Multiple Classifier Boosting and Tree-Structured Classifiers

Tae-Kyun Kim and Roberto Cipolla

Abstract. Visual recognition problems often involve classification of myriads of pixels, across scales, to locate objects of interest in an image or to segment images according to object classes. The requirement for high speed and accuracy makes the problems very challenging and has motivated studies on efficient classification algorithms. A novel multi-classifier boosting algorithm is proposed to tackle the multimodal problems by simultaneously clustering samples and boosting classifiers in Section 2. The method is extended into an online version for object tracking in Section 3. Section 4 presents a tree-structured classifier, called Super tree, to further speed up the classification time of a standard boosting classifier. The proposed methods are demonstrated for object detection, tracking and segmentation tasks.

1 Introduction

Boosting has become a standard method in object detection [3], tracking [26] and segmentation [33] problems, where a vast number of image sub-windows, across pixels and scales, need to be classified. Performing the tasks in a reasonable time demands extremely fast evaluation of each sub-window. A boosting classifier makes a decision by aggregating simple weak-learners such as Haar-like features, whose computations are accelerated by an integral image.

When object images exhibit multi-modalities (see Figure 1), a single boosting classifier is often not sufficient. A standard boosting classifier [25, 9] is represented by the weighted sum of binary weak-learners as

Tae-Kyun Kim
Department of Electrical and Electronic Engineering, Imperial College, London, UK
e-mail: tk.kim@imperial.ac.uk

Roberto Cipolla
Department of Engineering, University of Cambridge, UK
e-mail: cipolla@eng.cam.ac.uk

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 163–196.
springerlink.com

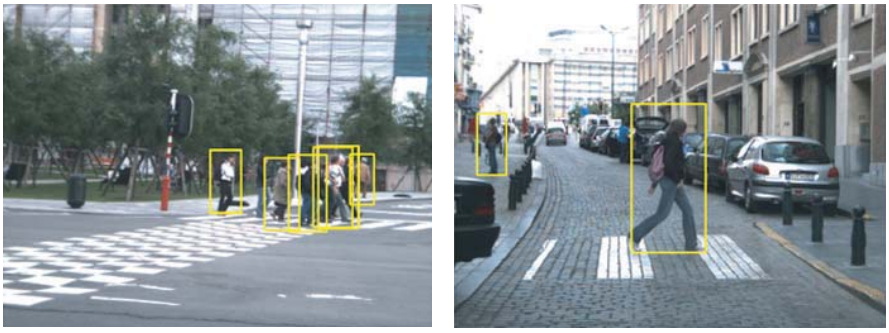


Fig. 1 Pedestrian detection. Wojek *et al.* [19] have shown that the proposed multiple classifier boosting algorithm [31] outperforms various standard methods (SVM, AdaBoost etc) for the multi-appearance pedestrian detection problems.

$$\mathrm{H}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathrm{h}_i(\mathbf{x}), \tag{1}$$

where α_i is the weight and h_i the i-th binary weak-learner in $\{-1, 1\}$. Object images of e.g., multi-poses or multi-object categories are difficult to be dichotomised from non-object images by a single aggregation of simple features. Conventionally, multiple boosting classifiers, each of which is for a defined sub-category, are required [4, 6]. However, manual labeling of object categories and/or poses is difficult for a large data set and how to partition images into sub-categories is often not clear. We present a new co-clustering problem of images and visual features in Section 2. The problem is tackled by simultaneously boosting multiple classifiers which compete for object images by their expertise. Each boosting classifier is an aggregation of weak-learners, i.e., simple visual features. The solution is achieved by a gradient-descent optimisation technique. We demonstrate by both synthetic and real image data sets that the obtained classifiers are capable of solving XOR i.e., multi-modal classification problems that a standard boosting classifier fails to solve.

In object tracking a major challenge is handling appearance changes of a target object due to factors such as changing pose, illumination and deformation. Recently a class of techniques using discriminative tracking has been shown to yield good results by treating tracking as a classification framework [21, 22, 24, 26] (see Figure 2). A classifier is iteratively updated using positive and negative training samples extracted from each frame. Online boosted classifiers have been widely adopted owing to their efficiency and good classification performance [22, 26, 27]. However, as they maintain a single boosted classifier, they are limited to single view tracking or slow view changes of a target object. Tracking tends to fail during rapid appearance changes, because most weak learners of a boosted classifier do not capture the new feature distributions. Rapid adaptation of an online classifier in order to track these changes increases the risk of incorrectly adapting to background regions. Section 3 presents a new multi-pose object tracking solution by extending the multi-classifier

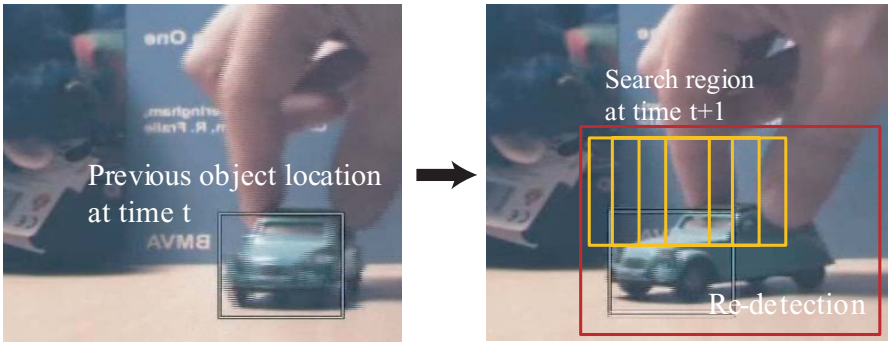


Fig. 2 Object tracking by fast re-detection. A search window is set based on the previous location and speed of a target object, and a classifier is applied to evaluate every sub-window within the search region. The proposed online multiple classifier boosting method [34] allows tracking during rapid pose changes.

boosting algorithm in Section 2 into an online version. The proposed algorithm *jointly* learns the classifiers and a soft partitioning of the input space, defining an area of expertise for each classifier. We show how this formulation improves the specificity of the strong classifiers, allowing simultaneous location and pose estimation in a tracking task. The proposed online scheme iteratively adapts the classifiers during tracking.

Despite the efficiency of a boosting classifier, it is often required to further reduce the evaluation time. A cascade of boosting classifiers, which could be seen as a degenerate tree, effectively improves the classification speed: by filtering out majority of negative class samples in its early stages [3]. Designing a cascade, however, involves manual efforts for setting a number of parameters: the number of classifier stages, the number of weak-learners and the threshold per stage. In Section 4, we present a novel way to speed up the evaluation time of a boosting classifier without needing a conventional multi-stage boosting cascade. We make a shallow (flat) network deep (hierarchical) by growing a tree from decision regions of a given boosting classifier. The obtained tree, called Super tree, provides many short paths for speeding up while it preserves the reasonably smooth decision regions of the boosting classifier for good generalisation. For converting a boosting classifier into a decision tree, we formulate a Boolean optimisation problem, which has been previously studied for circuit design but limited to a small number of binary variables. The method has been demonstrated for segmentation problems (see Figure 3).

The rest of the chapter is organised as follows: Section 2 presents the multiple classifier boosting algorithm to learn multi-modal appearances, Section 3 its online version for object tracking. Conversion of a boosting classifier into a decision tree for speeding up is explained in Section 4. The summary and conclusion is drawn in Section 5.

Fig. 3 Semantic segmentation. Every pixel of an input image (left) is assigned one of object categories (right).

2 Multiple Classifier Boosting

It is known that visual cells (*visual features*) selectively respond to *imagery patterns* in perception. Learning process may be associated with co-clusters of visual features and imagery data in a way of facilitating image data perception. We formulate this in the context of boosting classifiers with simple visual features for binary classification tasks e.g., object detection [3]. There are two sets of images: a set of object images and a set of non-object images, labeled as positive and negative class members respectively. There are also a huge number of simple image features, only a small fraction of which are selected to discriminate the positive class from the negative class by $\mathrm{H}(\mathbf{x}) = \sum_t \alpha_t \mathrm{h}_t(\mathbf{x})$ where $\mathbf{x}$ is an input vector, α_t, h_t are the weight and the score of t-th weak-learner using a single feature, and H is a boosting classifier. When object images exhibit multi-modalities, a single aggregation of simple features is often not sufficient to dichotomise object images from non-object images. Our problem is to find out subsets of object images, each of which is classified by an associated set of features i.e., a boosting classifier, for maximising classification performance. Note that image clusters to be obtained are coupled with selected features and likewise features to be selected are dependent on image clusters, requiring concurrent clustering of images and features.

See Figure 4 for an example where subsets of face images are pose-wise obtained with associated features by the proposed method (Section 2.1). Features are placed around eyes, a nose and mouth as the cues for discriminating faces from background. As such facial features are distributed differently mainly according to face pose in the example, the obtained pose-wise face clusters are, therefore, intuitive and desirable in perception. Note the challenges in achieving this: the input set of face images are mixed up by different persons, lighting conditions as well as poses. Some are photographs of real-faces and the others are drawings. Desired image clusters are *not observable* in input space. See Figure 5 for the clusters obtained by the traditional unsupervised clustering method (k-means clustering) on the face

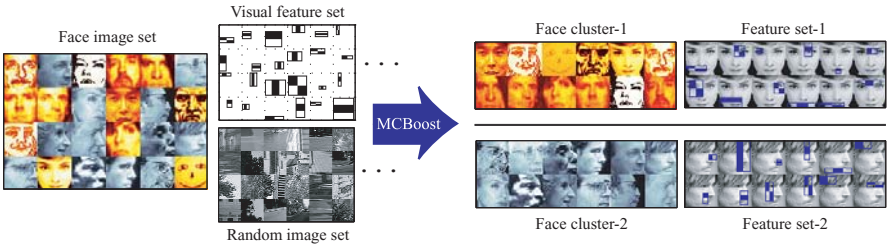


Fig. 4 Perceptual co-clusters of images and visual features. For given a set of face and random images and simple visual features, the proposed method finds the joint-clusters of face images and features, which facilitates classification of face images from random images. Face clusters are pose-wise obtained in the example.

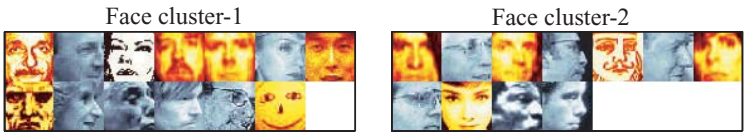


Fig. 5 Face clusters obtained by the k-means clustering method.

images. Images of the obtained clusters are almost random with respect to pose. A required method must have a discriminative process and part-based representations (like the simple features used) to obtain more meaningful face clusters for classification. Technically, it is also required to cope with an arbitrary initialisation of image clusters because the target clusters are hidden. The k-means clustering result is completely different from that obtained by the proposed method as shown in Figure 4 and Figure 5. Feature selection should be efficiently performed among a huge number of input features.

