Fast and Robust Object Detection Using Asymmetric Totally-corrective Boosting

Peng Wang, Chunhua Shen, Nick Barnes, and Hong Zheng

Abstract

Boosting based object detection has received significant attention recently. In this work, we propose totally-corrective asymmetric boosting algorithms for real-time object detection. Our algorithms differ from Viola-Jones' detection framework in two folds. Firstly, our boosting algorithms explicitly optimize asymmetric loss of objectives, while AdaBoost used by Viola and Jones optimizes a symmetric loss. Secondly, by carefully deriving the Lagrange duals of the optimization problems, we design more efficient boosting in that the coefficients of the selected weak classifiers are updated in a totally-corrective fashion, in contrast to the stage-wise optimization commonly used by most boosting algorithms. Column generation is employed to solve the proposed optimization problems. Unlike conventional boosting, the proposed boosting algorithms are able to de-select *those irrelevant weak classifiers in the ensemble while training a classification cascade. This results in improved detection performance as well as fewer weak classifiers in the learned strong classifier. Compared with AsymBoost of Viola and Jones [1], our proposed asymmetric boosting is non-heuristic and the training procedure is much simpler. Experiments on face and pedestrian detection demonstrate that our methods have superior detection performance than some of the state-of-the-art object detectors.*

Key Words—Object detection, asymmetric learning, AdaBoost, totally-corrective boosting, column generation.

I. Introduction

REAL-TIME object detection has broad applications in image analysis, computer vision and multimedia processing [2]–[5]. Despite extensive effort spent on this topic, it remains a challenging task. An object detector should be able to efficiently locate targets, which are usually only a few, from millions of sub-windows in a target image. It is a typical highly-imbalanced learning problem. The strict requirement on detection accuracy and speed makes it sub-optimal to directly apply those methods designed for standard learning problems.

The boosting cascade classification framework originally proposed by Viola and Jones [3] has made a great success for real-time object detection and arisen much extended work. There are three significant contributions in the Viola-Jones face detector: Haar features, the cascade classifier and AdaBoost. Haar features can be calculated extremely efficiently by using the concept of integral images. The cascade classifier is employed to partially address the imbalanced nature of the training data and reduce the computation time in the test phase. The target can be considered as a rare event among all the possible sub-windows in an image. Therefore, there are many more non-target examples than target examples. The distribution of those negative non-target data is also much more complicated than the positive data. A detector must have a very high detection rate (*e.g.*, 95%) and very low false positive rate (*e.g.*, 10^{-5}). It would be very difficult for a single classifier to achieve this requirement. The cascade structure is usually adopted, which consists of a sequence of node classifiers (see Fig. 1). Only those instances passing through the current node will be evaluated by the next one; and only those instances passing through all the nodes are classified as true detections. In practice, a majority of sub-windows are rejected in the first several nodes, which leads to a significant reduction of computation time. On the other hand, the learning goal of each node is much easier than the final learning goal. In each node, we want to train a classifier with a very high detection rate (*e.g.*, 99.5%) and a moderate false positive rate (*e.g.*, around 50%). AdaBoost is employed for learning node classifiers in Viola-Jones' method [3]. In theory, any other boosting algorithms can be used here, for example, [2], [6]–[8]. As weak classifiers are decision stumps, which are trained on a single Haar feature out of millions of possible Haar features, the process of training AdaBoost based detectors is also a procedure of feature selection.

Although the cascade classifier partially addresses the asymmetry of learning by splitting the detection process into multiple nodes, the learning goal for an individual node classifier is still asymmetric. A drawback of AdaBoost in the context of training a cascade classifier is that AdaBoost is designed to minimize a symmetric loss. This makes it unable to build

P. Wang and H. Zheng are with Beihang University, Beijing, China, 100161. P. Wang's contribution was made when visiting NICTA, Canberra Research Laboratory.

C. Shen is with The Australian Center for Visual Technologies, The University of Adelaide, SA 5005, Australia (e-mail: chunhua.shen@adelaide.edu.au). Correspondence should be addressed to C. Shen.

N. Barnes is with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia.

Appearing in IEEE Transaction on Neural Networks. This reprint differs from the original in pagination and typographic detail.

an *optimal* cascade classifier, considering that the asymmetric learning goal is not systematically taken into account.

Much subsequent work attempts to improve the performance of object detectors by introducing asymmetric learning strategies into boosting algorithms. Viola and Jones later proposed asymmetric AdaBoost (AsymBoost) [1], which applies an asymmetric multiplier parameter to one of the two classes. However, this asymmetry could be absorbed *immediately* by the first weak classifier because of AdaBoost's greedy optimization strategy. In practice, they have gradually applied the n-th root of the multiplier at each iteration in order to keep the asymmetric effect throughout the entire training process. Here n is the number of weak classifiers. *This heuristic cannot guarantee the solution to be optimal and the number of weak classifiers has to be specified before training.* Therefore, carefulness is needed in implementation. In contrast, as we will show in this work, our proposed algorithms are non-heuristic and the training procedure is much simpler.

AdaCost presented by Fan *et al.* [9] adds a cost adjustment function on the weight updating strategy of AdaBoost. They also pointed out that the weight updating rule should consider the cost not only on the initial weights but also at each iteration. Hou *et al.* [10] used varying asymmetric factors for training different weak classifiers. However, because the asymmetric factor changes during training, it remains unclear what objective function is optimized. Li and Zhang [2] proposed FloatBoost to reduce the redundancy of greedy search by incorporating floating search into AdaBoost. In FloatBoost, the poor weak classifiers are removed during the training. Wu *et al.* [11] observed that feature selection and ensemble classifier learning can be decoupled. They designed a linear asymmetric classifier (LAC) to adjust the linear coefficients of the selected weak classifiers.

On the other hand, research has also been devoted to optimize the cascade structure. Luo [12] proposed post-processing methods, which jointly adjust the thresholds of all nodes within the cascade. Brubaker *et al*. [13] developed a principled strategy to set the node thresholds and decide when to stop training a node. Xiao *et al.* [14] improved the backtrack technique in [2] and exploited the historical information of preceding nodes into successive node learning. Bourdev and Brandt [15] presented a new cascade structure, termed the soft cascade. In the soft cascade, each weak classifier is a node and the score of an instance is accumulated. Raykar *et al*. [16] presented a method which jointly train all weak classifiers in the cascade. Pham *et al.* [17] presented a method that trains the asymmetric AdaBoost [1] classifiers under a new cascade structure, the multi-exit cascade. Like the soft cascade [15], boosting chain [14] and dynamic cascade [18], the multi-exit cascade is a cascade structure which takes the historical information into consideration. In a multi-exit cascade, the t-th node "inherits" weak classifiers selected by the preceding $t-1$ nodes.

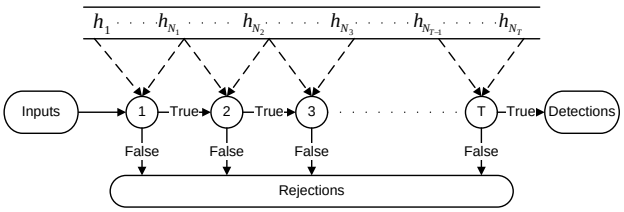
Most of the previous work is based on AdaBoost and achieves the asymmetric learning goal by *heuristic* weights manipulations or post-processing techniques. It is not trivial to assess how these heuristics affect the original loss function of AdaBoost. In this work, we design new boosting algorithms *directly* from asymmetric loss objectives. The optimization process is implemented by column generation. Experiments on toy data and real data show that our algorithms indeed achieve the asymmetric learning goal without any heuristic manipulation, and outperform previous methods. Part of this work appeared in [19].

In summary, the main contributions of this work are as follows.

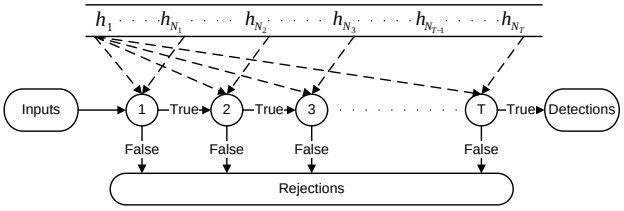
1) We explicitly optimize asymmetric losses in our proposed algorithms. In the stage-wise asymmetric boosting such as Viola and Jones' AsymBoost [1], the first weak classifier intends to absorb all the asymmetric cost and the subsequent weak classifiers are learned symmetrically. Heuristics are employed to prevent this drawback. In contrast, there is no sub-optimum-pruned heuristic strategy in our algorithms due to our convex optimization formulation of boosting. The asymmetric learning goal is introduced into both feature selection and ensemble classifier learning. Both the example weights and the linear classifier coefficients are learned in an asymmetric way.
2) To our knowledge, for the first time, our proposed approaches introduce asymmetric losses into totally-corrective boosting algorithms. This is also the first time that totally-corrective boosting algorithms are applied for cascade classification and object detection tasks. Improved results are achieved on face detection and pedestrian detection.
3) We demonstrate that with the totally-corrective optimization and the ℓ_1-norm regularization, the linear coefficients of some weak classifiers are automatically set to zero by the algorithm such that fewer weak classifiers are needed in the final strong/cascade classifier. We present analysis on the theoretical condition and show how useful the historical information is for the training of successive nodes.
4) A fast version of our algorithms is also proposed. At each iteration, multiple weak classifiers are produced and added into the strong classifier. The fast algorithms also present promising results on object detection tasks.

