling of the target object and negative samples from the background area. The results in Figure 6 shows histograms of classification scores for positive and negative samples for both cases, the fixed learned model and with on-line adaptation. The fixed model shows some overlap in the distributions indicating mis-classification, whereas the classes remain clearly separated in the adaptive algorithm.

Figure 7 shows frames from a sequence showing the pose specific output of a classifier learned with MCBQ. After a calibration sequence of approximately 5 seconds where the face is shown centered and rotated to the left and right, respectively, the tracker is able to



(a) Off-line MCBQ on test set



(b) On-line MCBQ on test set

Figure 6: Improvement by on-line updating. *Shown are histograms of classification scores on unseen test data for positive (blue) and negative (red) samples.* **(a)** *the fixed classifier shows some overlap in the distributions indicating mis-classification, whereas* **(b)** *the classes remain clearly separated in the adaptive algorithm.*

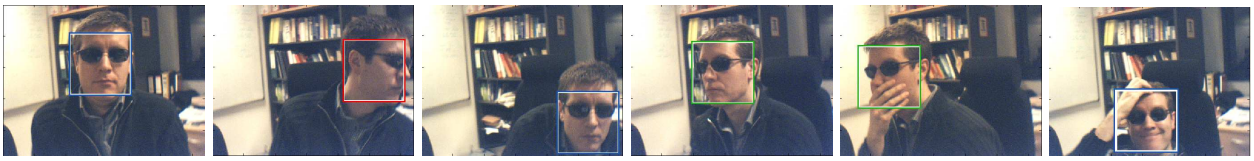


Figure 7: Tracking with changes of pose. *This figure shows results from tracking a face sequence (1000 frames) using the proposed MCBQ classifier. The highest classifier output is shown to be pose specific as indicated by the color of the bounding box (green=left, blue=center, red=right). Because the classifiers as well as their area of expertise in the input space is learned jointly, no manual pose labeling was necessary. The number of strong classifiers is set to 3 in this example.*

track the face as well as indicate the current pose, corresponding to the largest classifier output.

Note that in all experiments the number of strong classifiers has been set manually. However, an additional benefit of introducing the $\mathcal{Q}$ function is that the number of components in the Gaussian mixture can be chosen using model estimation techniques. All experiments have been implemented on a 2.3 GHz Laptop with an Intel Centrino Duo Processor. The training of a three component classifier on 200 frames takes approximately 10 seconds and tracking runs at approximately 8fps using unoptimized Matlab code.

6 Conclusion

We proposed MCBQ, a new multi-classifier boosting algorithm with a soft partitioning of the input space. This is achieved with a function $\mathcal{Q}$ that explicitly assigns areas of expertise in the input space to different strong classifiers. This leads to a more efficient use of weak classifiers compared to standard AdaBoost and a state-of-the-art multi-classifier boosting algorithm that clusters in the discriminative feature space. Motivated by visual tracking applications we propose an on-line scheme to iteratively adapt the classifiers during tracking. In experiments we have shown that the algorithm is able to learn multi-modal appearance models that can be used to detect rapid pose changes.

Bibliography

[1] S. Avidan. Ensemble tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 29(2):261–271, 2007.

[2] B. Babenko, P. Dollár, Z. Tu, and S. Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *Workshop on Faces in Real-Life Images*, Marseille, France, October 2008.

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Proc. Conf. Computer Vision and Pattern Recognition*, Miami, FL, June 2009. to appear.

[4] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, October 2005.

[5] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997.

[6] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.

[7] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *Trans. PAMI*, 22(9):1042–1049, 2000.

[8] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *Proc. ECCV*, Marseille, France, October 2008.

[9] T. Jebara and A. Pentland. Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 144–150, June 1997.

[10] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust on-line appearance models for visual tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 25(10):1296–1311, 2003.

[11] M. Jones and P. Viola. Fast multi-view face detection. Technical Report 96, MERL, 2003.

[12] M. I. Jordan and R. A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[13] T.-K. Kim and R. Cipolla. MCBoost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *Adv. Neural Information Processing Systems*, Vancouver, Canada, December 2008.

[14] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. *cviu*, 99(3):303–331, 2005.

[15] S. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proc. 7th European Conf. on Computer Vision*, volume IV, pages 67–81, Copenhagen, Denmark, May 2002.

[16] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *Proc. Conf. Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007.

[17] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Adv. Neural Information Processing Systems*, pages 512–518, 2000.

[18] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *Proc. 8th European Conf. on Computer Vision*, volume I, pages 28–39, Prague, Czech Republic, May 2004.

[19] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 762–769, Washington, DC, July 2004.

[20] P. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, pages 511–518, December 2001.

[21] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *Adv. Neural Information Processing Systems*, pages 1417–1426, 2006.

[22] O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 27:1292–1304, 2005.