We simultaneously boosts multiple boosting classifiers, each of which has expertise on a particular set of object images by a set of weak-learners. The proposed method (Section 2.1) has a potential for wide-applications in perceptual data exploration. It generally solves a new co-clustering problem of a data set (e.g., a set of face images) and a feature set (e.g., simple visual features) in a way to maximise discrimination of the data set from another data set (e.g., a set of random images).

Related Work

Existing co-clustering work (e.g., [1]) is formulated as an unsupervised learning task. It simultaneously clusters rows and columns of a co-occurrence table by e.g., maximising mutual information between the cluster variables. Conversely, we make use of class labels for discriminative learning. Using a co-occurrence table in prior work is also prohibitive due to a huge number of visual features that we consider.

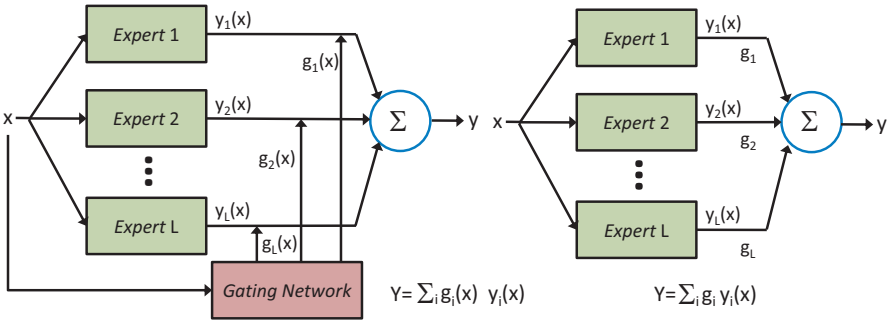


Fig. 6 Mixture of Experts vs Ensemble Learning. In MOE, the gating network as a function of input activates an expert encouraging specialization. In Ensemble learning i.e., Boosting, all experts contribute to form a decision with pre-determined weights.

Mixture of Experts [2] (MoE) jointly learns multiple classifiers and data partitions. It emphasises local experts and is suitable when input data are naturally divided into homogeneous subsets, which is, however, often not the case in practice as observed in Figure 5. Note that EM in MoE resorts to a local optimum and in practice, it is difficult to establish a good initial data partition. Furthermore, the data partitions of MoE could be undesirably affected by a large background class considered in our problem and the linear transformations used in MoE are limited for delivering a meaningful part-based representation of object images.

Boosting [8] is a sequential learning method for aggregating multiple weak classifiers. It finds weak-learners to correctly classify erroneous samples by previous weak-learners. While MoE makes a decision by dynamically selected local experts, all experts or weak-learners in a boosting classifier contribute to a decision with weights (See Figure 6). Expert selection required in MoE is generally a difficult problem when an input space is not naturally divided into sub-regions (clusters). A boosting classifier solves various non-linear classification problems but cannot solve XOR problems where only half the data can be correctly classified by each weak-learner (see [8] for the strength of weak learnability). Two disjointed sets of weak-learners, i.e., two boosting classifiers, are required to conquer each half of data by a set of weak-learners. Torralba *et al.*'s method for multiple boosting classifiers [4] relies on manual labels for cateogry/pose, whereas we optimise image clusters and boosting classifiers simultaneously.

2.1 MCBoost: Multiple Classifier Boosting

Our formulation considers K strong classifiers, each of which is represented by a linear combination of weak-learners as

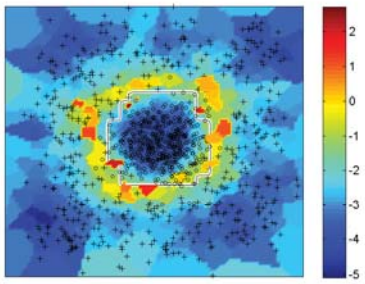


Fig. 7 Risk map computed for given two class data (circle and cross). Weak-learners (either a vertical or horizontal line) found by the Adaboost method [8] are placed on the high risk regions.

$$\mathrm{H}_k(\mathbf{x}) = \sum_t \alpha_{kt} \mathrm{h}_{kt}(\mathbf{x}), \qquad k = 1, ...K, \tag{2}$$

where α_{kt} and h_{kt} are the weight and the score of t-th weak-learner of k-th strong classifier. Each strong classifier is devoted to a subset of input patterns allowing repetition and each weak-learner in a classifier comprises of a single visual feature and a threshold. For aggregating multiple strong classifiers, we formulate Noisy-OR as

$$P(\mathbf{x}) = 1 - \prod_k (1 - P_k(\mathbf{x})), \tag{3}$$

where $P_k(\mathbf{x}) = 1/(1 + \exp(-\mathrm{H}_k(\mathbf{x})))$. It assigns samples to a positive class if any of classifiers does and to a negative class if every classifier does. That is, an individual classifier is forced to learn from a subset of positive samples and all negative samples. A positive sample is therefore required to be accepted as the positive class by at least one of the classifiers and a negative sample to be rejected by all. The Noisy-OR framework does not require classifier selection when making a decision: the joint probability in (3) is computed using all k classifiers for any $\mathbf{x}$. Note that the mixture of experts partitions an input space into many overlapping or disjoint regions and only classifiers of regions that a test data point falls in are used. This is a significant difference in design. A conventional design in object detection study [6] also favours the OR framework that does not need classifier selection. Our derivation builds on the previous Noisy-OR boosting algorithm [5], which has been proposed for multiple instance learning. It learns a single boosting classifier from given bags of samples whereas ours learns multiple boosting classifiers.

The sample weights are initialised e.g., by randomly partitioning positive samples, i.e., $w_{ki} = 1$ if $\mathbf{x}_i \in k$ and $w_{ki} = 0$ otherwise, where i and k denote i-th sample and k-th classifier respectively. We set $w_{ki} = 1/K$ for all k's for negative samples. For given weights, the method finds K weak-learners at t-th round of boosting, to maximise

$$\sum_i w_{ki} \cdot \mathrm{h}_{kt}(\mathbf{x}_i), \qquad \mathrm{h}_{kt} \in \mathcal{H}, \tag{4}$$

Algorithm 1. MCBoost

Input: A data set $(\mathbf{x}_i, y_i)$ and a set of pre-defined weak-learners
Output: Multiple boosting classifiers $\mathrm{H}_k(\mathbf{x}) = \sum_{t=1}^{T} \alpha_{kt} \mathrm{h}_{kt}(\mathbf{x}), k = 1..., K$

1.Compute a reduced set of weak-learners $\mathcal{H}$ by the risk map (5) and randomly initialise the weights w_{ki}.
2.Repeat for $t = 1, ..., T$:
3. Repeat for $k = 1, ..., K$:
4. Find weak-learners h_{kt} that maximise $\sum_i w_{ki} \cdot \mathrm{h}_{kt}(\mathbf{x}_i), \mathrm{h}_{kt} \in \mathcal{H}$.
5. Find the weak-learner weights α_{kt} that maximise $\mathrm{J}(\mathrm{H} + \alpha_{kt}\mathrm{h}_{kt})$.
6. Update the weights by $w_{ki} = \frac{y_i - P(\mathbf{x}_i)}{P(\mathbf{x}_i)} \cdot P_k(\mathbf{x}_i)$.
7. End
8.End

Fig. 8 Pseudocode of MCBoost algorithm

where $\mathrm{h}_{kt} \in \{-1, +1\}$ and $\mathcal{H}$ denotes a reduced set of weak-learners for speeding up the learning process. Previous methods e.g., [20] for reducing the boosting learning time may be independently deployed. The reduced set is obtained by restricting the location of weak-learners around the expected decision boundary. Each weak-learner, $h(\mathbf{x}) = \mathrm{sign}(\mathbf{a}^T\mathbf{x} + b)$, where $\mathbf{a}$ and b represent a simple feature and its threshold respectively, can be represented by $\mathbf{a}^T(\mathbf{x} - \mathbf{x}_o)$, where $\mathbf{x}_o$ is interpreted as the location of the weak-learner. By limiting $\mathbf{x}_o$ to the data points that have high risk to be misclassified, the complexity of searching weak-learners at each round of boosting is reduced. The risk is defined as

$$R(\mathbf{x}_i) = \exp\{-\frac{\sum_{j \in \mathcal{N}_i^B} \|\mathbf{x}_i - \mathbf{x}_j\|^2}{1 + \sum_{j \in \mathcal{N}_i^W} \|\mathbf{x}_i - \mathbf{x}_j\|^2}\} \tag{5}$$

where $\mathcal{N}_i^B$ and $\mathcal{N}_i^W$ are the set of predefined number of nearest neighbors of $\mathbf{x}_i$ in the opposite class and the same class of $\mathbf{x}_i$ (see Figure **??**). The weak-learner weights $\alpha_{kt}, k = 1, ..., K$ are then found to maximise $\mathrm{J}(\mathrm{H} + \alpha_{kt}\mathrm{h}_{kt})$ by a line search. Following the AnyBoost method [9, 5], we set the sample weights for the next round as the derivative of the cost function with respect to the classifier score. For the cost function $\mathrm{J} = \log \prod_i P(\mathbf{x}_i)^{y_i}(1 - P(\mathbf{x}_i))^{(1-y_i)}$, where $y_i \in \{0, 1\}$ is the label of i-th sample, the weight of k-th classifier over i-th sample is updated by

$$w_{ki} = \frac{\partial \mathrm{J}}{\partial \mathrm{H}_k(\mathbf{x}_i)} = \frac{y_i - P(\mathbf{x}_i)}{P(\mathbf{x}_i)} \cdot P_k(\mathbf{x}_i). \tag{6}$$

See Figure 8 for the pseudocode of the proposed method.

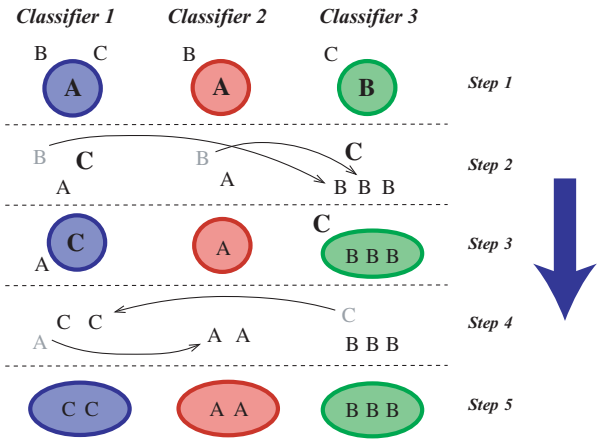


Fig. 9 State diagram for MCBoost.