Before we present the main results, we introduce the notation that is used in this paper.

Suppose that there are M training examples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, and the first M_1 are positive examples and the last M_2 are negatives. A pool $\mathcal{H}$ consists of N available weak classifiers. The matrix $H \in \mathbb{Z}^{M \times N}$ consists of binary outputs of weak classifiers in $\mathcal{H}$ over training examples, namely $H_{ij} = h_j(\boldsymbol{x}_i)$. H_i denotes the i-th row of H, which consists of the binary outputs of all weak classifiers on the i-th example. We are



(a) Viola-Jones cascade structure



(b) The multi-exit cascade structure

Fig. 1: A description of two cascade structures. In (a), the t-th node of Viola-Jones cascade use weak classifiers from index $N_{t-1}+1$ to N_t; For the multi-exit cascade in (b), each node shares weak classifiers with its anterior.

aiming to learn a linear ensemble classifier

$$F_{\boldsymbol{w}}(\cdot) = \textstyle\sum_{j=1}^{N} w_i h_j(\cdot),$$

where $w_1, w_2, \ldots, w_N$ are the coefficients of the linear ensemble classifier. In our boosting algorithms, we use $u_1, u_2, \ldots, u_M$ to denote the weights of training examples. The *edge* of a weak classifier h_j is defined as $\sum_{i=1}^{M} u_i y_i h_j(x_i)$, which can be seen as the inverse of the weighted error over training examples.

II. Asymmetric losses

In this section, we introduce two asymmetric losses, which are motivated by AsymBoost [1] and cost-sensitive LogitBoost [20], respectively.

We first introduce an asymmetric cost in the following form:

$$\text{Cost} = \begin{cases} C_1 & \text{if } y = +1 \text{ and } \operatorname{sgn}[F(\boldsymbol{x})] = -1, \\ C_2 & \text{if } y = -1 \text{ and } \operatorname{sgn}[F(\boldsymbol{x})] = +1, \\ 0 & \text{if } y = \operatorname{sgn}[F(\boldsymbol{x})]. \end{cases}$$

Here $\boldsymbol{x}$ is the input data, y is the label and $F(\boldsymbol{x})$ is the learned classifier. Viola and Jones [1] directly employ the product of this cost and the exponential loss function as the asymmetric loss:

$$\mathbb{E}_{\boldsymbol{X},Y}\left[\left(\frac{1+y}{2}C_1 + \frac{1-y}{2}C_2\right)\exp\left(-yF(\boldsymbol{x})\right)\right].$$

In a similar manner, we can also write an asymmetric loss with respect to the logistic loss function as follows:

$$\text{Loss}_1 = \mathbb{E}_{\boldsymbol{X},Y}\left[\left(\frac{1+y}{2}C_1 + \frac{1-y}{2}C_2\right)\text{logit}\big(yF(\boldsymbol{x})\big)\right], \quad (1)$$

where $\text{logit}(x) = \log(1+\exp(-x))$ is the logistic loss function.

Masnadi-Shirazi and Vasconcelos [20] proposed cost-sensitive boosting algorithms which optimize different versions of cost-sensitive losses by means of gradient descent. They proved that the optimal cost-sensitive predictor minimizes the following expected loss:

$$-\mathbb{E}_{\boldsymbol{X},Y}\left[\frac{1+y}{2}\log(\mathrm{p_c}(\boldsymbol{x})) + \frac{1-y}{2}\log(1-\mathrm{p_c}(\boldsymbol{x}))\right],$$

where

$$\mathrm{p_c}(\boldsymbol{x}) = \frac{\exp(\gamma F(x)+\eta)}{\exp(\gamma F(x)+\eta)+\exp(-\gamma F(x)-\eta)}, \quad (2)$$

with $\gamma = \frac{C_1+C_2}{2}$ and $\eta = \frac{1}{2}\log\frac{C_2}{C_1}$.

When fixing γ to 1, the expected loss can be reformulated as

$$\text{Loss}_2 = \mathbb{E}_{\boldsymbol{X},Y}\left[\text{logit}\big(yF(\boldsymbol{x}) + 2y\eta\big)\right]. \quad (3)$$

Next we show how to design totally-corrective boosting for optimizing objective functions that are directly related with the above asymmetric losses.

III. Asymmetric totally-corrective boosting algorithms

The concept of totally-corrective is firstly introduced by Kivinen and Warmuth [21]. In stage-wise boosting like AdaBoost, only the coefficient of the *latest* weak classifier is updated at each iteration. We refer to this type of algorithms as *stage-wise* boosting algorithms. In contrast, totally-corrective algorithms update example weights with constraints that the new weights should be orthogonal to mistake vectors of *all* weak classifiers. In order to make the updating problem feasible, the coefficients of *all* weak classifiers are updated at each iteration.

Recently, Warmuth *et al*. [7] described a totally-corrective algorithm from a different viewpoint. They state that AdaBoost only constrains the edge of the last weak classifier when updating example weights, while in "totally-corrective" algorithms the edges of all weak classifiers are constrained by a maximum value θ:

$$\textstyle\sum_{i=1}^{M} u_i y_i h_j(x_i) < \theta, \quad j = 1, 2, \ldots, N. \quad (4)$$

To satisfy all the constraints, the linear coefficient of all weak classifiers are updated as well. Warmuth *et al*. [7] did not derive the totally-corrective boosting from the optimization problems that a boosting algorithm actually optimizes. Instead, their TotalBoost is obtained by only considering the dual problems of boosting algorithms. Shen and Li [22], [23]

explicitly established the primal and dual relationship for a few existing boosting algorithms. Based on the Lagrange dual problems, column generation is then used to design totally-corrective boosting algorithms. Their work can be seen as an extension of the LPBoost [24] to more general convex loss functions. To our knowledge, little work has been reported on designing *asymmetric* totally-corrective boosting.

A. Asymmetric totally-corrective boosting

In this section, we construct asymmetric totally-corrective boosting algorithms (termed AsymBoost$_{\mathrm{TC}}$ here) from the losses (1) and (3) discussed previously. Compared with the methods for constructing boosting-like algorithms in [20], [25] and [26], we use column generation to design our totally corrective boosting algorithms, mainly inspired by [24] and [22].

The constants C_1 and C_2 are costs for misclassifying positive and negative datum points, respectively. We assign the asymmetric factor $k = C_1/C_2$ and restrict $\gamma = (C_1 + C_2)/2$ to 1, thus C_1 and C_2 are fixed for a given k.

By putting the learning into ℓ_1 regularization framework as discussed in [22], the asymmetric learning problems of the two AsymBoost$_{\mathrm{TC}}$ algorithms can be expressed as:

$$\min_{\boldsymbol{w}} \sum_{i=1}^{M} l_i \mathrm{logit}(z_i) + \theta \mathbf{1}^\top \boldsymbol{w}, \ \text{s.t.}\, \boldsymbol{w} \succcurlyeq \mathbf{0}, \ z_i = y_i H_i \boldsymbol{w}, \quad (5)$$

where $\boldsymbol{l} = [C_1/M_1, \cdots, C_2/M_2, \cdots]^\top$, and

$$\min_{\boldsymbol{w}} \sum_{i=1}^{M} e_i \mathrm{logit}(z_i + 2y_i\eta) + \theta \mathbf{1}^\top \boldsymbol{w}, \text{s.t.}\, \boldsymbol{w} \succcurlyeq \mathbf{0}, \ z_i = y_i H_i \boldsymbol{w}, \quad (6)$$

where $\boldsymbol{e} = [1/M_1, \cdots, 1/M_2, \cdots]^\top$. In both (5) and (6), z_i denotes the margin of the training example $\boldsymbol{x}_i$. Note that the introduction of this auxiliary variable $\boldsymbol{z}$ is important to arrive at the dual problems that we need, although the variable $\boldsymbol{z}$ is not of interest. We refer to (5) as AsymBoost$_{\mathrm{TC1}}$ and (6) as AsymBoost$_{\mathrm{TC2}}$. Note that here the optimization problems are ℓ_1-norm regularized. It is possible to use other format of regularization such as the ℓ_2-norm.

First, we introduce a fact that the Fenchel conjugate [27] of the logistic loss function $\mathrm{logit}(x)$ is $\mathrm{logit}^*(u) = (-u)\log(-u) + (1+u)\log(1+u)$, for $0 \geq u \geq -1$; and ∞, otherwise.

