Detection with Multi-exit Asymmetric Boosting

Minh-Tri Pham Viet-Dung D. Hoang Tat-Jen Cham

School of Computer Engineering
Nanyang Technological University
Singapore

mtpham@ntu.edu.sg hken@pmail.ntu.edu.sg astjcham@ntu.edu.sg

Abstract

We introduce a generalized representation for a boosted classifier with multiple exit nodes, and propose a method to training which combines the idea of propagating scores across boosted classifiers [14, 17] and the use of asymmetric goals [13]. A means for determining the ideal constant asymmetric goal is provided, which is theoretically justified under a conservative bound on the ROC operating point target and empirically near-optimal under the exact bound. Moreover, our method automatically minimizes the number of weak classifiers, avoiding the need to retrain a boosted classifier multiple times for empirical best performance as in conventional methods. Experimental results shows significant reduction in training time and number of weak classifiers, as well as better accuracy, compared to conventional cascades and multi-exit boosted classifiers.

1. Introduction

Cascading boosted classifiers has been a successful approach in appearance-based detection since the seminal work of Viola and Jones [12] on face detection. The key insight of a cascade is to decompose a detection problem into a sequence of binary classification sub-problems of increasing difficulty. Positively predicted examples from the boosted classifier for a sub-problem are used to train the boosted classifier for the next sub-problem, while negatively predicted examples are discarded. The final cascade obtained from this bootstrapping process often has high detection rate and extremely low false acceptance rate, while the early rejection mechanism allows the cascade to scan through a large set of examples for a rare detection event in a small amount of time.

Despite its utility in detection, the cascade approach imposes a number of issues in learning. At the stage level, one has to devise a learning strategy to find an optimal trade-off among three important factors of a boosted classifier: the detection rate, the false acceptance rate, and the number of weak classifiers. To maintain high detection rate and extremely low false acceptance rate for the overall cascade, each individual boosted classifier must ensure close-to-one detection rate and moderate false acceptance rate. It is essential to minimize the number of weak classifiers of a boosted classifier, as it is roughly proportional to the running time of the classifier. Conventional methods often use AdaBoost or one of its variants [3, 10] to train a boosted classifier with a fixed maximum number of weak classifiers. To find the best trade-off among the three factors, one has to re-train the classifier multiple times and choose the best candidate manually.

The overall detection rate of a cascade is the product of detection rates associated with all individual boosted classifiers in the cascade; similarly, the overall false acceptance rate is the product of all classifier false acceptance rates. However, it is not known beforehand how many classifiers are needed, nor which combination of ROC operating points (each defined by a detection rate and false acceptance rate) produces an optimal cascade. Currently, these parameters are obtained mainly by trial and error, though some progress has been made [2, 11].

In this paper, we introduce a notion called multi-exit boosted classifier to describe a boosted classifier *with multiple exits*. Each exit is associated with a weak classifier. A rejection decision at an exit is made if the intermediate boosted score, up to the associated weak classifier, is below a threshold. Some cascade variants can be cast as a multi-exit boosted classifier. We analyze recent cascade training methods in terms of training goals, and show that many of them result in training and/or using sub-optimal weak classifiers. We propose a method to train a multi-exit boosted classifier by minimizing the number of weak classifiers needed to achieve the desired detection and false acceptance rates simultaneously. It also removes the need to run multiple ad hoc trials to discover the best boosted

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

classifiers that satisfy the operating point requirements.

The remaining parts of the paper are organized as follows. In section 2, we define multi-exit boosted classifiers and cast both the normal cascade and regular boosted classifier as special cases. An analysis of recent cascade training methods is also offered in this section. In section 3, we describe our method to train a multi-exit boosted classifier, and discuss about important aspects in designing the method. Experimental results are presented in section 4.

The key contributions of the paper are:

- a generalized model that unifies existing models such as conventional boosted classifiers, cascades, and more recent multi-exit boosted classifiers;
- a new multi-exit asymmetric boosting method incorporating asymmetric training goals to achieve ROC operating point targets while minimizing the number of weak classifiers;
- a principled formulation of an asymmetric goal that is theoretically optimal for a conservative bound on an ROC operating point requirement, and empirically near-optimal for the exact bound; and
- results demonstrating that the combined framework outperforms existing methods.

2. Overview

2.1. General framework

In this section, we introduce a new model of which cascades and multi-exit boosted classifiers are defined as special cases. In section 2.2, we use the model to point out disadvantages of existing methods.