Data Clustering

Data clusters (of positive samples) are obtained by assigning samples $\mathbf{x}_i$ to a classifier (or cluster) that has the highest classifier probability $P_k(\mathbf{x}_i)$.

The sample weight of k-th classifier in (6) is determined by the joint probability $P(\mathbf{x})$ and the probability of k-th classifier $P_k(\mathbf{x})$. For a negative class ($y_i = 0$), the weights only depend on the probability of k-th classifier. The classifier gives high weights to the negative samples that are misclassified by itself, independently of other classifiers. For a positive class, high weights are assigned to the samples that are misclassified jointly (i.e., the left term in (6)) but may be correctly classified by the k-th classifier through next rounds (i.e., high $P_k(\mathbf{x})$). That is, classifiers concentrate on samples in their expertise through the rounds of boosting. This can be interpreted as data partitioning.

Toy Examples

Figure 9 illustrates the concept of the MCBoost algorithm. The method iterates two main steps: learning weak-learners and updating sample weights. States in the figure represent the mode of samples (A,B or C) that are correctly classified at each step. The sample weighting (6) is represented by data re-allocation. Assume that a positive class has the samples of three target clusters denoted by A, B and C. Samples of more than two target clusters are initially assigned to a classifier. Weak-learners are found to classify the dominant mode of samples (bold letters) in each classifier (step 1). Classifiers then re-assign samples according to their expertise (step 2): Samples C that are misclassified by all are given more importance in the first classifier (bold letter). Samples B are moved to the third classifier as the expert on B. The first classifier learns next weak-learners for classifying the samples C while the second

and third classifiers focus on the samples A and B respectively (step 3). Similarly, the samples A, C are moved into the respective most experts (step 4) and all re-allocated samples are finally correctly classified by weak-learners (step 5).

We present a toy example of XOR classification problems (see Figure 10). The positive class (circle) comprising the three sub-clusters and the negative class (cross) in background make the XOR configurations. Any standard single or double boosting classifiers, therefore, do not successfully dichotomise the classes in the example. We exploit vertical or horizontal lines as weak-learners and set the number of classifiers K to be three by a priori. We performed partitioning of positive samples as shown in the left by three different color blobs (randomly mixed) for initialising the sample weights: each classifier was initially assigned all three color data points having its data center around the center of the coordinate. The final decision boundaries and the tracks of data cluster centres of the three boosting classifiers are shown in the right. Despite the mixed-up initialisation, the method learns the three classifiers that nicely settle into the target clusters after a bit of jittering in the first few rounds. The weak-learner weights (bottom) show the convergence of the three classifiers. Note that the method obtains the data clusters purely by the boosting classifier scores i.e., in a discriminative sense. Although the same data clusters are obtained by conventional clustering methods e.g., k-means in this example, clusters by conventional ways i.e., in a generative sense are often different from those of our method as exemplified in Figure 5. The proposed method works well with random initialisations and desirably exhibits quicker convergence when a better initialisation is given.

2.2 Experiments

Discriminative Clustering

We performed experiments using a set of INRIA pedestrian data [11] and PIE face data [10]. The INRIA set contains 618 pedestrian images as a positive class and 2436 random images as a negative class in training and 589 pedestrian and 9030 random images in testing. The pedestrian images show wide-variations in background, human pose and shapes, clothes and illuminations (Figure 11). The PIE data set involves 900 face images as a positive class (20 persons, 9 poses and 5 lighting conditions) and 2436 random images as a negative class in training and 900 face and 12180 random images in testing. The 9 poses are distributed form left profile to right profile of face, and the 5 lighting conditions make sharp changes on face appearance as shown in Figure 11. Some facial parts are not visible depending on both pose and illumination. All images were cropped and resized into 24×24 pixel images. A total number of 21780 simple rectangle features (as shown in Figure 4) were exploited.

MCBoost learning was performed with the initial weights that were obtained by the k-means clustering method. Avoiding the case that any of the k-means clusters is too small (or zero) in size has helped quick convergence in the proposed method. We set the portion of high risk data as 20% of total samples for speeding up in

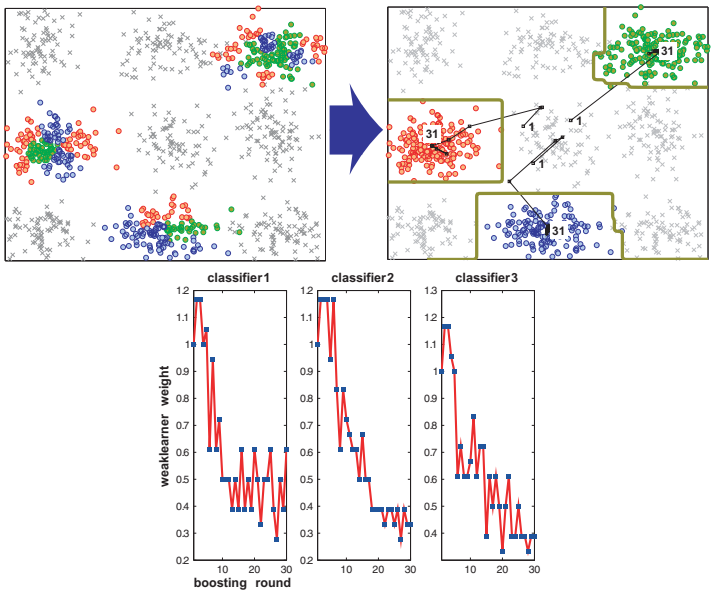


Fig. 10 Example of learning on the XOR classification problem. For a given random initialisation (three different color blobs in the left), the method learns the three classifiers that nicely settle into the desired clusters and correct decision boundaries (right). The weak-learner weights (bottom) show the convergence.

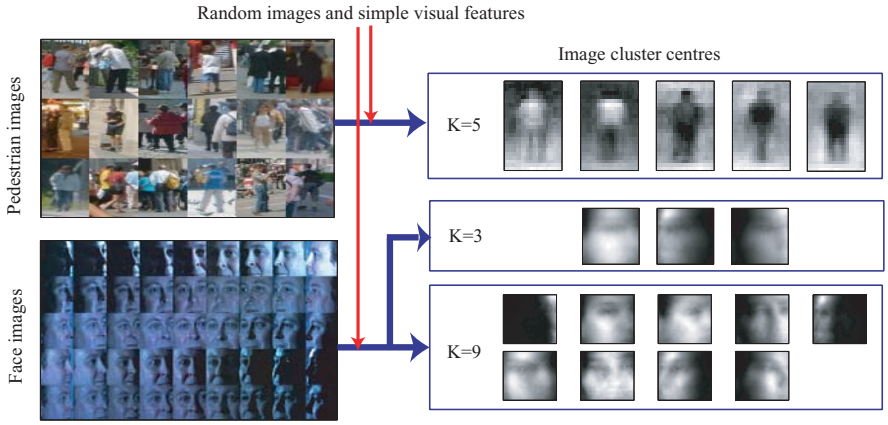


Fig. 11 Perceptual clusters of pedestrian and face images. Clusters are found to maximise discrimination power of pedestrian and face images from random images by simple visual features.

training. The number of classifiers was set as $K \in \{2,3,4,5\}$ and $K \in \{3,5,7,9\}$ for the INRIA and PIE data set respectively. For all cases, every classifier converged within 50 boosting rounds.

Figure 11 shows the cluster centers obtained by the proposed method. The object images were partitioned into K clusters (or classifiers) by assigning them to the classifier that has the highest $P_k(\mathbf{x})$. For the given pedestrian images, the first three cluster centers look unique and the last two are rather redundant. The three pedestrian clusters obtained are intuitive. They emphasise the direction of intensity changes at contours of the human body as discriminating cues of pedestrian images from random images. It is interesting to see distinction of upper and lower body in the second cluster, which may be due to different clothes. For the PIE data set, the obtained face clusters reflect both pose and illumination changes, which is somewhat different from our initial expectation of getting purely pose-wise clusters as in the case of Figure 4. This result is, however, also reasonable when considering the strong illumination conditions that shadow face parts. For example, frontal faces whose right-half side is not visible by the lighting cannot share any features with those having left-half side not visible. Certain profile faces rather share more facial features (e.g., one eye, eye brow and a half mouth) with the half-shadowed frontal faces, jointly making a cluster. All 9 face clusters seem to capture unique characteristics of the face images.

Multi-view Face Detection

Another experiment was designed using the CMU frontal and profile face image data sets [12, 13]. Some face examples were shown in Figure 4. The two data sets contain 322 images in total. The images were randomly and equally partitioned into a train and a test set. The train set of 161 images had 323 frontal faces and 192 profile faces and the test set had 271 frontal and 171 profile faces. Every face was cropped and resized into 24x24 pixel images and around 200 negative samples per image were randomly collected and resized into 24x24 pixel images. The number of negative samples in the initial train set was 32200. Two of standard AdaBoost classifiers (setting the number of weak learners be 50 per each) were initially trained using either the frontal or profile faces (by the manual pose label) with the random negative samples in the initial train set. A total number of 72000 simple rectangle features were exploited. We applied the two learnt classifiers on the train and test images for bootstrapping. The total number of bootstrapped negative samples was 7400 for the train set and 7635 for the test set. The train and test set used for comparison consisted of both frontal and profile face images and bootstrapped negative samples. The standard AdaBoost classifier (using 100 weak-learners), two AdaBoost classifiers either by the k-means clustering (K=2) or the manual pose labels (using 50 weak-learners per each) and the MCBoost classifier initialised by the k-means clustering (with K=2) (50 weak-learners per each) were compared. Figure 12 shows the ROC curves of the methods for the train (left) and test (right) sets respectively. Both graphs showed the same tendency. The MCBoost significantly outperformed the AdaBoost using 100 weak-learners (we varied the number of weak-learners and obtained the best performance by 100 weak-learners) and the AdaBoost with the k-means (K=2). The proposed method delivered the similar accuracy to that of the AdaBoost with the pose labels. The AdaBoost with the k-means outperformed the standard single

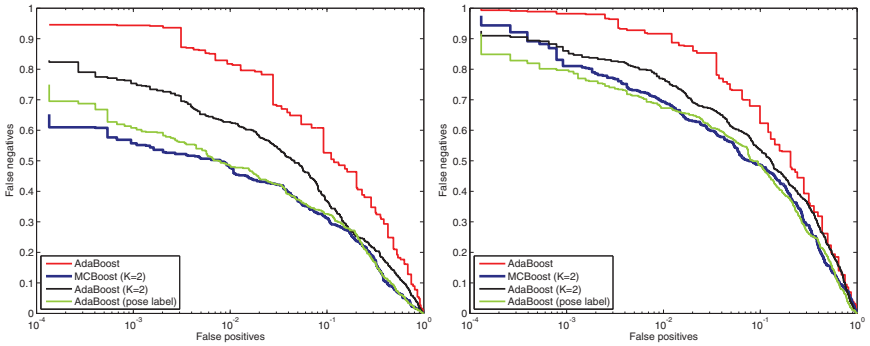


Fig. 12 ROC curves on the CMU frontal and profile face data sets. For the train set (left) and test set (right).