Now let us derive the Lagrange dual [27] of AsymBoost$_{\mathrm{TC1}}$. The Lagrangian of (5) can be written as

$$L(\underbrace{\boldsymbol{w}, \boldsymbol{z}}_{\text{primal}}, \underbrace{\boldsymbol{\lambda}, \boldsymbol{u}}_{\text{dual}}) = \sum_{i=1}^{M} l_i \mathrm{logit}(z_i) + \theta \mathbf{1}^\top \boldsymbol{w} - \boldsymbol{\lambda}^\top \boldsymbol{w} + \sum_{i=1}^{M} u_i (z_i - y_i H_i \boldsymbol{w}). \quad (7)$$

The Lagrangian function is

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{u}) &= \inf_{\boldsymbol{w}, \boldsymbol{z}} L(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{u}) \\ &= -\sum_{i=1}^{M} \underbrace{\sup_{z_i} \Big(-u_i z_i - l_i \mathrm{logit}(z_i) \Big)}_{l_i \mathrm{logit}^*(-u_i/l_i)} \\ &\quad + \inf_{\boldsymbol{w}} \underbrace{\Big(\theta \mathbf{1}^\top - \boldsymbol{\lambda}^\top - \sum_{i=1}^{M} u_i y_i H_i \Big)}_{\text{must be } \mathbf{0}} \boldsymbol{w}. \end{aligned} \quad (8)$$

So we can write the Lagrange dual problem as

$$\begin{aligned} \max_{\boldsymbol{u}} \ & -\sum_{i=1}^{M} \Big[u_i \log(u_i) + (l_i - u_i)\log(l_i - u_i) \Big] \\ \text{s.t.} \ & \sum_{i=1}^{M} u_i y_i H_i \preccurlyeq \theta \mathbf{1}^\top, \ 0 \preccurlyeq \boldsymbol{u} \preccurlyeq \boldsymbol{l}. \end{aligned} \quad (9)$$

Since the problem (5) is convex and the Slater's conditions are satisfied [27], the duality gap between the primal (5) and the dual (9) is zero. Therefore, the solutions of (5) and (9) are the same. Through the KKT condition, the gradient of Lagrangian (7) over primal variable $\boldsymbol{z}$ and dual variable $\boldsymbol{u}$ must vanish at the optimum. Therefore, we can easily obtain the relationship between the optimal value of $\boldsymbol{z}$ and $\boldsymbol{u}$:

$$u_i^* = \frac{l_i \exp(-z_i^*)}{1 + \exp(-z_i^*)}. \quad (10)$$

Similarly, we can also find the dual problem of AsymBoost$_{\mathrm{TC2}}$, which can be expressed as:

$$\begin{aligned} \max_{\boldsymbol{u}} & -\sum_{i=1}^{M} \Big[u_i \log(u_i) + (e_i - u_i)\log(e_i - u_i) + 2u_i y_i \eta \Big] \\ \text{s.t.} \ & \sum_{i=1}^{M} u_i y_i H_i \preccurlyeq \theta \mathbf{1}^\top, \ 0 \preccurlyeq \boldsymbol{u} \preccurlyeq \boldsymbol{e}, \end{aligned} \quad (11)$$

with

$$u_i^* = \frac{e_i \exp(-z_i^* - 2y_i\eta)}{1 + \exp(-z_i^* - 2y_i\eta)}. \quad (12)$$

From (9) and (11), we can find that, the first constraints of both the problems are identical to the constraints of TotalBoost [7], *i.e*. Equ. (4). In practice, the total number of weak classifiers, N, could be extremely large, so we may not be able to solve the primal problems (5) and (6) directly. However, equivalently, we can optimize the duals (9) and (11) iteratively using column generation [24].

Column generation is a technique originally used for large-scale linear programming problems. Demiriz *et al*. [24] used this method to design boosting algorithms. Column generation is based on the assumption that the solution is sparse, hence only a subset of variables need to be considered. At each iteration, one column—a variable in the primal or a constraint

in the dual—is added when solving the restricted problem. Till one can not find any column violating the constraint in the dual, the solution of the restricted problem is identical to the optimal solution. From the viewpoint of boosting algorithms, usually there are extremely large (even infinitely many) weak classifiers available. However, only a small number of weak classifiers are helpful for classification and should be selected into the ensemble classifier. Therefore, in the proposed algorithms, there are a huge number of primal variables (or equivalently dual constraints). Here each primal variable or dual constraint corresponds to one weak classifier. Based on column generation, we can add one weak classifier into the primal or the dual and solve the problem iteratively, until no violated constraint can be found.

To speed up the convergence, we add the most violated constraint in each round by finding a weak classifier satisfying:

$$h'(\cdot) = \underset{h(\cdot)}{\operatorname{argmax}} \sum_{i=1}^{M} u_i y_i h(\boldsymbol{x}_i). \tag{13}$$

Solving this subproblem is the same as training a weak classifier in AdaBoost or LPBoost, in which one tries to find a weak classifier with the maximum edge (*i.e.*, the minimum weighted error). Then we solve the restricted dual problem with one more constraint than the previous round, and update the linear coefficients of weak classifiers ($\boldsymbol{w}$) and the weights of training examples ($\boldsymbol{u}$). Clearly, adding one constraint into the dual problem corresponds to adding one variable into the primal problem. Since the primal problem and dual problem are equivalent, we may either solve the restricted dual or the restricted primal in practice. The algorithms of AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ are summarized in Algorithm 1.

Note that, in the context of training a boosting based object detector, in order to achieve the specific false negative rate (FNR) or false positive rate (FPR), an offset b is needed to be added into the final strong classifier: $F(\boldsymbol{x}) = \sum_{j=1}^{n} w_i h_j(\mathbf{x}) - b$, which can be obtained by a simple line search. The new weak classifier $h'(\cdot)$ corresponds to an extra variable to the primal and an extra constraint to the dual. Thus, the minimal value of the primal decreases with growing variables, and the maximal value of the dual problem also decreases with growing constraints. More formally, we want to show the following theorem.

Theorem 1. *Algorithm 1 makes progress (decreases the objective value) at each iteration and hence in the limit it solves the problem* (5) *or* (6) *globally to a desired accuracy.*

Proof: We consider the case of (5). For the problem (6), the proof follows the same discussion. Suppose that the current solution is a finite subset of weak classifiers and their corresponding linear weights are $\boldsymbol{w} = [w_1, \ldots, w_N]$ If we add a weak classifier $\hat{h}(\cdot)$ that is not in the current subset, the corresponding $\hat{w}$ is zero (which means that $\hat{h}(\cdot)$ is useless), then we can conclude that the current weak classifiers and $\boldsymbol{w}$ are the optimal solution.

Consider that this optimality condition is violated. We want to show that we can find a weak classifier $\hat{h}(\cdot)$ not being in the current selected weak classifiers, such that $\hat{w} > 0$ holds. Let us assume that $\hat{h}(\cdot)$ is a weak classifier found by solving the subproblem (13) and the convergence condition $\sum_{i=1}^{M} u_i y_i \hat{h}(\mathbf{x}_i) \leq \theta$ is not satisfied. In other words, $\sum_{i=1}^{M} u_i y_i \hat{h}(\mathbf{x}_i) > \theta$.

If after this weak classifier is added into the primal problem and the primal solution remains unchanged, *i.e.*, the corresponding $\hat{w} = 0$. From the optimality condition $\hat{\lambda} = \theta - \sum_{i=1}^{M} u_i y_i \hat{h}(\mathbf{x}_i) < 0$, which contradicts the fact $\hat{\lambda} \geq 0$.

Therefore, after the weak classifier $\hat{h}(\cdot)$ is added to the primal problem, its corresponding $\hat{w}$ must have a positive solution. It means that one more free variable is added into the problem and re-solving the primal problem (5) will reduce the objective value. Hence, a strict decrease in the objective is obtained, and Algorithm 1 makes progress at each iteration.

On the other hand, as the optimization problems involved are convex, there are no local optimal points. Hence, Algorithm 1 is guaranteed to converge to the global optimum. ■

The above analysis establishes the convergence of Algorithm 1 but it does not tell us anything about the convergence rate.

Next, we try to understand how AsymBoost$_{\mathrm{TC}}$ introduces the asymmetric learning into feature selection and ensemble classifier learning. Decision stumps are the most commonly used type of weak classifiers, and each stump only uses one dimension of the features. So the process of training weak classifiers (decision stumps) is also a procedure of selecting relevant features. In our framework, the weak classifier with the maximal edge (*i.e.*, the minimal weighted error) is selected. From (10) and (12), the weight of i-th example, namely u_i, is affected by two factors: the asymmetric factor k and the current margin z_i. If we set $k = 1$, the weighting strategy goes back to the symmetric boosting case. On the other hand, the coefficients of the weak classifiers, $\boldsymbol{w}$, are updated by solving the restricted primal problem at each iteration. The asymmetric factor k in the primal is absorbed by all the weak classifiers currently learned. So both feature selection and ensemble classifier learning consider the asymmetric factor k.

The number of variables in the primal problem is the number of weak classifiers; while for the dual problem, it is the number of training examples. In the cascade classifiers for object detection, the number of weak classifiers is usually much smaller than the number of training examples, so solving the primal is much cheaper than solving the dual. Since the primal problem has only simple box-bounding constraints, we can employ L-BFGS-B [28] to solve it. L-BFGS-B is a tool based on the quasi-Newton method for box-constrained optimization. Instead of maintaining the Hessian matrix, L-BFGS-B only needs the recent several updates of values and gradients of the cost function to approximate the Hessian matrix. Thus, L-BFGS-B requires less memory for running. In column generation, we can use the results from previous

Algorithm 1 AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$.