We restrict ourselves to a problem of imbalanced binary classification $C : \mathcal{X} \rightarrow \{-1, 1\}$ where the prior probability of the negative class far outweighs the prior probability of the positive class, *i.e.*, $P(y = 1) \ll P(y = -1)$ with y being the class label. We consider the following model:

$$C(\mathbf{x}) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } H_m(\mathbf{x}) \geq \theta_m \quad \forall m \in \mathcal{M} \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

$$H_m(\mathbf{x}) \overset{\text{def}}{=} \sum_{i=\mu(m)}^{m} h_i(\mathbf{x}), \tag{2}$$

In this model, there are M weak classifiers denoted in sequence by $h_i(\mathbf{x})$ with $i = 1, \ldots, M$, where $h_i : \mathcal{X} \rightarrow \mathbb{R}$; in the case of discrete-type weak classifiers, $h_i(\mathbf{x}) = c_i f_i(\mathbf{x})$ with $f_i : \mathcal{X} \rightarrow \{-1, 1\}$ and coefficient $c_i \in \mathbb{R}$. Out of these M weak classifiers, we specify a subset that acts as *exit nodes*, represented by a set of indices $\mathcal{M} \subset \{1, \ldots, M\}$. We assume that the last classifier is always an exit node, hence $M \in \mathcal{M}$. Associated with each exit node is a corresponding *entrance node*, represented through the function $\mu(m)$ which is an index to a weak classifier earlier in the sequence. The boosted classifier comprising the weak classifiers between a pair of entrance and exit nodes is associated with $H_m(\mathbf{x})$. Note that while each exit node is a unique weak classifier, entrance nodes may be shared (*i.e.*, they map to the same weak classifier).

This model is general in that it encompasses a range of existing and new models, *e.g.*, (a) the normal boosted classifier in this model is simply a single-exit boosted classifier utilizing all the weak classifiers, defined by $\mathcal{M} = \{M\}$ and $\mu(m) = 1$; and (b) the conventional cascade is represented by defining $\mu(m) = m_0 + 1$ where m_0 is the largest index in $\mathcal{M}$ satisfying $m_0 < m$, or $m_0 = 0$ if m is already the smallest index in $\mathcal{M}$. Conventional cascades as expected have entrance-exit intervals that are non-overlapping.

The variant we explore in this paper is the single boosted classifier with a single entrance but *multiple exits*. This model is characterized by $\mu(m) = 1$ and $|\mathcal{M}| > 1$; this relates to multiple exit nodes sharing the same entrance node at the first weak classifier. A special case of the multi-exit boosted classifier is the soft/dynamic cascade [1, 16] in which $\mathcal{M} = \{1, \ldots, M\}$. Other more complex variants exist that await future analysis.

In learning the m-th weak classifier for model C using AdaBoost or one of its variants [3, 10], the most important factor is the discrete weight distribution $\mathbf{w}^{(m)}$ associated with the training set provided to the weak classifier. It is often possible to express $\mathbf{w}^{(m)}$ in the form

$$w_n^{(m)} = Z_m^{-1} \exp(-y_n(S_m(\mathbf{x}_n) - b_m)), \tag{3}$$

where Z_m is the normalization factor to make $\mathbf{w}^{(m)}$ a distribution, b_m is a threshold to adjust the trade-off between false acceptance rate and false rejection rate when training the m-th weak classifier, and $S_m(\mathbf{x}_n)$ is a score function of input point $\mathbf{x}_n$, defined as:

$$S_m(\mathbf{x}) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } m = 1, \text{ or } \mu(m) = m \\ H_{m-1}(\mathbf{x}) & \text{otherwise} \end{cases} \tag{4}$$

In the original versions of AdaBoost [10], there are no thresholds b_m. It is known from literature [3, 10] that in such cases, the minimizer $h_m(\mathbf{x})$ of the classification error of the weighted training set $\{(\mathbf{x}_n, y_n, w_n^{(m)})\}_{n=1}^N$ is approximately the minimizer of an exponential loss:

$$E[\exp(-y(S_m(\mathbf{x}) + h(\mathbf{x})))]. \tag{5}$$

By introducing $b_m \neq 0$, the exponential loss becomes:

$$E[\exp(-y(S_m(\mathbf{x}) + h(\mathbf{x}) - b_m))], \tag{6}$$

and is an upper bound of an *asymmetric* goal [13]:

$$\begin{aligned} \mathcal{J}'_m(h) &= P(y=-1)e^{-b_m} E[e^{(S_m(\mathbf{x})+h(\mathbf{x}))} | y = -1] \\ &\quad + P(y=1)e^{b_m} E[e^{-(S_m(\mathbf{x})+h(\mathbf{x}))} | y = 1] \\ &\geq P(y=-1)e^{-b_m} FAR(S_m + h) \\ &\quad + P(y=1)e^{b_m} FRR(S_m + h), \end{aligned} \tag{7}$$

where $FAR(f) = E[1_{[f(\mathbf{x})>=0]}|y = -1]$ and $FRR(f) = E[1_{[f(\mathbf{x})<0]}|y = 1]$ is the false acceptance rate and the false rejection rate respectively of a function $f(\mathbf{x})$.