AdaBoost classifier. The results confirmed that the standard boosting classifier can not successfully classify samples in the XOR case (by the multi-modal face samples and bootstrapped negative samples) and the clusters learnt in the proposed discriminative manner are more suited to learn boosting classifiers than those obtained by standard unsupervised clustering methods. MCBoost exhibited very close accuracy to the classifiers learnt by the manual pose labels in the experiment.

The MCBoost method has been extensively tested by Wojek *et al.* for pedestrian detection problems in [19]. They have tested the various combinations of features (HOG, Haar, Oriented Histogram Flow) and classifiers (SVM, AdaBoost, MCBoost) on their new challenging pedestrian data sets. It has shown that MCBoost achieves superior performance to linear SVM-based detectors and significantly outperforms Adaboost for both static and dynamic pedestrian detection problems. MCBoost has been shown to be the most robust classifier with respect to challenging lighting conditions while being computationally less expensive than SVMs.

Discussions

We have introduced a discriminative co-clustering problem of images and visual features and have proposed a novel method of multiple classifier boosting called MCBoost. It simultaneously learns image clusters and boosting classifiers, each of which has expertise on an image cluster. The method works well with either random initialisation or initialisation by conventional unsupervised clustering methods. We have shown in the experiments that the proposed method yields meaningful co-clusters of images and features and significantly outperforms the conventional designs that individually learn multiple boosting classifiers by the clusters obtained by the k-means clustering method or pose-labels.

Useful future studies on the MCBoost method include development of a method to automatically determine K, the number of classifiers. At the moment, we first try a large K and decide the right number as the number of visually heterogeneous clusters obtained (see Section 2.2). A post-corrective step of initial weak-learners

would be useful for more efficient classification by less number of weak-learners in total. When the classifiers start from wrong initial clusters and oscillate between clusters until settling down, some initial weak-learners are wrong and others may be wasted to make up for the wrong ones. Once the classifiers find right clusters, they exhibit convergence by decreasing the weak-learner weights. Restaring MCBoost with the clusters found may yield economical sets of weak-learners for the same accuracy.

3 Online Multiple Classifier Boosting for Object Tracking

Object tracking has been often treated as a classification problem where it is done by fast re-detection. A search window is set based on the previous location and speed of a target object, and the object is detected within the search window. The detector is usually a binary classifier which evaluates sub-windows to tell if they contain a target object or not, at every pixel across scales, within the search window. It requires very efficient evaluation per sub-window, as it typically involves a huge number of sub-windows. Such a classification framework has been shown to yield good tracking results [21, 22, 24, 26]. A classifier is on-line updated to reflect environmental changes. Online boosted classifiers have been widely used owing to their efficiency and accuracy [22, 26, 27]. The work in [24] introduced online feature selection for tracking, where in each frame the most discriminative features are chosen to compute likelihoods. *Ensemble Tracking* [21] takes a similar approach by combining a small number of weak classifiers using AdaBoost. Online boosting for tracking [26] introduced a scheme where features are selected from a pool of weak classifiers and combined into a strong classifier. Online schemes without any target model tend to suffer from drift. One solution is to introduce an object model that is learned prior to the tracking phase [27, 29]. The work in [27] proposed semi-supervised learning, and included a boosted detector or simply the object region in the first frame as a prior to an online boosting scheme.

Maintaining a single boosted classifier during tracking is limited to single view tracking or slow view changes of a target object. Tracking tends to fail during rapid appearance changes, because most weak learners of a boosted classifier are not relevant to the new object appearance. Forcing an online classifier to adapt to these rapid changes increases the risk of incorrectly adapting to background regions. A multi-modal object representation and classifier is therefore required. Such a model can be either generative [23, 32] or discriminative [30]. Typically, in the latter case, distinct appearance clusters are found first and a classifier is trained on each [4].

Recently multi-classifier boosting was introduced, where clustering and classifier training is performed jointly [18, 31] (see Section 2.1). These methods have so far been applied to object detection, where the full training set is available from the beginning. However, direct application to the online tracking domain is not straightforward, the main reason being that in an online setting the number of positive and negative samples is not sufficient to ensure a good partitioning of the input space

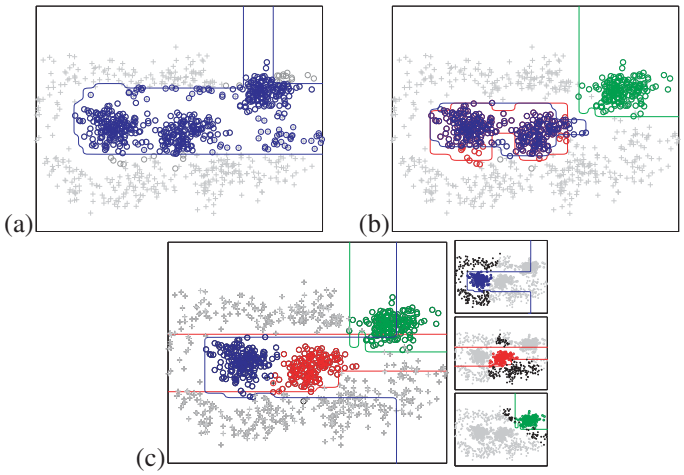


Fig. 13 Learning cluster-specific classifiers on toy data. The positive class (circles) exhibits three clusters and is surrounded by data from the negative class (crosses). (a) The classification result using a standard boosting classifier shows errors due to the XOR configuration (colored circles denote classification as positive class). (b) The Multi-classifier boosting algorithm of [31] successfully divides the two classes but uses two boosting classifiers (blue and red line) in the same region, leading to inefficient use of weak classifiers. The two clusters with no negative data points between them can be correctly classified by a single boosting classifier. (c) The classification result of the proposed MCBQ algorithm shows improved classifier expertise.

in terms of classifier expertise in the initial phase. Figure 13 illustrates the classification results of (a) standard Adaboost [25], (b) MCBoost [31] and (c) the proposed algorithm called MCBQ on a toy XOR classification problem. The positive class exhibits three clusters, but two of them actually form a single cluster in a discriminative sense as there are no negative points between them. Standard AdaBoost shows poor separation of the classes because it is unable to resolve XOR configurations. For the MCBoost algorithm and the proposed solution, we set the number of classifiers to be three. MCBoost successfully divides the two classes but shows overlapping areas of expertise for the two classifiers, since the two clusters without negative data points in-between can be correctly classified by a single boosting classifier. In contrast, the proposed algorithm shows improved partitioning of the input space. As a consequence, weak classifiers are used more efficiently. While tracking continues, additional negative samples are collected, eventually establishing three positive clusters in a discriminative sense in this example. However, in the case of MCBoost, the initially incorrectly assigned boosting classifiers are difficult to be correctly reassigned during online updates. We have observed this case when classifiers are initially trained on a short sequence that contains multi-views of a target object and are subsequently updated.

We therefore propose an extension 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 online version of the algorithm to dynamically update the classifiers and the partitioning for the task of multi-modal object tracking. The algorithm is applied to object tracking where it is used to learn different appearance clusters during a short initial supervised learning phase.

Other related work is online multiple instance learning (MIL) [22, 5]. Our proposed method can be seen as a multi-class extension of [22].

3.1 Joint Boosting and Clustering

This section explains our improvements in the MCBQ algorithm, based on the multi-classifier boosting algorithm in Section 2. 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 labels $y_i \in \{-1,+1\}$ corresponding to non-object and object, respectively. Additionally, each of the object samples can be considered belonging to one of K groups where the class membership is a priori unknown.

Multi-classifier Boosting creates strong classifiers with different areas of expertise. However, it relies on the training data set containing negative samples which separate the positive samples into distinct regions in the classifiers' discriminative feature space. This also implies that there is no guarantee of pose-specific clustering. 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 functions $\mathcal{Q}^k(\mathbf{x}_i) : \mathcal{X} \rightarrow [0,1]$ which weight the influence of strong classifier k on a sample $\mathbf{x}_i$. By mapping $\mathbf{x}_i$ into a suitable metric space, we can impose any 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 method $\mathcal{Q}$ 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 using a EM-like algorithm alongside the weak classifiers in the boosting algorithm (Algorithm 1).

The new noisy-OR function in Equation 3 becomes:

$$p(\mathbf{x}_i) = 1 - \prod_k (1 - \mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)), \tag{7}$$

leading to the new weight update equation:

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \, \frac{\mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)\,(1 - p^k(\mathbf{x}_i))}{1 - \mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)}. \tag{8}$$

The full MCBQ algorithm is summarized in Algorithm 2. Note that compared to the original multi-classifier boosting algorithm additional steps 1, 2, and 8 are required and step 7 is modified.

Algorithm 1. Updating Weighting Function

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

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, $\mathcal{Q}^k(\mathbf{x}_i)p^k(\mathbf{x}_i)$

3. Update the k-th cluster by the mean and covariance matrix of the samples under this probability.

Algorithm 2. Multi-classifier Boosting with Weighting Function (MCBQ)

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

3.2 Online MCBQ for Object Tracking

The goal is to learn an object-specific appearance model using a short initial training sequence in order to guide the tracker [27, 32]. The number of training samples is limited, but is sufficient 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 of iteratively adapting multiple strong classifiers with MCBQ. In order to move MCBQ into an online setting we need a mechanism for rapid feature selection and incremental updates of the weak classifiers as new training samples become available. The online boosting algorithm [26] addresses this issue, allowing for the continuous learning of a strong classifier from training data. 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. Secondly, additional training samples are supplied to update the classifier with new data during tracking.

Weak Learning and Selection

All weak classifiers use a single Haar-like feature. For online learning from a feature f and labeled samples $(\mathbf{x}_i, y_i)$ we create a decision threshold θ_m^k with parity p_m^k 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:

$$h_{t,m}^k(\mathbf{x}_i) = p_m^k \operatorname{sign}(f(\mathbf{x}_i) - \theta_m^k), \tag{9}$$

$$\theta_m^k = (\mu^{k,+} + \mu^{k,-})/2, \quad p_m^k = \operatorname{sign}(\mu^{k,+} - \mu^{k,-}), \tag{10}$$

$$\mu^k = \frac{\Sigma_i |w_i^k| f(\mathbf{x}_i)}{\Sigma_i |w_i^k|}. \tag{11}$$

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

$$e_{t,m}^k = \frac{\sum_i \mathbf{1}(h_{t,m}^k(x_i) \neq y_i)|w_i^k|}{\sum_i |w_i^k|}. \tag{12}$$

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

Supervised Learning

During the supervised learning stage, we have a set of weighted samples, and a global feature pool $\mathcal{F}$. Weight distributions are initialized to randomly assign positive samples to a strong classifier k, and at each round t and strong classifier k the equations 9, 10, 11, 12 are applied 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 only keeps the best M performing weak classifiers for use in the incremental update stage. After each round of boosting, image weights are updated as in Equation 8, and voting weights calculated based on the error of the chosen weak classifier.