Input: A training set with M labeled examples (M_1 positives and M_2 negatives); termination tolerant $\varepsilon > 0$; regularization parameter θ; asymmetric factor k; maximum number of weak classifiers $N_{\max}$.

Initialization: $N = 0$; $\boldsymbol{w} = \mathbf{0}$; and $u_i = l_i/2$ or $e_i/(1+k^{-y_i})$, $i = 1 \cdots M$. [a]

for iteration $= 1 : N_{\max}$ **do**

- · Train a weak classifier maximizing the edge: $h'(\cdot) = \mathrm{argmax}_{h(\cdot)} \sum_{i=1}^{M} u_i y_i h(\boldsymbol{x}_i)$.
- · Check for the termination condition: **if** iteration > 1 and $\sum_{i=1}^{M} u_i y_i h'(\mathbf{x}_i) < \theta + \varepsilon$, **then** break;
- · Increment the number of weak classifiers $N = N + 1$.
- · Add $h'(\cdot)$ to the restricted master problem;
- · Solve the primal problem (5) or (6) (or the dual problem (9) or (11)) and update $\boldsymbol{u}$ and $\boldsymbol{w}$.

Output:

The final strong classifier: $F(\boldsymbol{x}) = \sum_{j=1}^{N} w_j h_j(\boldsymbol{x})$.

[a] $\boldsymbol{u}$ is initialized from Equ. (10) or (12) by setting $\boldsymbol{z} = 0$.

iteration as the starting point of current problem (warm start), which leads to further reduction in computation time.

The complementary slackness condition [27] suggests that $\lambda_j w_j = 0$. So we can get the conditions of sparseness:

$$\text{If } \lambda = \theta - \textstyle\sum_{i=1}^{M} u_i y_i H_{i,j} > 0, \text{then } w_j = 0. \qquad (14)$$

This means that, if the weak classifier $h_j(\cdot)$ is so "weak" that its edge is less than θ under the current distribution $\boldsymbol{u}$, its contribution to the ensemble classifier is "zero". From another viewpoint, the ℓ_1-norm regularization term in the primal (5) and (6), leads to a sparse result. The parameter θ controls the degree of the sparseness. The larger θ is, the sparser the result would be.

B. The multi-exit cascade

Pham *et al*. [17] introduced a generalized cascade framework, termed the multi-exit cascade (see Fig. 1(b)). The basic motivation is that previous scores can be helpful for the current node. In the multi-exit framework, the t-th node incorporates the scores of all previous nodes. Fig. 1 demonstrates the difference between Viola-Jones standard cascade and the multi-exit cascade structure.

The desirable property of the multi-exit cascade is that, each node exploits the historical information. Consequently, given the same number of weak classifiers, the multi-exit cascade can utilize more information than the Viola-Jones standard cascade. The soft cascade [15] and the dynamic cascade [18] can be viewed as special cases of the multi-exit cascade.

With a minor modification, we incorporate our asymmetric totally-corrective boosting into the multi-exit cascade. The formulation is expressed as follows:

$$F(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \sum_{j=1}^{N_t} w_{tj} h_j(\boldsymbol{x}) + \theta_t \geq 0, \forall t \in \{1, \cdots T\}; \\ -1 & \text{otherwise.} \end{cases} \qquad (15)$$

Here T is the number of nodes, and $N_1, \ldots, N_T$ is the index of weak classifiers on the exits. The same weak classifier could have different coefficients with respect to different nodes.

IV. On the fast training of AsymBoost$_{\mathrm{TC}}$

In this section, we propose a fast training method, termed AsymBoost$_{\mathrm{TC}}$-fast, which reduces the training time significantly. In object detection, as features are usually obtained by exhaustive search, the dimension is extremely high. To reduce the training complexity, common practice is to apply the weak learner on one or a small number of dimensions and then select the weak classifier with the maximal edge (the minimal weighted error) from a tractable number of candidates. In this fashion, all the other sub-optimal candidates are discarded, although they can still be useful.

Unlike stage-wise boosting, there is no restriction that only one constraint (or weak classifier) can be added at each iteration. Actually, we can add multiple violated constraints at each iteration. For example, at each iteration, we can select $q > 1$ weak classifiers from K best ones with maximal edges ($q < K$). Notice that the first q weak classifiers with maximal edges should not be selected altogether because they usually are highly correlated, especially in the context of training an object detector. We want to find a few weak classifiers with large edges and at the same time they are not very correlated (image features that are not highly overlapped).

To achieve this goal, we adopt a simple criterion for choosing a promising combination of weak classifiers. Using this simple technique, we can significantly speed up the training process. First, we introduce a pseudo-distribution p_j over training samples for the j-th weak classifier $h_j(\cdot)$:

$$p_j(\boldsymbol{x}_i) \propto \frac{\exp(y_i h_j(\boldsymbol{x}_i))}{1 + \exp(y_i h_j(\boldsymbol{x}_i))}. \qquad (16)$$

Note that other strategies may be used to make a pseudo-distribution. The output of $h_j(\boldsymbol{x}_i)$ can be binary or real valued. Then we use the Jensen-Shannon divergence (JSD) as a measure of the similarity of q distributions:

$$\mathsf{JSD}(p_1, p_2, \ldots, p_q) = \mathsf{H}\left(\sum_i \pi_i p_i\right) - \sum_i \pi_i \mathsf{H}(p_i), \qquad (17)$$

where $\pi_1, \pi_2, \ldots, \pi_q$ are the weights of distributions and $\mathsf{H}(p) = -\sum_i p(\boldsymbol{x}_i) \log p(\boldsymbol{x}_i)$ is the Shannon entropy. We simply set $\pi_1 = \pi_2 = \cdots = \pi_q = 1/q$ in our case because we do not have any prior knowledge about the weight of each pseudo-distribution. The Jensen-Shannon divergence can

Algorithm 2 AsymBoost$_{\mathrm{TC1}}$-fast and AsymBoost$_{\mathrm{TC2}}$-fast.

Input: The number of weak classifiers to be add in each round q; the candidates number K; the maximum number of weak classifiers $N_{\max}$.

Initialization: $N = 0$;

while $N < N_{\max}$ **do**

- Train weak classifiers on all dimensions or dimension groups, based on the sample weights $\boldsymbol{u}$; and choose K candidates with large edges.
- Select the candidate with the largest edge from all candidates.
- **for** iteration $= 1 : q - 1$ **do**
  - Select a weak classifier from the remaining candidates, which maximizing JSD over all currently selected weak classifiers.
- Add the selected q weak classifiers to the restricted master problem.
- Solve the primal problem (5) or (6) (or the dual problem (9) or (11)) and update u_i $(i = 1 \cdots M)$ and w_j $(j = 1 \cdots N)$.
- Increment the number of weak classifiers $N = N + q$.

Output:

The final strong classifier: $F(\boldsymbol{x}) = \sum_{j=1}^{N} w_j h_j(\boldsymbol{x})$.

be viewed as a symmetrized version of the Kullback-Leibler divergence.

We summarize the fast training method AsymBoost$_{\mathrm{TC}}$-fast in Algorithm 2. In each iteration, q weak classifiers are selected from K candidates, such that the Jensen-Shannon divergence is maximized. This ensures the selected weak classifiers are uncorrelated. Since the computation of JSD is fast and only applied on the pre-selected K candidates, the AsymBoost$_{\mathrm{TC}}$-fast is almost q times faster than AsymBoost$_{\mathrm{TC}}$.

In the following, we apply the proposed methods to the problem of face detection and pedestrian detection.

V. Experiments

A. Results on synthetic data

To demonstrate the behavior of our algorithms, we construct a 2D data set, in which the positive data follow the 2D normal distribution ($\mathcal{N}(0, 0.1\mathbf{I})$), and the negative data form a ring with uniformly distributed angles and normally distributed radius ($\mathcal{N}(1.0, 0.2)$). Here $\mathbf{I}$ is the identity matrix. In total, 2000 examples are generated (1000 positives and 1000 negatives), 50% of data for training and the other half for test. We compare AdaBoost, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ on this data set. All the training processes are stopped at 100 decision stumps. For AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$, we fix θ to 0.01, and use a group of k's $\{1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0\}$. Fig. 2 shows the results.

From Figs. $2(a)$ and (c), we can see that the larger k is, the bigger the area for positive output becomes, which means that the asymmetric LogitBoost tends to make a positive decision for the region where positive and negative data are mixed together. Another observation is that AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ have almost the same decision boundaries on this data set with same k's. Figs. $2(b)$ and (d) demonstrate the trends of false rates with the growth of asymmetric factor (k). The results of AdaBoost is considered as the baseline. For all k's, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ achieve lower false negative rates and higher false positive rates than AdaBoost. With the growth of k, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ become more aggressive to reduce the false negative rate, with the sacrifice of a higher false positive rate.

B. Face detection

We collect 9832 mirrored frontal face images and about 10115 large background images. 5000 face images and 7000 background images are used for training, and 4832 face images and 3115 background images for validation. Five basic types of Haar features are calculated on each 24×24 image, and totally generate 162336 features. Decision stumps on those 162336 features construct the pool of weak classifiers.