2.2. Related work

In their original work [12], Viola and Jones sampled N_1 positive examples and N_2 negative examples, and initially set weights $0.5/N_1$ for positive examples and $0.5/N_2$ for negative examples, respectively. Effectively, this removed the prior probabilities not only from training the first weak classifier, but also from training subsequent weak classifiers. Since they did not use any threshold b_m, the goal at training the m-th weak classifier became finding a function h that minimizes an upper bound of:

$$J_m(h) = FAR(S_m + h) + FRR(S_m + h). \quad (8)$$

This symmetric goal, however, did not guarantee high detection rate for a boosted classifier. Viola and Jones proposed to threshold the boosted classifier's score $H_m(\mathbf{x})$ by a value $\theta_m \neq 0$. By adjusting θ_m, they were able to achieve high detection rate (and also high false acceptance rate). Many recent methods followed this strategy [1, 2, 4, 5, 11, 14, 16, 17]. While this approach is computationally trivial, it is also sub-optimal in that the θ_m-defined ROC operating point was not arrived at by training the weak classifiers with the asymmetric goal in (6)[1].

In a subsequent paper [13], Viola and Jones addressed this problem by introducing an asymmetric goal for training a boosted classifier:

$$J'_m(h) = FAR(S_m + h) + \lambda FRR(S_m + h), \quad (9)$$

where λ is related to b_m by:

$$b_m = 0.5 \log \lambda. \quad (10)$$

There were no suggestions on how λ should be selected. While Pham and Cham [8] proposed a skewness balancing method for selecting λ, they require that the number of weak classifiers be known in advance.

Xiao *et al.* [17] and Wu *et al.* [14] independently noted that scores obtained from the previous boosted classifier may be exploited downstream, and proposed to propagate the previous scores from one boosted classifier to the next. Though their weak classifiers were trained using the symmetric goal in (8), they showed significant improvements in their empirical results. Nevertheless, their cascades may be considered the initial boosted classifiers with multiple exits.

Bourdev and Brandt [1] trained a very long boosted classifier using the symmetric goal and subsequently utilized a calibration algorithm to break this strong classifier into a cascade, through a rejection threshold at every weak classifier. Because the thresholds were obtained *after* all the weak classifiers were trained, the weak classifiers became even less optimal *w.r.t.* the goal of the cascade. Xiao *et al.* [16] corrected this by updating the training set *before* training each weak classifier. Nevertheless, the symmetric goal was employed by both these methods to train weak classifiers, resulting in sub-optimal feature selection. While the approach of making decisions at each weak classifier is potentially interesting, a number of issues are raised. First, bootstrapping is required at every weak classifier, incurring significant extra computational cost. Second and more importantly, the requirement for early rejection diminishes as progress is made down the cascade, where the problem becomes dominated by accuracy (as classification becomes harder) as opposed to speed (since most of the obvious negative samples will have already been rejected earlier). In such cases, having a decision made at each weak classifier effectively discards important information that may have been exploited if decisions were postponed until further downstream, leading to less accurate classification.

Another problem with the methods above is that the number of weak classifiers per boosted classifier must be specified prior to training. This number is important because it is a trade-off between a better ROC operating point and a shorter running time for a boosted classifier. Tuning the number in an ad hoc fashion is difficult and time consuming. Our approach avoids this by choosing a proper asymmetric goal that greedily minimizes the number of weak classifiers to reach the target detection rate and false acceptance rate simultaneously.

3. Multi-exit Asymmetric Boosting

3.1. Overview of the method

In what follows, we propose a method, called multi-exit asymmetric boosting, to train a multi-exit boosted classifier using an asymmetric learning goal (algorithm 1). Unlike conventional cascades, a multi-exit boosted classifier may be considered as a collection of boosted classifiers that *share* overlapping sets of weak classifiers. Training multi-exit boosted classifiers thus entails a number of added complexities. Similar to Xiao *et al.* [17], we incorporate bootstrapping and AdaBoost into a single method.

We use the same strategy as conventional methods [12, 13] in designing the target minimum detection rate and maximum false acceptance rate at every exit node. That is, we are interested in considering a rejection decision only when the false acceptance rate is below α_0 and detection rate over $1 - \beta_0$, where α_0 and β_0 are predefined as part of the cascade design. The problem of finding the optimal combination of desired rates for all exit nodes are not ad-

[1]Consequently the ROC curve obtained by varying θ_m upper-bounds the error expressed by the "proper" ROC curve obtained by varying b_m.