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

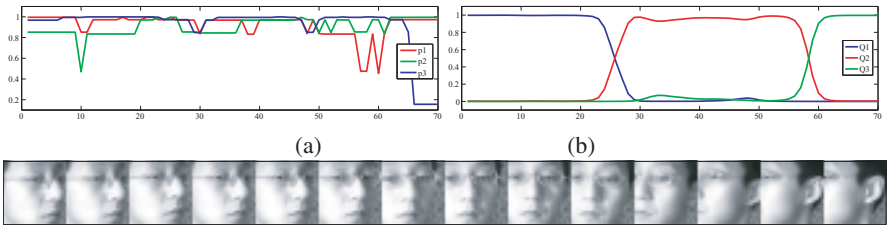


Fig. 14 Improved pose expertise: Plots of the contributions of three strong classifiers given the image input (bottom row). (a) MCBoost [31] shows no clear separation of expertise over different poses, while (b) MCBQ has learned pose-specific classifiers, corresponding to left, center and right view of the face.

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 9, 10, 11. The error values from Equation 12 are used to choose the best 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 as a Gaussian mixture in PCA space, we update the PCA space by the algorithm of Hall *et al.* [28] before updating $\mathcal{Q}$. Pseudo-code is given in Algorithm 3.

3.3 Results

Pose Clustering

For this experiment we captured short training and testing sequences (about 100 frames each) of a face rotating from left to right, see Fig. 14. We trained classifiers using MCBoost [31] and the MCBQ algorithm on face images and random patches sampled from the training sequence. In both cases the number of strong classifiers K is set to 3 by hand. The $\mathcal{Q}$ function is defined by a 3-component Gaussian mixture on the first 30 principal components. The graph in Fig. 14 shows the contribution of each strong classifier on the test 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 changes in classifier weights.

Tracking Performance

In order to evaluate the performance on the multi-appearance tracking problem, we captured four sequences where the target object rapidly changes its pose. The sequences are *toyface* (452 frames), *handball* (210 frames), *cube* (357 frames), and *face* (185 frames). We also compared on the public *Sylvester* sequence (1345 frames). The performance was evaluated against manually labeled ground truth. We compared AdaBoost, MCBoost and MCBQ trackers (both manually set to $K = 2$),

Algorithm 3. Online MCBQ – Incremental Update

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

// 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 $\left(h_{t,m}^k, (\mathbf{x}_i, y_i), w_i^k\right)$
      // Calculate new error $e_{t,m}^k = \sum_i \mathbf{1}(h_{t,m}^k(\mathbf{x}_i) \neq y_i) |w_i^k|$
    end for
    // Choose the weak classifier with the lowest error
    $m^* = \mathrm{argmin}_m \left(e_{t,m}^k\right)$, $h_t^{k*} = h_{t,m^*}^k$ and $e_t^{k*} = e_{t,m*}^k$
    // Calculate voting weight $\alpha_t^k = 1 / \left(1 + \exp\{-\ln\left(\frac{1-e_t^{k*}}{e_t^{k*}}\right)\}\right)$
    // Replace the weak classifier with the highest error
    $m^- = \mathrm{argmax}_m \left(e_{t,m}^k\right)$ and replace h_{t,m^-}^k
  end for

  // Update $\mathcal{Q}^k(\mathbf{x}_i)$ function
  // Update importance weights by Equation 8, then re-normalize.
end for

as well as two publicly available trackers, Semi-supervised Boosting [27] and MIL tracking [22]. For each sequence the initial classifier was trained on a short initial training set (25-40 frames), capturing the appearance variation, and updated online during tracking. Examples of positive training samples are shown in Fig. 15. Because such a training set is generally not available for public tracking sequences, the training data for the *Sylvester* sequence was constructed by randomly sampling 30 frames from the whole sequence. For AdaBoost, MCBoost and MCBQ 50 random patches per frame were collected as negative class samples. We stopped boosting rounds when the classification error reached zero on the training samples. The public code for semi-supervised Boosting and MIL tracking was modified so that these methods can also be trained on the initial set, otherwise their default parameters were used. Parameter settings were unchanged for all experiments. Fig. 16 shows the tracking errors on the five sequences. While none of the algorithms was able to successfully track the target in all sequences, MCBQ showed the best overall performance, in particular outperforming AdaBoost and MCBoost. The MIL tracker performed best on two sequences, however, was not able to recover from drift in two

Fig. 15 Positive class samples for training. A subset of the positive samples is shown for the four sequences.

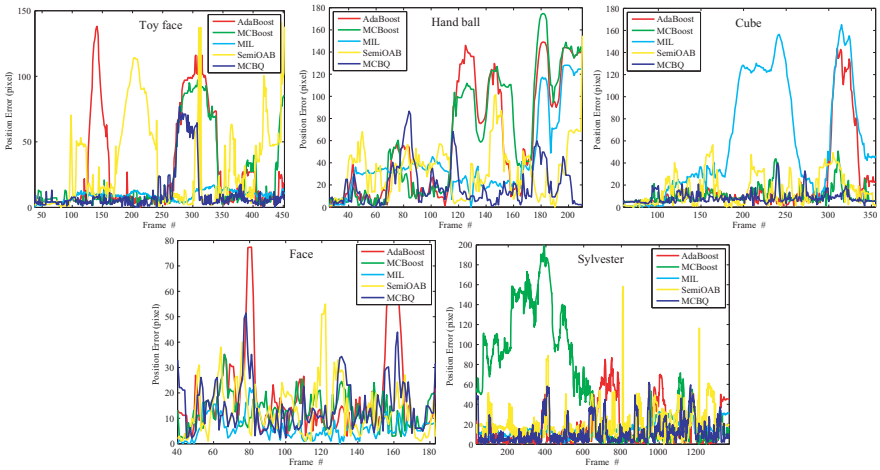


Fig. 16 Tracking error on test sequences. The plots show the tracking error over time on four test sequences for AdaBoost (red) MCBoost (green), MCBQ (blue), MILTrack (cyan), and SemiBoost (yellow). MCBQ shows the best overall performance.

of the other sequences. Overall, the single classifier trackers tend to adapt to a current appearance mode forgetting previous appearance modes, which often makes them fail when target objects rapidly change appearance modes. Fig. 17 shows example frames from the test sequences.

Discussions

This section proposed MCBQ, a multi-classifier boosting algorithm with a soft partitioning of the input space. This is achieved with a weighting function Q ensuring that coherent clusters are formed. We applied the method to simultaneous tracking and pose estimation. The learned model allows tracking during rapid pose changes, since it captures multiple appearances. Existing single classifier trackers tend to adapt to a single appearance mode, forgetting previous modes. MCBQ can be seen as an extension of MCBoost [31] for the online setting, or a multi-class extension of the MIL tracker [22]. Future work includes a more principled selection of the number of strong classifiers and exploring other choices for the weighting function.

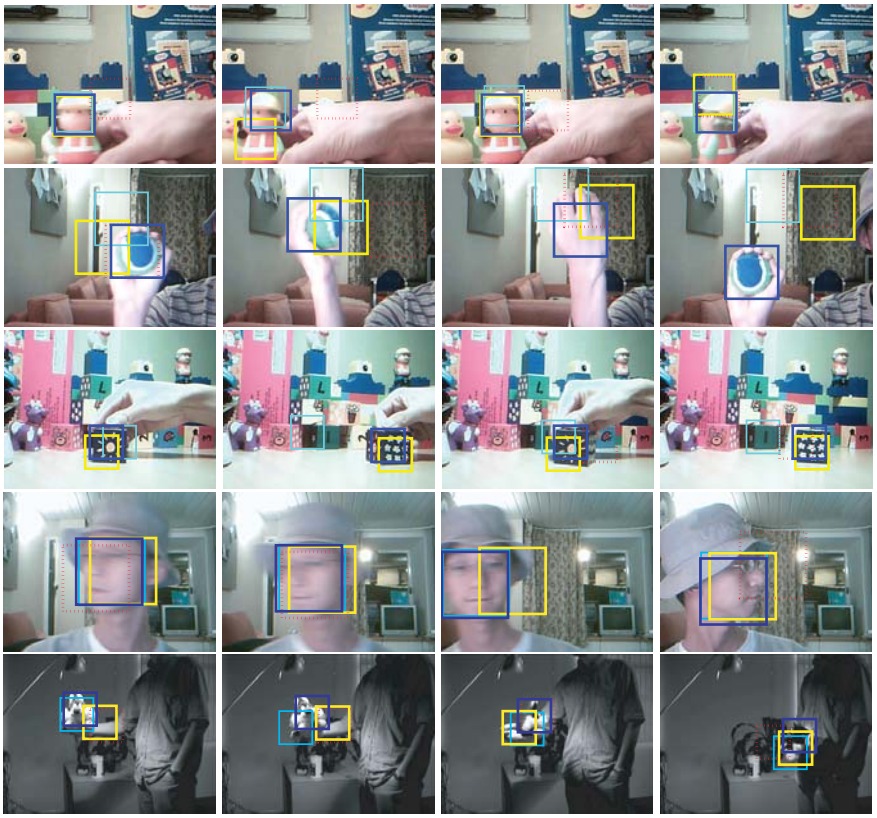


Fig. 17 Example tracking results on test sequences. The comparison shows tracking results for MCBQ (blue), AdaBoost (red), MILTrack (cyan), and SemiBoost (yellow) in the evaluation. See text for details.

4 Conversion of a Boosting Classifier into a Decision Tree by Boolean Optimisation

Boosting is a popular method in object detection [3], tracking [26] and segmentation [33] tasks, which demand very fast classification. Boosting makes a decision by aggregating simple weak-learners e.g., Haar-like features, which are computed very fast on an integral image. Despite its efficiency, it is often required to further reduce the classification time. A cascade of boosting classifiers, which could be seen as a degenerate tree (see Figure 18(a)), effectively improves the classification speed: by filtering out majority of negative class samples in its early stages [3]. Designing a cascade, however, involves manual efforts for setting a number of parameters: the number of classifier stages, the number of weak-learners and the threshold per stage.