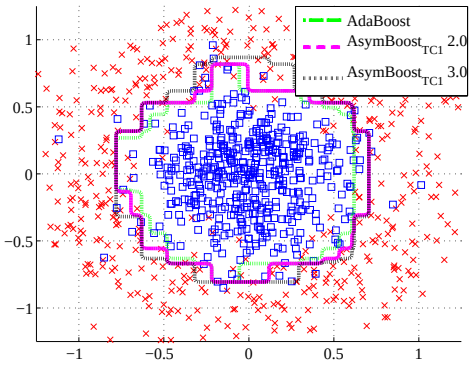
Single-node detectors Single-node classifiers with AdaBoost, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ are trained. The parameters θ and k are simply set to 0.001 and 7.0. 5000 faces and 5000 non-faces are used for training, while 4832 faces and 5000 non-faces are used for test. The training/validation non-faces are randomly cropped from training/validation background images.

Fig. $3(a)$ shows curves of detection rate with the false positive rate fixed at 0.25, while curves of false positive rates with 0.995 detection rate are shown in Fig. $3(b)$. We set the false positive rate fixed to 0.25 rather than the commonly used 0.5 in order to slow down the increasing speed of detection rates, otherwise detection rates would converge to 1.0 immediately.

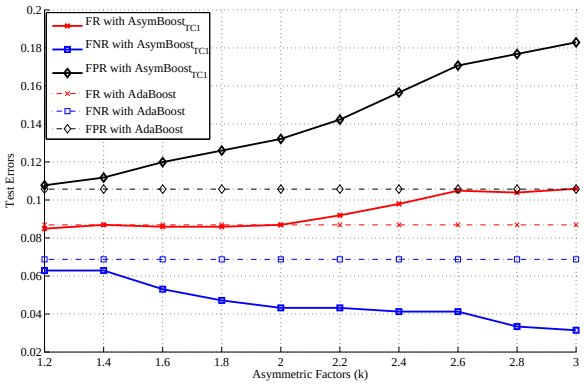
The increasing/decreasing speed of detection rate/false positive rate is faster than reported in [2] and [14]. The reason is possibly that we use 10000 examples for training and 9832 for testing, which are smaller than the data used in [2] and [14] (18000 training examples and 15000 test examples). We can see that under both situations, our algorithms convergent faster than AdaBoost.

The benefits of our algorithms can be expressed in two-fold:

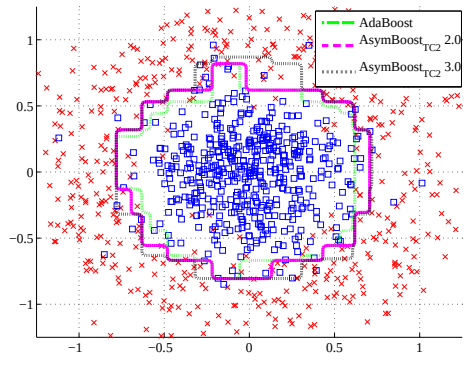
1) Given the same learning goal, our algorithms tend to use a smaller number of weak classifiers. For example, from Fig. 3 (2), if we want a classifier with a 0.995 detection rate and a 0.2 false positive rate, AdaBoost needs at least 43 weak classifiers while AsymBoost$_{\mathrm{TC1}}$ needs 32 and AsymBoost$_{\mathrm{TC2}}$ needs only 22.



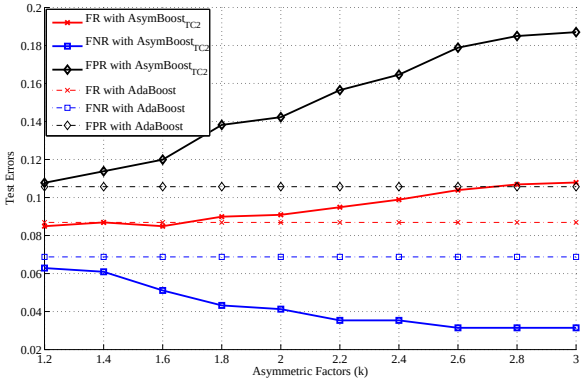
(a) AsymBoost$_{\mathrm{TC1}}$ vs AdaBoost



(b) False rates for AsymBoost$_{\mathrm{TC1}}$



(c) AsymBoost$_{\mathrm{TC2}}$ vs AdaBoost

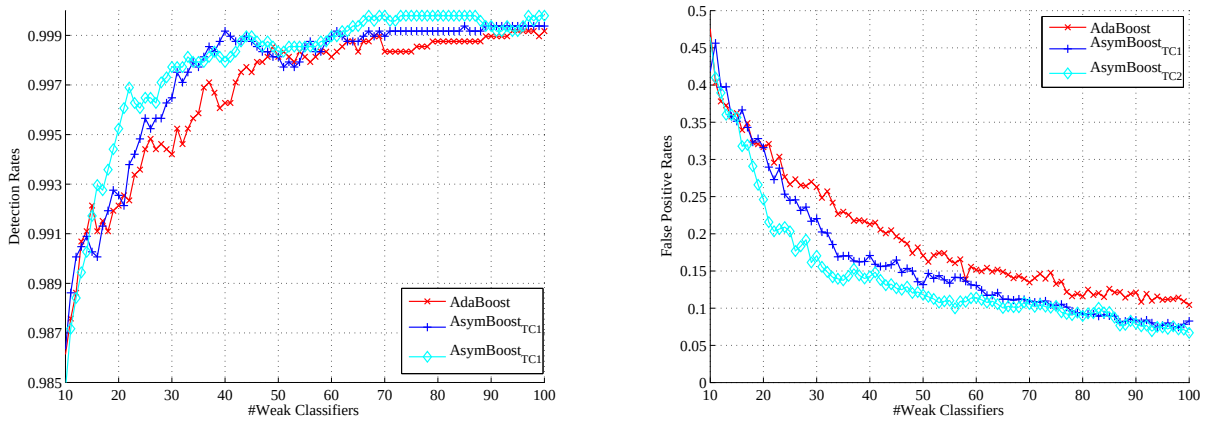


(d) False rates for AsymBoost$_{\mathrm{TC2}}$

Fig. 2: Results on the synthetic data for AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$, with a group of asymmetric factor k's. As the baseline, the results for AdaBoost are also shown in these figures. (a) and (c) demonstrate decision boundaries learned by AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$, with k is 2.0 or 3.0. The $\times$'s and $\square$'s stand for training negatives and training positives respectively. (b) and (d) demonstrate false rates (FR), false positive rates (FPR) and false negative rates (FNR) on test set with a group of k's ($1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8$ or 3.0), and the corresponding rates for AdaBoost is shown as dashed lines.

2) Using the same number of weak classifiers, our algorithms achieve a higher detection rate or a lower false positive rate. For example, from Fig. 3 (2), using 30 weak classifiers, both AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ achieve higher detection rates (0.9965 and 0.9975) than AdaBoost (0.9945).

Complete detectors Secondly, we train complete face detectors with AdaBoost of Viola-Jones [3], AsymBoost (asymmetric AdaBoost) of Viola-Jones [1], and the proposed AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$. All detectors are trained using the same training set. We use two types of cascade framework for the detector training: the traditional cascade of Viola and Jones [3] and the multi-exit cascade presented in [17]. The latter utilizes decision information of previous nodes when judging instances in the current node. For a fair comparison, all detectors use 24 nodes and 3332 weak classifiers. For each node, 5000 faces and 5000 non-faces are used for training, and 4832 faces and 5000 non-faces are used for validation. All non-faces are cropped from background images that do not contain any face. The asymmetric factor k for AsymBoost, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ are selected from $\{1.2,\ 1.5, 2.0, 3.0, 4.0, 5.0, 6.0\}$. The regularization factor θ for AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$ are chosen from $\{\frac{1}{50}, \frac{1}{60}, \frac{1}{70}, \frac{1}{80}, \frac{1}{90}, \frac{1}{100}, \frac{1}{200}, \frac{1}{400}, \frac{1}{800}, \frac{1}{1000}\}$. It takes about four hours to train a AsymBoost$_{\mathrm{TC}}$ face detector on a machine with 8 Intel Xeon E5520 processors and 32GB



(a) DR with fixed FPR

(b) FPR with fixed DR

Fig. 3: Testing curves of single-node classifiers for AdaBoost, AsymBoost$_{\mathrm{TC1}}$ and AsymBoost$_{\mathrm{TC2}}$. All the classifiers use the same training and test data sets. (a) shows curves of detection rates (DR) with false positive rates (FPR) fixed to 0.25, (b) shows curves of FPR with DR fixed to 0.995. FPR or DR are evaluated at every 10 weak classifiers.

memory. Comparing with AdaBoost, only around 0.5 hour extra time is spent on solving the primal problem at each iteration. We can say that, in the context of face detection, the training time of AsymBoost$_{\mathrm{TC}}$ is nearly the same as AdaBoost.

ROC curves on the CMU/MIT data set are shown in Fig. 4. For testing, the scale ratio is set to 1.25 and the scanning step-size is 1 pixel. A heuristic method to merge multiple detection windows is employed for post-processing, same as in [3]. The detected windows are grouped together if they are overlapped, and the merged window border is the mean of all windows in one group. Groups with more than three windows are reported as final detections.