Algorithm 1 Multi-exit Asymmetric Boosting

```
Require:
    a generator $G$ that produces i.i.d. training examples
    maximum false acceptance rate $\alpha_0$
    maximum false rejection rate $\beta_0$
    a stopping condition $D$ {see section 3.1}
1:  $M_0 = m = 0$
2:  $\mathcal{M} = \emptyset$
3:  $\lambda = \alpha_0/\beta_0$ {see section 3.2}
4:  Generate a training set $\mathcal{Z} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_1+N_2}$ of $N_1$
    positive examples and $N_2$ negative examples
5:  repeat
6:    Initialize weights: {see section 3.3}
        $w_n = e^{-(S_m(\mathbf{x}_n)-0.5\log\lambda)}/N_1 \quad \forall n : y_n = 1$
        $w_n = e^{(S_m(\mathbf{x}_n)-0.5\log\lambda)}/N_2 \quad \forall n : y_n = -1$
7:    repeat
8:      $m = m + 1$
9:      Normalize weights $w_n$
10:     Train $f_m(\mathbf{x})$ using $\{(\mathbf{x}_n, y_n, w_n)\}_{n=1}^{N_1+N_2}$
11:     Find the weighted training error of $f_m$:
          $e_m = \sum_{n=1}^{N_1+N_2} w_n 1_{[y_n \neq f_m(\mathbf{x}_n)]}$
12:     Compute the corresponding coefficient:
          $c_m = 0.5\log\frac{1-e_m}{e_m}$
13:     Update weights: {see section 3.3}
          $w_n = w_n e^{-y_n c_m f_m(\mathbf{x}_n)} \quad \forall n$
14:     Estimate the training error rates:
          $\alpha_m = \frac{1}{N_2}\sum_{n:y_n=-1} 1_{[H_m(\mathbf{x}_n)>=0]}$
          $\beta_m = \frac{1}{N_1}\sum_{n:y_n=1} 1_{[H_m(\mathbf{x}_n)<0]}$
15:   until $D$ or $(\alpha_m \leq \alpha_0 \wedge \beta_m \leq \beta_0)$
16:   if not $D$ then
17:     $M_0 = m$
18:     $\mathcal{M} = \mathcal{M} \cup \{m\}$
19:     Re-generate $\mathcal{Z} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_1+N_2}$ of $N_1$ posi-
        tive examples and $N_2$ negative examples such that
        $H_m(\mathbf{x}_n) \geq 0 \quad \forall m \in \mathcal{M}, n \in \{1, \ldots, N_1 + N_2\}$
20:   end if
21: until $D$
22: return $\mathcal{M}$ and $\{(c_i, f_i)\}_{i=1}^{M_0}$ as the parameters of $C$
```

dressed in this paper, and it is assumed that (α_0, β_0) is provided for every exit node.

Line 16 to line 20 is a bootstrap to generate new examples after a rejection decision is made. As the sub-problem gets harder, it is possible that the desired rates may not be achievable. Therefore, a stopping condition D is needed. In our experiments, we stopped training when the number of weak classifiers between two exit nodes exceeds 200. Conventional methods [12, 13] also used at most 200 weak classifiers to train a boosted classifier.

3.2. Analysis of asymmetric goals

One of the main contributions of our approach is to select a proper asymmetric goal that greedily minimizes the number of weak classifiers to achieve detection rate over $1-\beta_0$ and false acceptance rate below α_0 concurrently. Before analyzing how the choice of an asymmetric goal affects the final boosted classifier, let us assume for the moment that AdaBoost minimizes an asymmetric goal instead of its exponential upper bound[2]. For notational simplicity, we denote the false rejection rate and the false acceptance rate of a classifier H as $\beta(H)$ and $\alpha(H)$ respectively. In the conventional methods, one often trains a boosted classifier using AdaBoost with the symmetric goal in (8). If we plot all the training ROC operating points $(\beta(H_m), \alpha(H_m))^{\mathrm{T}}$ with an increasing number of weak classifiers $m = 1, 2, \ldots$ we obtain a set of points in figure 1a (shown in red).

In AdaBoost and many of its variants [3, 10], when an m-th weak classifier is trained, all other weak classifiers are considered fixed. The proper ROC curve of the new boosted classifier is simply the locus of all operating points of the boosted classifier obtained through training the m-th weak classifier with different *asymmetric goals* (as opposed to simply varying the threshold on the symmetric goal function). Because of the immense computational effort involved, the proper ROC curve is seldom obtained in entirety for training purposes; instead only a single operating point is derived for each weak classifier. While most of the proper ROC curve remains unknown, a number of ROC curve properties may be inferred as described below.

Let the unknown training ROC curve be denoted by Q_m and expressed in the form

$$L_m(\beta, \alpha) = 0 \tag{11}$$

with each instance of (β, α) that satisfies (11) representing an ROC operating point trained with a different asymmetric goal. In this paper, we make the assumption that Q_m is first-order continuous. Equation (9) representing an asymmetric goal may be rewritten as

$$G_\lambda(\beta, \alpha) = \lambda\beta + \alpha, \tag{12}$$

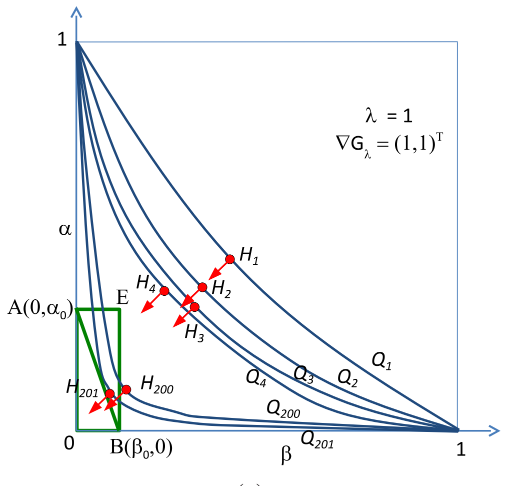
with a controlling parameter λ. When using G_λ to train the m-th weak classifier, it becomes:

$$\begin{aligned} J_{m,\lambda}(h) &= G_\lambda(\beta(S_m + h), \alpha(S_m + h)) && (13) \\ &= \lambda\beta(S_m + h) + \alpha(S_m + h) && (14) \end{aligned}$$