In this work, we propose a novel way to reduce down the classification time of a boosting classifier up to an order of magnitude without sacrificing its accuracy,

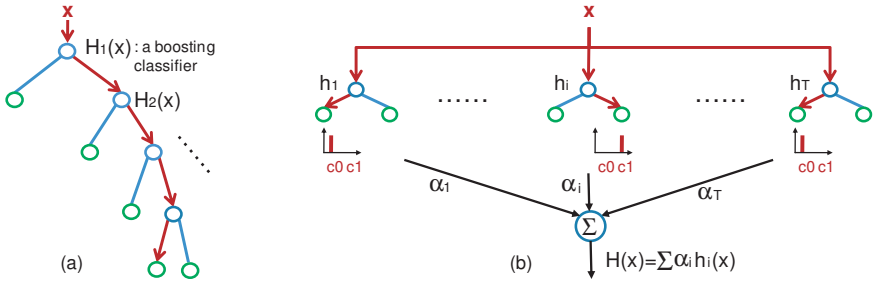


Fig. 18 Boosting as a tree. (a) A boosting cascade is seen as an imbalanced tree, where each node is a boosting classifier. (b) A boosting classifier has a very shallow and flat network where each node is a decision-stump i.e., weak-learner.

not relying on a design of cascade. The chance for improvement comes from the fact that a standard boosting classifier can be seen as a very shallow network, see Figure 18(b), where each weak-learner is a decision-stump and all weak-learners are used to make a decision. The flat structure ensures reasonably smooth decision regions for generalisation, however it is not optimal in classification time. The proposed method converts a shallow network (a boosting classifier as input) to a deep hierarchical structure (a decision tree as output). The obtained tree speeds up a boosting classifier by having many short paths: easy data points are classified by a small number of weak-learners. Since it preserves the same decision regions of the boosting classifier, the method alleviates a highly-overfit behaviour of conventional decision trees. We introduce a novel Boolean optimisation formulation and method. A boosting classifier splits a data space into 2^n primitive regions by n binary weak-learners. The decision regions of the boosting classifier are encoded by the boolean codes and class labels of the primitive regions. A decision tree is then grown using the region information gain. Further details are about a better way of packing the region information and the two stage cascade allowing the conversion with any number of weak-learners. Without designing a many-stage cascade our method offers a convenient way of speeding up, while the method incorporated in such a cascade could provide a further speed-up.

Related Work

For speeding up the classification of a boosting classifier, the shortest set of weak-learners for a given error rate has been obtained by the sequential probability ratio test in the work of Sochman *et al.* [7]. It takes an early exit when the boosting sum reaches a certain value whose sign cannot be altered by the remaining weak-learners. Similarly, Zhou has proposed Fast exit method [35]. This line of methods utilises so called *a single path of varying length*, while our tree method *multiple paths of different lengths* (See Figure 19). The proposed method yields a more optimal speed (see Section 4.2).

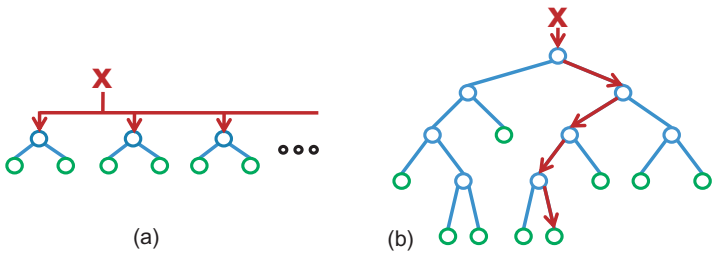


Fig. 19 Fast-exit vs super tree. Fast-exit methods have the structure of a single path of varying lengths (a), while our method yields the structure of multiple paths of different lengths (b).

The closest work to ours is Zhou's [35]. He has introduced representation of a boosting classifier by a Boolean table and implemented a binary decision tree [35]. His solution, however, is a brute force search for all possible tree configurations, which is highly computationally-costly. It therefore affords to only about 5 and 10 weak-learners. The speed gain reported was not significant over a standard boosting classifier and Fast exit method.

Tree-structured multiple boosting classifiers have been proposed for multi-pose or multi-category detection problems. The common structure is a tree hierarchy each path of which is a strong boosting classifier. Torralba *et al.* have proposed sharing weak-learners among multiple boosting classifiers [4] for accelerating classification speed. While Torralba's method requires pre-defined sub-category labels, the methods in [14, 15, 16] automatically learn the sub-category labels for multiple boosting classifiers in a tree. Whereas all these methods are useful for *multiple* boosting classifiers, our work focuses on a *single* boosting classifier. A further conceptual difference lies in that the previous studies [17, 14, 15, 16] present a novel way of learning boosting classifiers and ours takes a boosting classifier learnt in a standard way as input. We do not alter the decision regions of an input classifier but speed it up.

Boolean expression minimisation is to minimize the number of terms and binary variables in the Boolean expression. Algorithms for the minimisation have mainly been studied in the circuit design [38]. Since circuits have strictly predefined specifications, exact minimization was the goal of most studies. The complexity of a logic expression rises exponentially when the number of binary variables increases. Therefore, conventional minimisation methods are limited to a small number of binary variables, typically from a few to about 15 variables [38]. Boolean minimisation has been also applied to size down a redundant decision tree, represented by a Boolean table [39].

4.1 Conversion of a Boosting Classifier into a Tree

Both a boosting classifier and a decision tree are composed of weak-learners (or called decision-stumps/split-nodes). Whereas a boosting classifier places decision

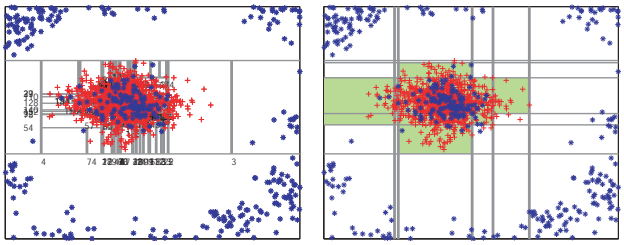


Fig. 20 Decision regions. The decision regions of a boosting classifier (right) are smooth compared to those of a conventional decision tree (left).

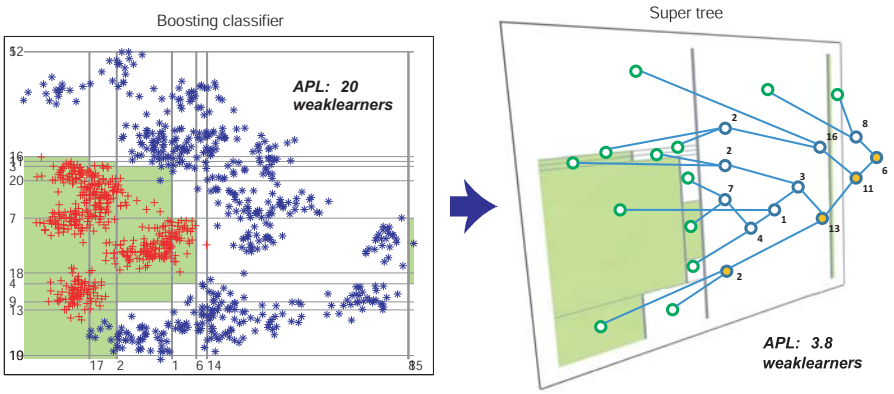


Fig. 21 Converting a boosting classifier into a tree for speeding up. The proposed conversion preserves the Boosting decision regions and has many short paths speeding up 5 times.

stumps in a flat structure, a decision tree has a deep and hierarchical structure (see Figure 18(b) and 21). The different structures lead to different behaviours: Boosting has a better generalisation via reasonably smooth decision regions. See Figure 20 for the decision regions of the two methods. Here a part of negative (blue) data points are scattered in the middle of positive (red) samples. Whereas a conventional decision tree forms complex decision regions trying classification of all training points, a boosting classifier exhibits a reasonable smoothness in decision regions. We propose a method to grow a tree from the decision regions of a boosting classifier. As shown in Figure 21, the tree obtained, called *super tree*, preserves the Boosting decision regions: it places a leaf node on every region that is important to form the identical decision boundary (i.e., accuracy). In the mean time, Super tree has many short paths that reduce the average number of weak-learners to use when classifying a data point. In the example, super tree on average needs 3.8 weak-learners to perform classification whereas the boosting classifier needs 20: all 20 weak-learners are used for every point.

4.1.1 Boolean Optimisation Formulation

A standard boosting classifier is typically represented by the weighted sum of binary weak-learners as

$$\mathrm{H}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathrm{h}_i(\mathbf{x}), \tag{13}$$

where α_i is the weight and h_i the i-th binary weak-learner in $\{-1, 1\}$. The boosting classifier splits a data space into 2^m primitive regions by m binary weak-learners. Regions $R_i, i = 1, ..., 2^m$ are expressed as boolean codes (i.e., each weak-learner h_i corresponds to a binary variable w_i). See Figure 22 for an example, where the boolean table is comprised of 2^3 regions. The region class label c is determined by Equation 13. Region R_8 in the example does not occupy the 2D input space and thus receives the *don't care* label marked "x" being ignored when representing decision regions. The region prior $p(R_i)$ is introduced for data distribution as $p(R_i) = M_i/M$ where M_i and M are the number of data points in the i-th region and in total. The decision regions of the boosting classifier are encoded by a set of regions represented as

$$\begin{cases} B(R_i): \text{ boolean expression} \\ c(R_i): \text{ region class label} \\ p(R_i): \text{ region prior} \end{cases} \tag{14}$$

With the region coding, an optimally short tree is defined in terms of average expected path length of data points as

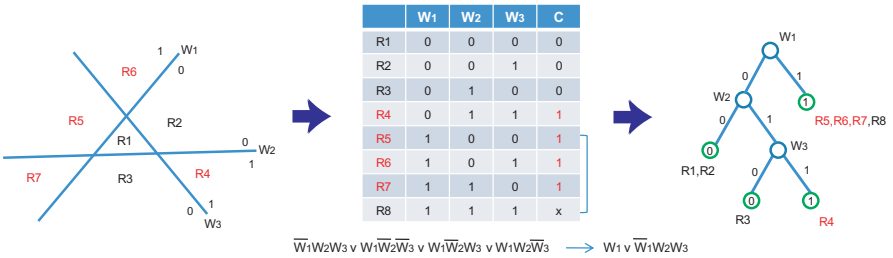
$$\mathbf{T}^* = \min_{\mathbf{T}} \sum_i E(l_{\mathbf{T}}(R_i)) p(R_i), \tag{15}$$

where $\mathbf{T}$ denotes all possible configurations of a decision tree. $E(l_{\mathbf{T}}(R_i))$ is the expected path length of the i-th region in $\mathbf{T}$. The path length is simply the number of weak-learners (or split-nodes) on the path to the i-th region. The decision tree should closely duplicate the decision regions of the boosting classifier as an optimisation constraint: the regions that do not share the same class label $c(R_i)$ must not be put in the same leaf-node of the tree. Any regions of *don't care* labels are allowed to be merged with other regions for the shortest path possible.