Those images in the CMU/MIT data set containing ambiguous faces are removed and 120 images are retained. From the figure, we can see that, AsymBoost outperforms AdaBoost in both Viola-Jones cascade and the multi-exit cascade, which coincides with what was reported in [1]. Our algorithms have even better performance than all the other methods in all points and the improvements are more significant when the false positives are less than 100, which is the most important ROC region in practice. Fig. 5 shows the comparison results of our method and other state-of-the-arts (FloatBoost [2], cost-sensitive AdaBoost [20], BoostingChain [14], nested cascade [29]) on the CMU/MIT data set. These ROC curves are directly obtained from their original papers. Note that the experimental setups are not exactly the same, so the comparison is not very fair. For example, the nested cascade has used real-valued histograms based weak classifiers instead of the simplest discrete decision stumps, which may also contribute to the performance improvement.

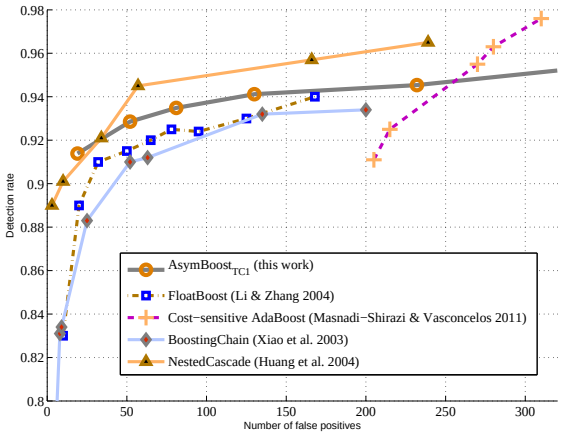
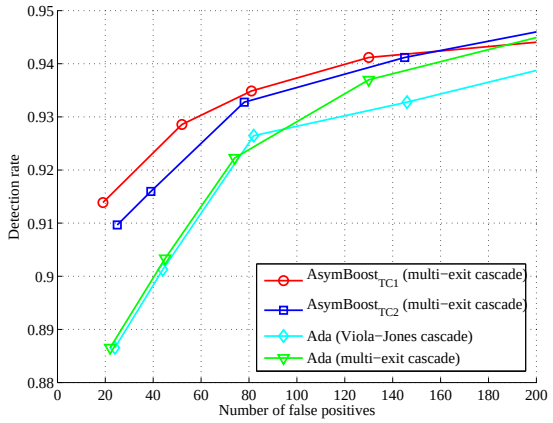


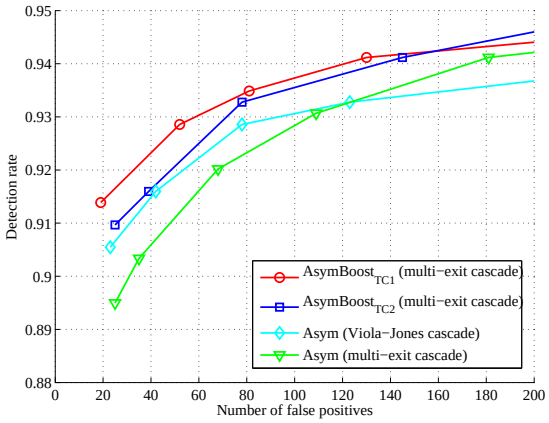
Fig. 5: Comparison of our algorithms with some state-of-the-art methods. See text for details.

The results shows that ours is the best in the low false positive rate part ($<$ 30 false positives). In terms of the overall performance, only the nested cascade is better than ours. However, the weak classifiers they used are much more powerful (confidence-rated weak classifiers) and we use the simplest discrete decision stumps. Fig. 6 shows some detected faces from the CMU/MIT data set.

As mentioned in the previous section, our algorithms produce sparse results to some extent. Some linear coefficients are zero when the corresponding weak classifiers satisfy the con-



(a) AsymBoost$_{\mathrm{TC}}$ vs AdaBoost



(b) AsymBoost$_{\mathrm{TC}}$ vs AsymBoost

Fig. 4: Performance of cascades evaluated by ROC curves on the MIT+CMU data set. AdaBoost is referred to "Ada", and AsymBoost of [1] is referred as to "Asym". "Viola-Jones cascade" means the traditional cascade used in [3] .

Fig. 6: Detected faces in the MIT+CMU data set by AsymBoost$_{\mathrm{TC1}}$.

dition (14). In the multi-exit cascade, the sparse phenomenon becomes more clear. Since correctly classified negative data are discarded after each node is trained, the training data for each node are different. The "closer" nodes share more common training examples, while the nodes "far away" from each other have distinct training data. The greater the distance between two nodes, the more uncorrelated they become. Therefore, the weak classifiers in the early nodes may not be very helpful in the last node, thus tending to be assigned zero coefficients. We call those weak classifiers with non-zero coefficients "effective" weak classifiers. Table I shows the ratios of "effective" weak classifiers contributed by one node to a specific successive node. To save space, only the first 15 nodes are demonstrated. We can see that, the ratio decreases with the growth of the node index, which means that the farther the preceding node is from the current node, the less useful it is for the current node. For example, the first node has almost no contribution after the eighth node. Table II shows the number of effective weak classifiers used by our algorithm and the traditional stage-wise boosting. All weak classifiers in stage-wise boosting have non-zero coefficients, while our totally-corrective algorithm uses much fewer effective weak classifiers.

To compare real-time speed in the test phase, we obtain a sequence of 320×240 pixels face images. Each image is scanned with 2 pixels stride and 1.2 scale ratio. Totally $13,477,226$ sub-windows are scanned in 289 images. All the programs are written in C++ and tested on one Intel Xeon E5520 2.27GHz processor. From Table III, we find that the speed of our algorithms and AdaBoost is similar, while ours has better performance.

TABLE III: Real-time performance of face detectors. There are the same number of nodes and weak classifiers in both the multi-exit cascade and the Viola-Jones cascade. This figure shows windows per second (win ps), seconds per image (sec pi) and features per window (fea pw), on average over all images.

	win ps	sec pi	fea pw
AsymBoost$_{\mathrm{TC1}}$+ multi-exit cascade	641467	0.0715	48.55
AsymBoost$_{\mathrm{TC2}}$+ multi-exit cascade	647321	0.0720	49.09
AdaBoost + Viola-Jones cascade	652650	0.0727	55.36

TABLE I: This table shows the sparseness of weak classifier coefficients, for the face detector trained with AsymBoost$_{\mathrm{TC1}}$and the multi-exit cascade. The ratio of weak classifiers selected at the i-th node (column) appearing with non-zero coefficients in the j-th node (row). The ratios decrease along with the growth of the node index in each column.

Node Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.00														
2	1.00	1.00													
3	1.00	1.00	1.00												
4	0.86	1.00	0.97	1.00											
5	0.43	0.93	0.97	0.97	1.00										
6	0.71	0.93	0.90	1.00	0.96	1.00									
7	0.43	0.87	0.87	0.97	0.92	0.92	1.00								
8	0.29	0.40	0.70	0.73	0.74	0.88	0.74	1.00							
9	0.00	0.27	0.50	0.60	0.76	0.72	0.66	0.67	1.00						
10	0.14	0.27	0.43	0.60	0.62	0.70	0.62	0.66	0.60	1.00					
11	0.00	0.20	0.33	0.50	0.52	0.54	0.60	0.59	0.56	0.48	1.00				
12	0.14	0.20	0.40	0.40	0.56	0.50	0.54	0.61	0.55	0.46	0.36	1.00			
13	0.00	0.13	0.33	0.37	0.36	0.54	0.40	0.47	0.47	0.46	0.43	0.25	1.00		
14	0.00	0.07	0.17	0.40	0.28	0.50	0.42	0.49	0.50	0.53	0.45	0.43	0.35	1.00	
15	0.00	0.13	0.20	0.27	0.36	0.38	0.46	0.41	0.52	0.42	0.49	0.44	0.34	0.27	1.00

TABLE II: Comparison of the numbers of the effective weak classifiers for the stage-wise boosting (SWB) and the totally-corrective boosting (TCB). We take AdaBoost and AsymBoost$_{\mathrm{TC1}}$ as representative types of SWB and TCB, both of which are trained in the multi-exit cascade for face detection.

Node Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SWB	7	22	52	82	132	182	232	332	452	592	752	932	1132	1332	1532	1732	1932	2132
TCB	7	22	52	80	125	174	213	269	331	441	464	538	570	681	717	744	742	879

C. Pedestrian detection

The INRIA pedestrian dataset is used here for evaluating our algorithms and others. This data set has pre-defined training set and test set. The training set contains 1208 cropped pedestrian sub-windows and 1200 large non-pedestrian images. The size of sub-windows is 64×128 pixels, and a 16 pixels pad on each side is added to preserve border information. To make the maximum use of the training data, we also use the mirrored pedestrian sub-windows, thus totally 2416 training pedestrians obtained. The test set is made up of 288 images holding 588 annotated pedestrians inside and 453 non-pedestrian images.