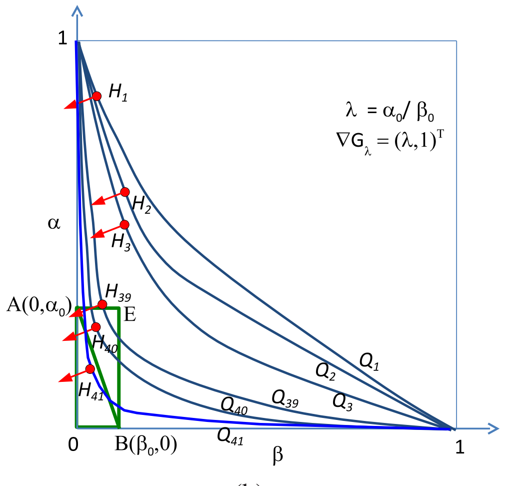
We assume the resulted weak classifier h_m minimizes $J_{m,\lambda}(h)$ *globally*:

$$h_m = \arg\min_h J_{m,\lambda}(h). \tag{15}$$

[2]In practice, because the minimization is applied on the exponential upper bound, the value of the true goal tends to exhibit a locally oscillating but globally decreasing behavior *w.r.t.* the number of weak classifiers.



(a)



(b)

Figure 1. Illustration of how the operating point of a boosted classifier moves when more weak classifiers are trained. Blue solid curves are equivalent to the ROC curves of the boosted classifier. (a): A case when the symmetric goal G_1 is used, *i.e.*, $\lambda = 1$. (b): A case when an asymmetric goal G_λ is used with $\lambda = \alpha_0/\beta_0$.

As in conventional methods, the boosted classifier is updated via

$$H_m = S_m + h_m. \tag{16}$$

Training the weak classifier can be cast as a minimization problem *w.r.t.* (β, α), with objective function $G_\lambda(\beta, \alpha)$, and a constraint $L_m(\beta, \alpha) = 0$. Thus, at any solution point (if it exists) lying on Q_m, the gradient of goal G_λ (*i.e.*, ∇G_λ) has to be perpendicular to the ROC curve Q_m. Note that ∇G_λ is independent of β and α once λ is fixed.

By using the symmetric goal G_1 (*i.e.*, $\lambda = 1$) as in conventional methods, one effectively chooses an operating point on Q_m at which the gradient is perpendicular to $\nabla G_1 = (1, 1)^\mathrm{T}$. If instead we choose an asymmetric goal G_λ for some $\lambda \geq 0$, we recover a different operating point on Q_m with gradient perpendicular to $G_\lambda = (\lambda, 1)^\mathrm{T}$. By varying λ from 0 to ∞ and retraining the weak classifier each time, we can reconstruct the ROC curve Q_m. Though we have no knowledge of Q_m prior to training a weak classifier, we can somewhat control where the next operating point might be located, by controlling λ.

If we further assume that Q_m is not only first-order continuous but smoothly varies as λ is changed, then a key observation is that Q_m *must be convex*. This is because as λ is varied from 0 to ∞, the gradient ∇G_λ monotonically changes from vertical to horizontal uniformly throughout the α-β space. Convexity is guaranteed as the operating point (β, α) for a particular λ must not only be on a point of Q_m perpendicular to ∇G_λ but also be the global minimum of G_λ.

Another important point is, if we use the same λ to train every weak classifier of a boosted classifier, convergence is guaranteed due to iterative decreases in values of a *single* goal function. If we use different goals (*i.e.*, different λ) to train different weak classifiers, it is possible that when a goal function decreases, other goal functions increase. Hence, no convergence is guaranteed. Despite this, previous work shows empirically reasonable results when different goals are used for different weak classifiers [8, 13], but they require that the number of weak classifiers be specified before training. Therefore, to ensure convergence and also to avoid the need to specify *apriori* the number of weak classifiers to be trained, we use the same λ to train every weak classifier of a boosted classifier.

Generally, the training process should be stopped once we reach an operating point in rectangle $OAEB$ in figure 1, which represents an exact bound ensuring detection rate over $1 - \beta_0$ and false acceptance rate below α_0. However, let us focus our attention on a more conservative but easier-to-analyze bound, expressed by the triangle OAB. This triangle has an interesting property as follows.