The boolean expression for the table in Figure 22 can be minimised by optimally joining the regions that share the same class label or *don't care* label as

$$\begin{gathered} \overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3 \vee \mathrm{w}_1 \overline{\mathrm{w}}_2 \overline{\mathrm{w}}_3 \vee \mathrm{w}_1 \overline{\mathrm{w}}_2 \mathrm{w}_3 \vee \mathrm{w}_1 \mathrm{w}_2 \overline{\mathrm{w}}_3 \\ \longrightarrow \mathrm{w}_1 \vee \overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3 \end{gathered} \tag{16}$$

where $\vee$ denotes OR operator. The minimised expression has a smaller number of terms. Only the two terms, w_1 and $\overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3$ are remained representing the joint regions $R_5 - R_8$ and R_4 respectively. A short tree is then built from the minimised boolean expression by placing more frequent variables at the top of the tree (see Figure 22(right)). The method for Boolean expression minimisation is close, but not suited to our problem that involves a large number of variables i.e., weak-learners.



	W1	W2	W3	C
R1	0	0	0	0
R2	0	0	1	0
R3	0	1	0	0
R4	0	1	1	1
R5	1	0	0	1
R6	1	0	1	1
R7	1	1	0	1
R8	1	1	1	x

Fig. 22 Boolean expression minimisation for an optimally short tree. (a) A boosting classifier splits a space by binary weak learners (left). The regions are represented by the boolean table and the boolean expression is minimised (middle). An optimal short tree is built on the minimum expression (right).

Furthermore, all regions are treated with equal importance in the kind of methods, while an optimally short tree is learnt by considering data distribution i.e., region prior in Equation 15.

4.1.2 Growing a Super Tree

We propose a novel boolean optimisation method for obtaining a reasonably short tree for a large number of weak-learners of a boosting classifier. The classifier information is efficiently packed by using the region coding and a tree is grown by maximising the region information gain. The base algorithm is explained here, see [46] for its limitations and an improved method. The number of primitive regions 2^m is intractable when m is large. Regions R_i that are occupied by any training data points are only taken as input s.t. $p(R_i) > 0$. The number of input regions is thus smaller than the number of data points. Regions with no data points are labeled *don't care*.

Huffman coding [40] is related to our optimisation. It minimises the weighted (by region prior in our problem) path length of code (region). The technique works by creating a binary tree of nodes by maximising the entropy-based information gain. We similarly grow a tree based on the region information gain for an optimally short tree. For a certain weak-learner $w_j, j = 1, ..., m$, the regions in the left split and the right split w.r.t. the weak-learner are readily given from the boolean expressions as

$$\begin{aligned} \mathbf{R}_l &= \{R_i | B(R_i) \wedge \overline{\mathrm{w}}_1 \cdots \mathrm{w}_j \cdots \overline{\mathrm{w}}_m) = 0\} \\ \mathbf{R}_r &= \mathbf{R}_n \setminus \mathbf{R}_l \end{aligned} \tag{17}$$

where $\mathbf{R}_n$ is the set of regions arriving at the node n and $\wedge$ is AND operator. At each node, it is found the weak-learner that maximises

$$\Delta I = -\frac{\sum_{\mathbf{R}_l} p}{\sum_{\mathbf{R}_n} p} E(\mathbf{R}_l) - \frac{\sum_{\mathbf{R}_r} p}{\sum_{\mathbf{R}_n} p} E(\mathbf{R}_r) \tag{18}$$

where p is the region prior and E is the entropy function of the region class distribution, which is

Algorithm: Growing a super tree

Input: a set of data point regions R encoded by $\{B, c, p\}$
Output: a decision tree

1.Start with a root node $n = 1$ containing the list of all regions $\mathbf{R}_n$.
2.For i=1,...,m
3. Spit the node: $(\mathbf{R}_l, \mathbf{R}_r) = \text{split}(\mathbf{R}_n, \text{w}_i)$ by (17).
4. Compute the gain: $\Delta I = \text{gain}(\mathbf{R}_l, \mathbf{R}_r)$ by (18).
5.Find w_i^* that maximises the information gain.
6.If the gain is sufficient, save it as a split node. Else, save it as a leaf node.
7.Go to a child of split node and recurse the steps 2-6 setting $\mathbf{R}_n = \mathbf{R}_l$ or $\mathbf{R}_r$.

Fig. 23 Pseudocode of the algorithm.

$$Q(c^*) = \sum_{\mathbf{R}_c^*} p, \text{ where } \mathbf{R}_c^* = \{R_i | c(R_i) = c^*\}. \tag{19}$$

The node splitting is continued until all regions in a node have the coherent region label. The key ideas in the method have two-folds: 1) growing a tree from the decision regions and 2) using the region prior (data distribution). Compared to conventional decision trees built on data points, the proposed tree is grown upon smooth decision regions, guaranteeing better generalisation. Using the region prior helps getting an optimally short tree in the sense of average path length of data points. See Figure 23 for the pseudo-code of the proposed algorithm.

4.1.3 Cascade of Super Tree and Fast-Exit

Designing a cascade involves a number of parameters to set. The setting is more difficult with more stages. Our solution explained in the previous section can be seen as a convenient way of speeding up a boosting classifier up to several tens of weak-learners without need of a multi-stage cascade. We use a two stage cascade to cope with any larger number of weak-learners of a boosting classifier. It places the super tree in the first stage and the fast-exit method in the second stage. The fast-exit method, which yields the same accuracy as a boosting classifier of any number of weak-learners, is required to meet the target accuracy of a cascade. We first designed a two-stage cascade of standard boosting classifiers in a conventional way, by varying the number of weak-learners (but limiting the number of weak-learners of the first stage to less than a hundred) and the thresholds. Then, the two standard boosting classifiers were replaced with the super-tree and the fast-exit method. The proposed cascade significantly speeds up a two-stage cascade of standard boosting classifiers and the same of the fast-exits at both stages, as well as a single boosting classifier (see Section 4.2).

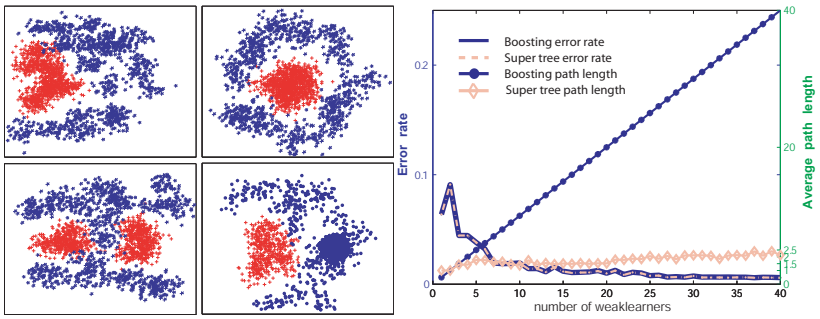


Fig. 24 Experimental results on the synthetic data. Examples of 2D synthetic data sets (left). Super tree obtains the same accuracy as the booting classifier significantly shortening the average path length (right).

4.2 Experiments and Discussions

Classification of Synthetic 2D Data

We have made twelve 2D synthetic data sets. Data points of two classes were generated from Gaussian mixtures as exemplified in Figure 24. The six test sets were created by randomly perturbing the train sets. We have compared the two methods here: a boosting classifier (AnyBoost implementation [9]) and the proposed tree using the data point regions. Vertical and horizontal lines are weak-learners of boosting. Figure 24(right) shows the results. The left and right y-axis in the graph show the classification error rate and the average path length i.e., number of weak-learners used per point respectively. Note first that the both methods do drop the error rate when the number of weak-learners is increased indicating good generalisation. The proposed method exhibited the same accuracy as the boosting classifier for all number of weak-learners. While the boosting classifier linearly increased the average path length for the number of weak-learners, the proposed method quickly converged significantly reducing down the average path length. At 40 weak-learners, the super tree speeds up the boosting classifier by 16 times.

Object Detection

For training, we used the MPEG-7 face data set [41] that has 11,845 face images collected from a few public face data sets such as Yale and XM2VTS, and non-public face data sets. BANCA face set (520 faces) and Caltech background image sets (900 images) were exploited for bootstrapping. The total number of negative-class images for training, which were either bootstrapped or randomly drawn, is 50,128. We used 21,780 Haar-like features on integral images as weak-learners. We have tested on the MIT+CMU frontal face test set [12] which consists of 130 images with 507 labeled frontal faces. The 507 face and 57000 random image patches were cropped and resized into 24x24 images. Example images are shown in

Fig. 25 Example features (weak-learners) and face images used.

No. of weak learners	Boosting			Fast exit (cascade)			Super tree (cascade)		
	False positives	False negatives	Average path length	False positives	False negatives	Average path length	False positives	False negatives	Average path length
20	501	120	20	501	120	11.70	476	122	***7.51***
40	264	126	40	264	126	23.26	231	127	***12.23***
60	222	143	60	222	143	37.24	212	142	***14.38***
100	148	146	100	148 (144)	146 (149)	69.28 (37.4)	(145)	(152)	(***15.1***)
200	120	143	200	120 (146)	143 (148)	146.19 (38.1)	(128)	(146)	(***15.8***)

Fig. 26 Experimental results on the face images. The numbers in the the brackets are for the two-stage cascades.

Figure 25. The methods compared include a standard boosting classifier, Fast exit, the cascade of two Fast exits, Super tree and the two-stage cascade of Super tree and Fast-exit. For the super tree, we used the extended regions [46]. Fixing the accuracy at 0 threshold, we have compared the average path lengths of the methods in Figure 26. For all different numbers of weak-learners, the super tree significantly reduces the average path length of the boosting classifier and the fast exit. The two-stage cascade solution of 60 weak-learner super tree and 200 weak-learner fast exit speeded up the standard boosting by 6.6-12.7 times and even the two-stage cascade of 60 and 200 weak-learner fast exits by 2.5 times. Note that the super tree exploits various combinations of weak-learners (i.e., paths) for an optimal classification speed, whereas the fast exit takes the combinations always in the order of the weak-learner weights. One can also compare the results of [35] with ours using the standard boosting and the fast-exit as proxies. Whereas the solution in [35] didn't gain much over the boosting and the fast exit method, ours significantly improved the both. More importantly, the method [35] has been tested only for 5 or 10 weak learners whereas our method in a single stage is conveniently scalable up to several tens of weak learners.