Histogram of oriented gradient (HOG) features are applied to pedestrian detection by Dalal and Triggs [30]. They use HOG with linear support vector machines (SVMs) to train a pedestrian detector. Zhu *et al*. [31] use AdaBoost and the cascade framework for pedestrians. Similar to the work in [31], we exhaustively scan blocks with different sizes, scales and ratios in the sub-window. Five types of width/height ratios are adopted: $1:1$, $1:2$, $2:1$, $1:3$ and $3:1$. The scales are from 12×12 to 64×128, and the block stride is 4 pixels. Thus, totally 7735 blocks are obtained for a 64×128 pixels sub-window. In order to save training time, only 10% blocks are uniformly sampled in each round for training weak classifiers. There are 2×2 cells within each block. For each cell, a 9-bin histogram of gradients are summarized with respect to orientations. Then the concatenated 36-D feature vector is ℓ_1 normalized in each block. To accelerate the computational speed, we have used integral gradient images for calculating HOG features. In this way, 9 integral images are generated corresponding to histogram bins. For sequel operations, only 9×9 memory accesses are needed for compute the 36-D vector for each block.

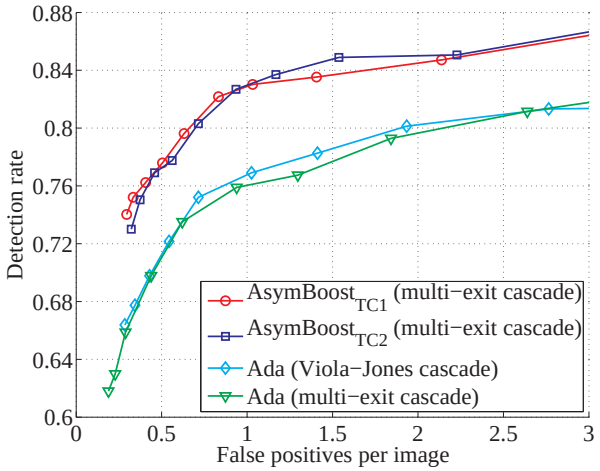
Besides our algorithms, AdaBoost and AsymBoost are evaluated in the experiments with the same criterion. We also compare our algorithms with HOG with linear SVM [30] and pyramid HOG (PHOG) with intersection kernel SVM (IKSVM) [32][1]. ROC curves on the test set of INRIA data are shown in Fig. 7.

In [31], linear SVM are used as weak classifiers. Nevertheless, linear SVM does not have a closed-form solution and usually a quadratic problem is solved for each weak classifier, which is time-consuming. Moreover, there is a model parameter C to be determined commonly via cross-validation, which is not feasible here. Alternatively, we use weighted LDA as weak classifiers, which has a closed-form solution and is easy to solve.

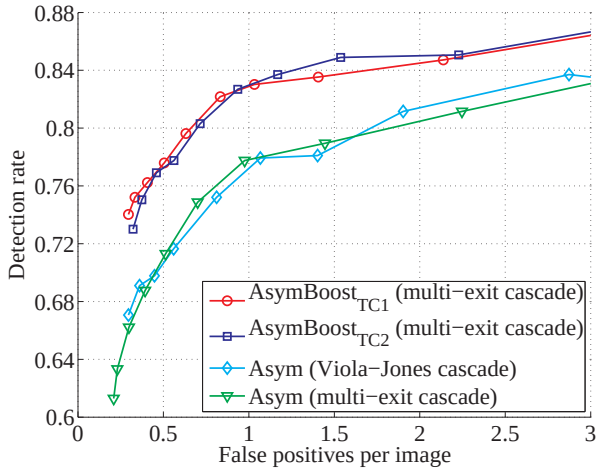
Same as in face detection, we prescribe the number of nodes (21) and weak classifiers (612) for all methods. The first three nodes have 4 weak classifiers, and the maximum number is restricted to 60.

The training set consists of 2416 positives and 2416 negatives. The positive examples are kept the same in the entire training process. The negative examples for the first node are obtained by randomly sampling from those 1200 non-pedestrian images in the INRIA dataset. For the latter nodes, true rejections for the current detector are removed and the

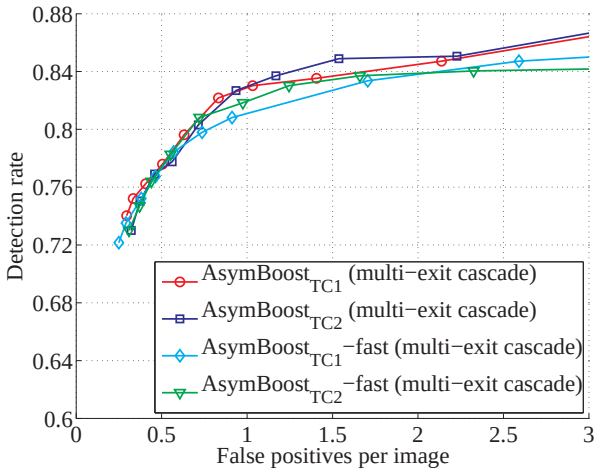
[1]The trained models and codes for HOG with linear SVM and PHOG with intersection kernel SVM are obtained from http://www.cs.berkeley.edu/~smaji/projects/ped-detector/



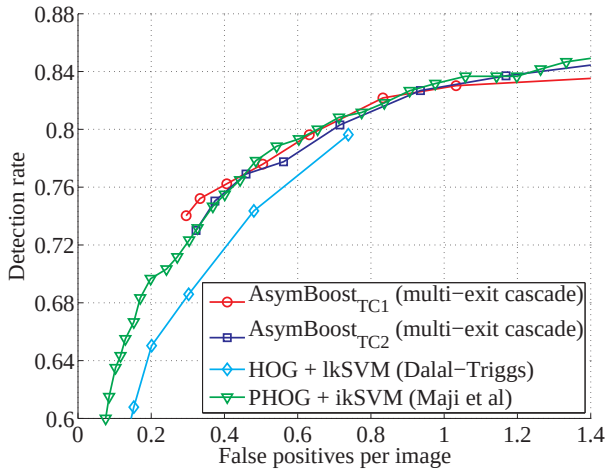
(a) AsymBoost$_{\mathrm{TC}}$ vs AdaBoost



(b) AsymBoost$_{\mathrm{TC}}$ vs AsymBoost



(c) AsymBoost$_{\mathrm{TC}}$ vs AsymBoost$_{\mathrm{TC}}$-fast



(d) AsymBoost$_{\mathrm{TC}}$ vs SVM

Fig. 7: ROCs curves for INRIA dataset. "Ada" denotes AdaBoost, and "Asym" denotes AsymBoost [3]. "Viola-Jones cascade" means the traditional cascade used in [1]. "LSVM" is linear SVM, while "IKSVM" is intersection kernel SVM [32].

vacancies are filled with false positives randomly scanned from the non-pedestrian images (*i.e.*, bootstrapping). For the purpose of validation, 500 extra false positives are collected by bootstrapping.

The asymmetric parameter k is selected from the same range in face detection. On the other hand, the regularization factor θ is selected from $\{\frac{1}{15}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}, \frac{1}{40}, \frac{1}{50}, \frac{1}{60}\}$.

Our algorithms perform better than AdaBoost, AsymBoost, HOG with linear SVM, and achieve similar performance with PHOG with IKSVM. We also test the AsymBoost$_{\mathrm{TC1}}$-fast and AsymBoost$_{\mathrm{TC2}}$-fast, 4 weak classifiers are added in each iteration. From Fig. 7(c), we find that the performance of AsymBoost$_{\mathrm{TC}}$-fast are slightly worse than AsymBoost$_{\mathrm{TC}}$, but they are still better than AdaBoost, AsymBoost and HOG with linear SVM. The scale ratio for input images is 1.0905 and the sub-window stride is 8×8 pixels. In the same manner with [32], we utilize mean shift to merge multiple sub-windows for the same object. Since the 64×128 sub-windows keep 16 pixels of margin on four sides, we shrink those sub-windows to 32×96 pixels before merging them. It takes about 5 hours to train a complete detector on the same machine mentioned in the face detection section. To determine a detected sub-window is true positive or not, we adopt the criterion in PASCAL VOC

Challenge [33] as follows:

$$\alpha = \frac{\text{area}\{B_d \cap B_{gt}\}}{\text{area}\{B_d \cup B_{gt}\}} > 0.5, \tag{18}$$

which means that if the overlap between detected sub-window B_d and the ground truth B_{gt} exceeds 50%, we will consider it is a true positive. The multiple detections on the same object is treated as false positive. Fig. 8 shows some detection examples in INRIA data set.

Fig. 8: Detected pedestrians in the INRIA data set by AsymBoost$_{\text{TC1}}$.

Table IV and Table V show the effective number of weak classifiers in each node of AsymBoost$_{\text{TC1}}$ pedestrian detector. Similar phenomenon is demonstrated, like face detector. One interesting result is that last several nodes share very few effective weak classifiers with preceding nodes. For example, the 15-th node does not "inherit" any effective weak classifiers from the first to the 14-th nodes. In this fashion, the multi-exit cascade does not bring improvements, as all the historical information is useless.