Theorem 3.1. *If Q_m penetrates triangle OAB for some m, then the use of goal G_{λ^*} with $\lambda^* = \alpha_0/\beta_0$ ensures that point $(\beta(H_m), \alpha(H_m))^T$ is located inside the triangle.*

Proof. Because Q_m is first-order continuous and intersects with the line segment AB at two locations, there exists a point (β^*, α^*) on Q_m inside triangle OAB at which the gradient of Q_m is parallel to AB. Alternatively:

$$\nabla L_m(\beta^*, \alpha^*) \parallel (\alpha_0, \beta_0)^\mathrm{T}. \tag{17}$$

If we use $\lambda^* = \alpha_0/\beta_0$ to train the m-th weak classifier, we obtain H_m such that:

$$\nabla L_m(\beta(H_m), \alpha(H_m)) \parallel (\alpha_0/\beta_0, 1)^\mathrm{T}. \tag{18}$$

Because Q_m is convex, $(\beta(H_m), \alpha(H_m))$ is a unique point.

Since $(\alpha_0/\beta_0, 1)^{\mathrm{T}} \parallel (\alpha_0, \beta_0)^{\mathrm{T}}$, these two points must coincide:

$$(\beta(H_m), \alpha(H_m)) = (\beta^*, \alpha^*). \tag{19}$$

$\square$

Theorem 3.1 states that even though the next ROC curve is unknown, if it intersects with triangle OAB, then by using goal $\lambda^* = \alpha_0/\beta_0$, a stopping point is found in no more than one single iteration. This is an absolute guarantee. Other choices of λ may result in one single iteration, but the guarantee is not absolute.

Furthermore, if the ROC curve does not intersect with triangle OAB, we still have a guarantee that the newly found operating point minimizes function G_{λ^*}. Since $\nabla G_{\lambda^*} \parallel (\alpha_0, \beta_0)^{\mathrm{T}}$, it means the newly found operating point is the one nearest to the line containing segment AB in Euclidean distance.

Therefore, we use $\lambda = \lambda^*$ to train every weak classifier of a boosted classifier. In theory, the line containing segment AB is $\lambda^*\beta + \alpha = \alpha_0$, so we can use $J_{m,\lambda^*}(h_m) \leq \alpha_0$ as the stopping condition. In practice, we use the original stopping condition: $\alpha(H_m) \leq \alpha_0$ and $\beta(H_m) \leq \beta_0$.

Experiments in section 4.2 verify our analysis.

3.3. Updating weights

An important advantage in our method is that every weak classifier is trained with the same asymmetric goal as the boosted classifier. Therefore, the number of weak classifiers needed to reach the desired rates is greedily the shortest. In addition, no threshold adjustment need be applied to a boosted classifier as in many previous methods. Conversely, the use of a threshold would mean that the asymmetric goal of the boosted classifier is different from that used in training the weak classifiers; hence the weak classifiers would have been sub-optimally trained *w.r.t.* the goal.

We use the same weight-updating rule proposed by Viola and Jones [13]. By simply multiplying the weights of every example $(\mathbf{x}_n, y_n)$ with $\sqrt{\lambda^{y_n}}$, before training the first weak classifier, the training goal of every weak classifier essentially becomes asymmetric: $FAR + \lambda FRR$. Notice that for every weak classifier, λ and b_m are related by $b_m = 0.5 \log \lambda$ as in (10).

4. Experimental Results

4.1. Setup

To justify the arguments we made in the previous section, we ran a few experiments. We collected 1521 face images from the BioID Face Database[3], 508 face images from the AR Face Database [6], and about 4000 frontal face images used by Xiao *et al.* in [16]. Altogether we obtained about 6000 face images. We used a generator that selects known face locations randomly, and resizes them down to a base resolution of 24×24 pixels, with some perturbation bias to achieve a slightly more robust performance [11]. We used the same collection of a few thousand large images containing no faces used by Wu *et al.* in [15] to generate sufficient non-face sub-windows.

[3]http://www.bioid.com/downloads/facedb/index.php

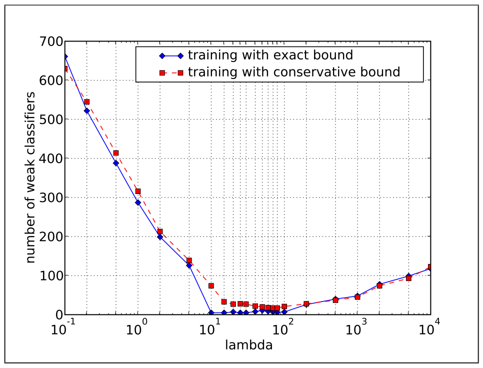


Figure 2. Comparison of training a boosted classifier with different λ and stopping bounds

In order to speed up our comparisons, we implemented the technique for fast-training weak classifiers described in [7]. Nineteen types of Haar-like features were used, generating nearly 300,000 Haar features. We observed a training time of 4 seconds per weak classifier. This technique was uniformly applied to all methods compared in table 1.

For testing, we used the MIT+CMU test set which consists of 130 grayscale images with 507 frontal faces [9]. Post-processing was the same as in [12]. The experiments were done on a 3.2GHz Intel Pentium IV PC with 1 GB memory running Windows XP.