Single conventional decision trees of various pruning [37] were very poor. The best accuracy among those of the different pruned trees (false positives: 1995/false negatives: 120) is by far worse than that of the super tree of 20 weak-learners (false positives: 476/false negatives: 122). The super tree was even shorter than the decision tree: the depth of the super tree and conventional tree was about 7.5 and 9 respectively.

Fig. 27 Segmentation results. Pixels classified into the building class by Super tree (or Boosting) are shown by a darker hue.

Although the comparison has been made on the two-stage cascades, the proposed method affords a speed up over a standard multi-stage boosting cascade by replacing each stage of a boosting classifier in the cascade with a Super Tree. The speed gains over the different numbers of weak-learners in a single stage boosting classifier are reported in Figure 26. In the other sense, the proposed method can be seen as a convenient way of obtaining the comparable speed-up to a multi-stage cascade by the single super tree (or the proposed two-stage cascade).

Segmentation by pixel-wise classification

The car driving sequences [42] were exploited for the experiment. Boosting classifier and super tree were trained for the binary problem for the building class against non-building class. 1323 DCT features were drawn from 21x21 RGB image patch as weak-learners. The train set consisted of 7143 positive and 23217 negative pixels from 184 images of 11x15 pixel resolution. Randomisation in learning (similarly to [45]) reduced the train time of the boosting classifier. The test set contained 38445 points from 233 images. The correct recognition rate of Boosting of 40 weak-learners was 0.71 (as global accuracy) or 0.736 (as average class accuracy). The super tree learnt by 10 extended regions per region obtained the close accuracy as 0.70 (as global accuracy) or 0.728 (as average class accuracy) using only 15 weak-learners on average. The accuracy obtained seems comparable to [42]. Figure 27 shows the segmentation results. The blocky effect was due to the low pixel image resolution used.

Discussions

We have proposed a novel way to speed up a boosting classifier. The problem is formularised as boolean optimisation and a new optimisation method is proposed for a large number of weak-learners. The tree grown from the decision regions of a boosting classifier, called Super tree, provides many short paths and preserves the

Boosting decision regions. The single super tree delivers the close accuracy to a boosting classifier with a great speed-up for up to several tens of weak-learners. The proposed two stage cascade allows any number of weak-learners. Experiments have shown that the tree obtained is reasonably short in terms of average path length outperforming a standard boosting classifier, fast exit, their cascade. The method has been also demonstrated for segmentation problems.

5 Summary and Conclusion

We have formulated a novel discriminative co-clustering problem of images and features, and have presented the solution called MCBoost by simultaneously learning multiple boosting classifiers. Each boosting classifier in the method cooperates and competes with others, taking expertise on a subset of images. The method has been shown to yield meaningful co-clusters of images and features, significantly outperforming the conventional designs of single or multiple boosting classifiers.

The MCBoost method has been extended into a online version with a soft partitioning of the input space for object tracking. It incorporated a weighting function $\mathcal{Q}$, which ensures that coherent clusters are formed, in the MCBoost framework. Whereas existing single classifier trackers tend to adapt to a single appearance mode, forgetting previous modes, the method called MCBQ allows tracking during rapid pose changes, since it captures multiple appearances. The method can also be seen as a multi-class extension of the MIL tracker [22].

Lastly, we have proposed a novel way to convert a standard boosting classifier into a decision tree for speeding up. The conversion problem is formularised as boolean optimisation and a new optimisation method is proposed for a large number of weak-learners. The tree grown from the decision regions of a boosting classifier, called Super tree, provides many short paths (i.e., speeding up the evaluation time) and preserves the Boosting decision regions (i.e., the same accuracy). In the experiments, the super tree outperformed a standard boosting classifier, fast exit, their cascade in speed, delivering the same accuracy to that of an input boosting classifier.

Many visual recognition problems are formulated as classification problems of image sub-windows. Every possible sub-window, across pixels and scales, is evaluated to decide whether it contains an object of interest or not. The number of sub-windows is often massive, requiring the evaluation of each sub-window in a very fast manner. This study has presented required efficient classification methods. We began with a standard boosting method and have introduced largely three extensions of it to improve the performance in both accuracy and time. Major issues for future work include a more principled way to select the number of boosting classifiers in MCBoost and to form a forest of Super Trees by randomisation (similarity to Random Forests [36]). See also the discussions of each section.

Acknowledgements. Section2, Section 3 and Section 4 have been compiled from the authors' previous publications [31, 34] and [46] respectively.

References

1. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 89–98 (2003)
2. Jordan, M.I., Jacobs, R.A.: Hierarchical mixture of experts and the EM algorithm. Neural Computation 6(2), 181–214 (1994)
3. Viola, P., Jones, M.: Robust real-time object detection. Int'l J. Computer Vision 57(2), 137–154 (2002)
4. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE Trans. on PAMI 29(5), 854–869 (2007)
5. Viola, P., Platt, J.C., Zhang, C.: Multiple Instance Boosting for Object Detection. In: Proc. Advances in Neural Information Processing Systems, pp. 1417–1426 (2006)
6. Li, S.Z., Zhang, Z.: Floatboost learning and statistical face detection. IEEE Trans. on PAMI 26(9), 1112–1123 (2004)
7. Sochman, J., Matas, J.: Waldboost - learning for time constrained sequential detection. Proc. CVPR 2, 150–157 (June 2005)
8. Schapire, R.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
9. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Proc. Advances in Neural Information Processing Systems, pp. 512–518 (2000)
10. Sim, T., Baker, S., Bsat, M.: The CMU Pose, Illumination, and Expression Database. IEEE Trans. on PAMI 25(12), 1615–1618 (2003)
11. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Proc. CVPR, pp. 886–893 (2005)
12. Rowley, H.A., Baluja, S., Kanade, T.: Neural Network-Based Face Detection. IEEE Trans. on PAMI 20(1), 23–38 (1998)
13. Schneiderman, H., Kanade, T.: A Statistical Model for 3D Object Detection Applied to Faces and Cars. In: Proc. CVPR (June 2000)
14. Wu, B., Nevatia, R.: Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In: Proc. ICCV (2007)
15. Huang, C., Ai, H., Li, Y., Lao, S.: Vector Boosting for Rotation Invariant Multi-View Face Detection. In: Proc. ICCV (2005)
16. Tu, Z.: Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In: Proc. ICCV (2005)
17. Grossmann, E.: AdaTree: boosting a weak classifier into a decision tree. In: IEEE Workshop on Learning in Computer Vision and Pattern Recognition, p. 105 (2004)
18. Babenko, B., Dollár, P., Tu, Z., Belongie, S.: Simultaneous learning and alignment: Multi-instance and multi-pose learning. In: ECCV Workshop on Faces in Real-Life Images (2008)
19. Wojek, C., Walk, S., Schiele, B.: Multi-Cue Onboard Pedestrian Detection. In: Proc. CVPR (2009)
20. Pham, M.T., Cham, T.J.: Fast training and selection of Haar features using statistics in boosting-based face detection. In: Proc. ICCV (2007)
21. Avidan, S.: Ensemble tracking. IEEE Trans. PAMI 29(2), 261–271 (2007)
22. Babenko, B., Yang, M.-H., Belongie, S.: Visual tracking with online multiple instance learning. In: Proc. CVPR, Miami, FL (June 2009)
23. Black, M.J., Jepson, A.: Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. In: Buxton, B.F., Cipolla, R. (eds.) ECCV 1996. LNCS, vol. 1064, pp. 329–342. Springer, Heidelberg (1996)
24. Collins, R., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. on PAMI 27(10), 1631–1643 (2005)
25. Freund, Y., Schapire, R.: A decision theoretic generalization of on-line learning and an application to boosting. J. of Computer and System Sciences 55(1), 119–139 (1997)
26. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proc. CVPR, vol. 1, pp. 260–267 (2006)

27. Grabner, H., Leistner, C., Bischof, H.: Semi-Supervised On-line Boosting for Robust Tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
28. Hall, P., Marshall, D., Martin, R.: Merging and splitting eigenspace models. IEEE Trans. on PAMI 22(9), 1042–1049 (2000)
29. Jebara, T., Pentland, A.: Parameterized structure from motion for 3d adaptive feedback tracking of faces. In: Proc. CVPR, pp. 144–150 (June 1997)
30. Jones, M., Viola, P.: Fast multi-view face detection. Technical Report 96, MERL (2003)
31. Kim, T.-K., Cipolla, R.: MCBoost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In: Proc. Advances in Neural Information Processing Systems, Vancouver, Canada (December 2008)
32. Lee, K.-C., Ho, J., Yang, M.-H., Kriegman, D.: Visual tracking and recognition using probabilistic appearance manifolds. Computer Vision and Image Understanding 99(3), 303–331 (2005)
33. Avidan, S.: SpatialBoost: Adding Spatial Reasoning to AdaBoost. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 386–396. Springer, Heidelberg (2006)
34. Kim, T.-K., Woodley, T., Stenger, B., Cipolla, R.: Online Multiple Classifier Boosting for Object Tracking. In: Proc. of IEEE CVPR Workshop on Online Learning for Computer Vision, San Francisco, USA (June 2010)
35. Zhou, S.: A binary decision tree implementation of a boosted strong classifier. In: IEEE Workshop on Analysis and Modeling of Faces and Gestures, pp. 198–212 (2005)
36. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
37. Quinlan, J.: Bagging, boosting, and c4.5. In: Proc. National. Conf. on Artificial Intelligence, pp. 725–730 (1996)
38. Schwender, H.: Minimization of Boolean Expressions Using Matrix Algebra, Technical report, Collaborative Research Center SFB 475. University of Dortmund (2007)
39. Chen, J.: Application of Boolean expression minimization to learning via hierarchical generalization. In: Proc. ACM Symposium on Applied Computing, pp. 303–307 (1994)
40. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill (2001)
41. Kim, T.-K., Kim, H., Hwang, W., Kittler, J.: Component-based LDA Face Description for Image Retrieval and MPEG-7 Standardisation. Image and Vision Computing 23(7), 631–642 (2005)
42. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and Recognition Using Structure from Motion Point Clouds. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 44–57. Springer, Heidelberg (2008)
43. Basak, J.: Online adaptive decision trees. Journal of Neural Computation 16, 1959–1981 (2004)
44. Yeh, T., Lee, J., Darrell, T.: Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning. In: Proc. ICCV (2007)
45. Rahimi, A., Recht, B.: Random Kitchen Sinks: Replacing Optimization with Randomization in Learning. In: Proc. Neural Information Processing Systems (2008)
46. Kim, T.-K., Budvytis, I., Cipolla, R.: Making a Shallow Network Deep: Growing a Tree from Decision Regions of a Boosting Classifier. In: Proc. of British Machine Vision Conference, Aberystwyth, UK (2010)