We use a surveillance video for testing the real-time performance[2], containing 295 frames of 384×288 pixels images. The images are scanned with 8×8 stride and 1.0905 scale ratio, totally incurring 1413050 sub-windows. AsymBoost$_{\text{TC1}}$, AsymBoost$_{\text{TC2}}$ and AdaBoost are implemented by C++, while the code for PHOG with IKSVM is written using Matlab and C. All the programs are tested on a single Intel Xeon E5520 2.27GHz processor. Table VI shows the speeds of AsymBoost$_{\text{TC1}}$, AsymBoost$_{\text{TC2}}$, AdaBoost, and PHOG with IKSVM. Our algorithms runs as fast as AdaBoost, and is around 10 times faster than PHOG with IKSVM. From the number of evaluated features, we can obtain the same conclusion.

[2]The pedestrian video is obtained from http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

TABLE VI: Real-time performance of pedestrian detectors. The same numbers of weak classifiers and nodes are used in both the multi-exit cascade and the Viola-Jones cascade. The average number of windows per second (win ps), seconds per image (sec pi) and features per window (fea pw) are demonstrated. Note that, for boosting methods, one block is divided into 2×2 cells.

	win ps	sec pi	fea pw
AsymBoost$_{\text{TC1}}$ + multi-exit cascade	22479	0.213	14.9 blocks
AsymBoost$_{\text{TC2}}$ + multi-exit cascade	22358	0.214	15.6 blocks
AdaBoost + Viola-Jones cascade	21341	0.224	17.7 blocks
PHOG + intersection kernel SVM	2275.7	2.10	252 cells

VI. Conclusion and Discussion

We have proposed asymmetric totally-corrective boosting algorithms for object detection. Our algorithms directly optimize asymmetric loss functions using the column generation technique. The algorithms are guaranteed to converge to the global optimum of the empirical risk. Our algorithms update training example weights in a totally-corrective fashion, and the coefficients of all weak classifiers are updated at each iteration. With the ℓ_1 norm regularization, the linear coefficients are sparse in the context of cascade classifiers. By learning multiple classifiers in a single iteration, we establish a fast training version of proposed methods, which significantly reduces the training time.

Experiments have demonstrated that both our algorithms achieve better results for face/pedestrian detection than AdaBoost and Viola-Jones' AsymBoost. One observation is that we do not see remarkable differences in performance between AsymBoost$_{\text{TC1}}$ and AsymBoost$_{\text{TC2}}$ in our experiments. For the face detection task, AdaBoost already achieves a very promising result, so the improvements of our method are not that significant, as expected. For pedestrian detection, our algorithms considerably outperform AdaBoost, AsymBoost and the linear SVM with HOG features, and achieve comparable performance with PHOG with the intersection kernel SVM, which is considered one of the state-of-the-art pedestrian detectors. However, our methods run about 10 times faster than PHOG with the intersection kernel SVM.

We find that solely using the multi-exit cascade with AdaBoost achieves better results in the early nodes and worse results in the late nodes. The explanation might be that, for the late nodes close to the end, the negative training examples differ from those in the early nodes. Since AdaBoost is stage-wise, only the last weak classifier's weight is updated. So

TABLE IV: The sparseness situation for the pedestrian detector trained with AsymBoost$_{\mathrm{TC1}}$and the multi-exit cascade. The ratio of weak classifiers selected at the i-th node (column) appearing with non-zero coefficients in the j-th node (row). For the last several rows, the non-zero coefficients ratios of previous nodes are zero, which means that the historical information is useless for the latter nodes.

Node Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.00														
2	1.00	1.00													
3	1.00	1.00	1.00												
4	1.00	1.00	1.00	1.00											
5	1.00	1.00	1.00	1.00	1.00										
6	1.00	1.00	1.00	0.88	0.75	1.00									
7	0.50	0.75	1.00	0.75	1.00	0.63	1.00								
8	0.25	0.75	1.00	0.88	0.75	1.00	0.50	1.00							
9	0.50	0.50	0.75	0.38	0.75	0.88	0.75	0.00	1.00						
10	0.25	0.50	0.50	0.63	0.50	0.38	0.33	0.42	0.00	1.00					
11	0.00	0.25	0.25	0.25	0.25	0.25	0.33	0.33	0.08	0.15	1.00				
12	0.25	0.00	0.00	0.50	0.25	0.13	0.42	0.17	0.08	0.05	0.10	1.00			
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	1.00		
14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	1.00	
15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

TABLE V: For pedestrian detection, the numbers of the effective weak classifiers for the stage-wise boosting (SWB) and the totally-corrective boosting (TCB).

Node Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SWB	4	8	12	20	28	36	48	60	72	92	112	132	172	212	252	312	372	432
TCB	4	8	12	20	28	33	40	47	44	46	40	39	41	41	41	61	63	63

all the previous weak classifiers are inherited, no matter if a weak classifier is helpful or not. On the contrary, our totally-corrective algorithms update all weights at each round. Those ineffective weak classifiers are automatically "switched off" by setting the corresponding weights to zeros.

The framework for constructing totally-corrective boosting algorithms is general, so we can consider other asymmetric losses (*e.g.*, asymmetric exponential loss) to form new asymmetric boosting algorithms. We leave this as a future research topic.

Motivated by the analysis of sparseness, we find that the very early nodes contribute little information for training the later nodes. Based on this, we can exclude some useless nodes when the node index grows, which will simplify the multi-exit structure and decrease the testing time.

REFERENCES

[1] P. Viola and M. Jones, "Fast and robust classification using asymmetric AdaBoost and a detector cascade," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, B. C., Canada, 2001, pp. 1311–1318.

[2] S. Z. Li and Z. Zhang, "FloatBoost learning and statistical face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1112–1123, 2004.

[3] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comp. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[4] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Trans. Neural Netw.*, 2010. [Online]. Available: http://dx.doi.org/10.1109/TNN.2010.2066286

[5] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht, "Neural network approach to background modeling for video object segmentation," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1614–1627, 2007.

[6] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority oversampling in boosting," *IEEE Trans. Neural Netw.*, 2010. [Online]. Available: http://dx.doi.org/10.1109/TNN.2010.2066988

[7] M. Warmuth, J. Liao, and G. Rätsch, "Totally corrective boosting algorithms that maximize the margin," in *Proc. ACM Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 1001–1008.

[8] V. Gomez-Verdejo, J. Arenas-Garcia, and A. R. Figueiras-Vidal, "A dynamically adjusted mixed emphasis method for building boosting ensembles," *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 3–17, 2008.

[9] W. Fan, S. Stolfo, J. Zhang, and P. Chan, "Adacost: Misclassification cost-sensitive boosting," in *Proc. ACM Int. Conf. Mach. Learn.*, Bled, Slovenia, 1999, pp. 97–105.

[10] X. Hou, C. Liu, and T. Tan, "Learning boosted asymmetric classifiers for object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, New York City, USA, 2006, pp. 330–338.

[11] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 369–382, 2008.

[12] H. Luo, "Optimization design of cascaded classifiers," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, San Diego, CA, USA, 2005, pp. 480–485.

[13] S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Towards optimal training of cascaded detectors," in *Proc. Eur. Conf. Comp. Vis.*, Graz, Austria, 2006, pp. 325–337.

[14] R. Xiao, L. Zhu, and H. Zhang, "Boosting chain learning for object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, Nice, France, 2003, pp. 709–715.

[15] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, San Diego, CA, US, 2005, pp. 236–243.

[16] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: Tradeoff between accuracy and cost," in *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, Washington, DC, USA, 2010, pp. 853–860.

[17] M.-T. Pham, V.-D. D. Hoang, and T.-J. Cham, "Detection with multi-exit asymmetric boosting," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Anchorage, Alaska, USA, 2008.

[18] R. Xiao, H. Zhu, H. Sun, and X. Tang, "Dynamic cascades for face detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, Rio de Janeiro, Brazil, 2007.

[19] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Asymmetric totally-

corrective boosting for real-time object detection," in *Proc. Asian Conf. Comp. Vis.*, 2010, Lecture Notes in Computer Science, Springer.
[20] H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.
[21] J. Kivinen and M. K. Warmuth, "Boosting as entropy projection," in *Proc. ACM Annual Conf. Comp. Learn. Theory*, Santa Cruz, CA, USA, 1999, pp. 134–144.
[22] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.47
[23] ——, "Boosting through optimization of margin distributions," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 659–666, 2010.
[24] A. Demiriz, K. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1-3, pp. 225–254, 2002.
[25] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
[26] G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller, "Constructing boosting algorithms from SVMs: An application to one-class classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1184–1199, 2002.
[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[28] C. Zhu, R. H. Byrd, and J. Nocedal, "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization," *ACM Trans. Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.
[29] C. Huang, H. Ai, B. Wu, and S. Lao, "Boosting nested cascade detector for multi-view face detection," in *Proc. IEEE Int. Conf. Patt. Recogn.*, 2004, pp. 415–418.
[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, vol. 1, San Diego, CA, 2005, pp. 886–893.
[31] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, vol. 2, New York City, USA, 2006, pp. 1491–1498.
[32] S. Maji, A. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Anchorage, Alaska, USA, 2008, pp. 1–8.
[33] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2010 (VOC2010) results." [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2010/workshop/