4.2. Training with different asymmetric goals

To test the effects between choosing different λ for training and the resulting number of weak classifiers, we ran the following experiment. We trained multiple single-entrance single-exit boosted classifiers by varying only λ and the stopping bound (*i.e.*, either exact or conservative). All of these were trained using the same face training set with 5,000 examples per class, which had been filtered from a cascade of a few layers. This was to avoid a trivial case that a boosted classifier might end up using too few weak classifiers. We set the desired rates to be: $\alpha_0 = 0.8$ for false acceptance rate, and $\beta_0 = 0.01$ for false rejection rate. Hence the choice of λ selected by our method was at $\lambda^* = 80$. We counted the numbers of weak classifiers obtained when the algorithms stopped. The results are plotted in figure 2.

From figure 2, the resultant curves are similar for $\lambda \leq 5$ or $\lambda \geq 100$. They differ at the range $5 \leq \lambda \leq 100$, where they both have low numbers of weak classifiers. Figure 3 illustrates an expanded view of this smaller range.

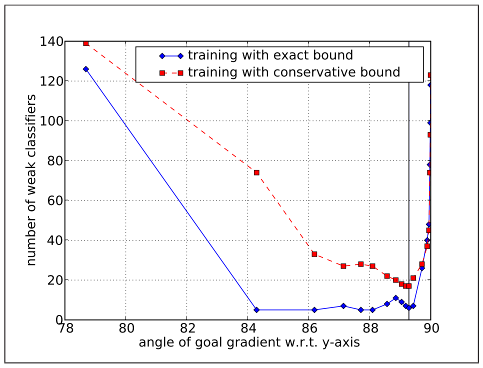


Figure 3. A closer view on the comparison of training a boosted classifier with different λ and bounds

In figure 3, the x-axis represents the angle of the gradient of the goal function G_λ, *w.r.t.* y-axis. The angle is calculated as $\tan^{-1}\lambda$. As $\lambda \to 0$, the angle approaches $0°$, while the angle approaches $90°$ as $\lambda \to \infty$. The figure shows that most of the values of λ with few weak classifiers correspond to angles above 80 degrees. When training with the conservative bound, the number of weak classifiers at $\lambda = \lambda^*$ (indicated by the vertical line at $\tan^{-1} 80 = 89.3°$) was observed to be empirically optimal. We saw some fluctuations when training with the exact bound. However, the observed number of weak classifiers at $\lambda = \lambda^*$ was very close to that of an optimal one.

4.3. Performance comparisons

We compared our method with original cascade [12] and asymmetric cascade [13] of Viola and Jones, boosting chain of Xiao *et al.* [17], nesting-structured cascade of Wu *et al.* [14], soft cascade of Bourdev *et al.* [1], and dynamic cascade of Xiao *et al.* [16]. To obtain free parameters for these methods, we used either their proposed best parameters, or obtained manually by trial and error. Also to make the comparisons as reasonable as possible, we simplified their versions by removing ideas that are reasonably irrelevant to the comparisons. For example, in nesting-structured cascade, we replaced Real AdaBoost with Discrete AdaBoost. We used the same type of weak classifiers for all methods.

Table 1 shows some statistics obtained when training these methods. We report the total training time based on the best boosted classifiers. The first four methods had many free parameters thus requiring much more trial and error to get the best results than the last three.

The table showed that the methods that propagate scores (*i.e.*, boosting chain, nesting-structured cascade, dynamic cascade, and multi-exit asymmetric boosting) resulted in significantly smaller numbers of weak classifiers. Among these four methods, multi-exit asymmetric boosting gave the smallest number of weak classifiers, less than two third that of the second best, *i.e.*, nesting-structured cascade.

Method	No of weak classifiers	No of exits	Total training time
Viola-Jones [12]	4,297	32	6h20m
Asym cascade [13]	3,502	29	4h30m
Boosting chain [17]	959	22	2h10m
Nested cascade [14]	894	20	2h
Soft cascade [1]	4,871	4,871	6h40m
Dynamic cascade [16]	1,172	1,172	2h50m
Multi-exit boosting	**575**	**24**	**1h20m**

Table 1. Some key factors after training recent methods.

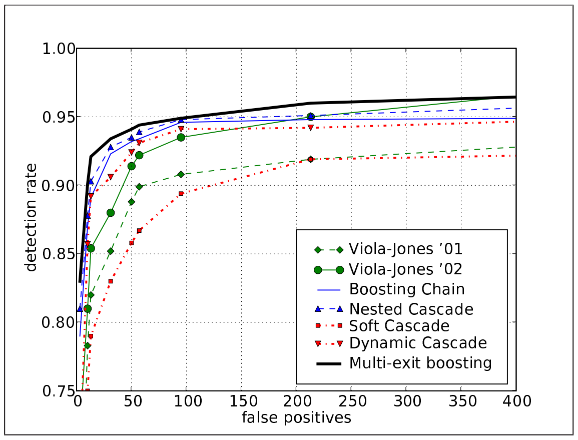


Figure 4. Comparing different methods on the MIT+CMU test set

Figure 4 shows different ROC curves of these methods. There are a few interesting points that can be understood from these ROC curves.

Soft cascade turned out poorer than Viola and Jones' original cascade. This is explainable as the former was "calibrated" from a fixed and very long boosted classifier, while in the latter, bootstrapping was used after every successful training of a boosted classifier, implying the use of better datasets. Xiao *et al.* [16] observed a similar result.

Viola and Jones' asymmetric cascade performed significantly better than both soft cascade and Viola and Jones' original cascade, but not as good as other methods. It performed better than dynamic cascade, boosting chain, and nesting-structured cascade at first, but became inferior when a lot more weak classifiers were trained. In the first few layers, training with an asymmetric goal indeed had an advantage over that with the symmetric goal. As the sub-problem became much harder as in the final layers of a cascade, the use of propagating the scores from previous boosted classifier became very useful.

Dynamic cascade performed worse than nesting-structured cascade and boosting chain. While dynamic cascade was trained automatically, boosting chain and nesting structured cascade was trained with key parameters manually selected. This result justified our argument in section 2.1: as the sub-problem becomes harder, it is better to train

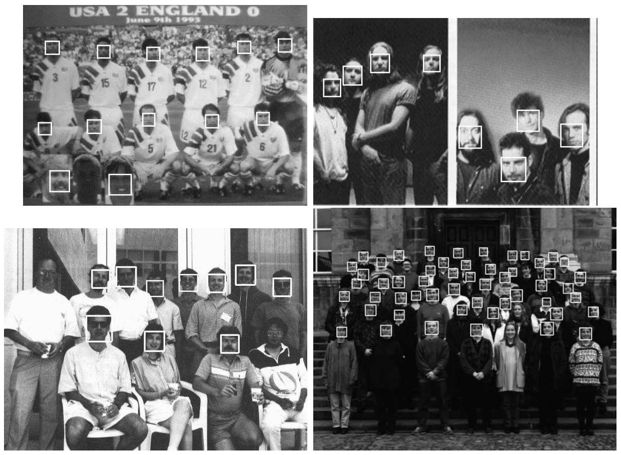


Figure 5. Some results of our method on the MIT+CMU test set

longer but more accurate boosted classifiers than making early but less accurate rejection predictions.

Nesting-structured cascade performed comparable to boosting chain. However, multi-exit asymmetric boosting performed better than all previous methods. This was expected because not only multi-exit asymmetric boosting inherited the idea of asymmetric goals from asymmetric cascade, it also inherited the idea of propagating scores from one segment to another. Besides, the values of λ selected by our method often resulted in low numbers of weak classifiers and high accuracy, and were often similar to those manually chosen for asymmetric cascade [13]. Multi-exit asymmetric boosting avoided the need to specify the number of weak classifiers to train per segment, as well as to train a segment multiple times.

5. Conclusions and Future Work

We proposed a method to train a multi-exit boosted classifier by combining the idea of propagating scores in [14, 17] and training with an asymmetric goal in [13]. We showed how to properly select an goal that achieves the desired error rates with a minimum number of weak classifiers, avoiding the need to run multiple ad hoc trials to discover the best boosted classifiers that satisfy the operating point requirements. Experimental results showed not only significant reduction in training time and number of weak classifiers, but also better accuracy compared to conventional cascades and multi-exit boosted classifiers.

We did not address the problem of selecting the desired rates for all the exit nodes, leaving it as an open problem for future work. In the paper, we used fixed desired rates for every exit node, similar to that of Viola and Jones [12, 13]. Also, our analysis of asymmetric goals might be the first, but clearly future work should provide a deeper investigation into the issue.

Acknowledgments

This work was carried out in the Centre for Multimedia and Network Technology (CeMNet), NTU. We would like to thank Rong Xiao and Jianxin Wu for having graciously shared their databases, as well as Minhtuan Pham, ThanhGiang T. Vo and ThanhVu H. Nguyen for their support on this work.

References

[1] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, volume 2, pages 236–243 vol. 2, 2005.

[2] S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Towards optimal training of cascaded detectors. In *ECCV*, pages 325–337, 2006.

[3] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[4] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, pages 67–81, 2002.

[5] H. Luo. Optimization design of cascaded classifiers. In *CVPR*, volume 1, pages 480–485 vol. 1, 2005.

[6] A. Martinez and R. Benavente. The ar face database. Technical Report 24, CVC, U.A.B., June 1998.

[7] M.-T. Pham and T.-J. Cham. Fast training and selection of haar features using statistisc in boosting-based face detection. In *ICCV*, 2007.

[8] M.-T. Pham and T.-J. Cham. Online learning asymmetric boosted classifiers for object detection. In *CVPR*, pages 1–8, 2007.

[9] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Proc. CVPR*, June 1998.

[10] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[11] J. Sun, J. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. In *CVPR*, volume 2, pages II–276–II–283 Vol.2, 2004.

[12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001.

[13] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *NIPS 14*. MIT Press, Cambridge, MA, 2002.

[14] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *FGR*, pages 79–84, 17-19 May 2004.

[15] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Trans. PAMI*, 20(3):369–382, 2008.

[16] R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *ICCV*, 2007.

[17] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *ICCV*, pages 709–715vol.1, 2003.