Efficiently Learning a Detection Cascade With Sparse Eigenvectors

Chunhua Shen, Sakrapee Paisitkriangkrai, and Jian Zhang, *Senior Member, IEEE*

***Abstract*—Real-time object detection has many computer vision applications. Since Viola and Jones [1] proposed the first real-time AdaBoost based face detection system, much effort has been spent on improving the boosting method. In this work, we first show that feature selection methods other than boosting can also be used for training an efficient object detector. In particular, we introduce greedy sparse linear discriminant analysis (GSLDA) [2] for its conceptual simplicity and computational efficiency; and slightly better detection performance is achieved compared with [1]. Moreover, we propose a new technique, termed boosted greedy sparse linear discriminant analysis (BGSLDA), to efficiently train a detection cascade. BGSLDA exploits the sample reweighting property of boosting and the class-separability criterion of GSLDA. Experiments in the domain of highly skewed data distributions (e.g., face detection) demonstrate that classifiers trained with the proposed BGSLDA outperforms AdaBoost and its variants. This finding provides a significant opportunity to argue that AdaBoost and similar approaches are not the only methods that can achieve high detection results for real-time object detection.**

***Index Terms*—AdaBoost, asymmetry, cascade classifier, feature selection, greedy sparse linear discriminant analysis (GSLDA), object detection.**

I. Introduction

REAL-TIME objection detection such as face detection has numerous computer vision applications, e.g., intelligent video surveillance, vision based teleconference systems and human motion analysis [3], [4]. Various detectors have been proposed in the literature [1], [5]–[8]. Object detection is challenging due to large variations of the visual appearances, poses and illumination conditions. Furthermore, object detection is a highly imbalanced classification task. A typical natural image contains many more negative background patterns than object patterns. The number of background patterns can be 100 000 times larger than the number of object patterns. That means, if one wants to achieve a high detection rate, together with a low false detection rate, one needs to design a specific and sensitive classifier that takes the imbalanced data distribution into consideration [9].

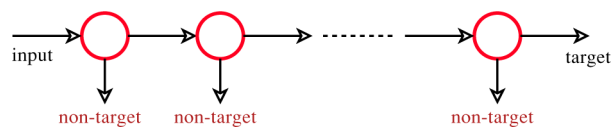


Fig. 1. Cascade classifier with multiple nodes. Here a circle represents a node classifier. An input patch is classified as a target only when it passes tests at each node classifier.

Viola and Jones [1] proposed the first real-time AdaBoost based face detector. They introduced a framework for selecting discriminative features and training classifiers in a cascaded manner as shown in Fig. 1. The cascade framework allows most nonface patches to be rejected quickly before reaching the final node, resulting in fast performances. A test image patch is reported as a face only if it passes tests in all nodes. This way, most nonface patches are rejected by these early nodes. Cascade detectors have led to very fast detection speed and high detection rates. Due to their tremendous success, numerous further work have been proposed. Most of them focused on improving the underlying boosting method or accelerating the training process. For example, AsymBoost was introduced in [9] to alleviate the limitation of AdaBoost in the context of highly skewed example distributions. Li *et al.* [10] proposed FloatBoost for a better detection accuracy by introducing a backward feature elimination step into the AdaBoost training procedure. Wu *et al.* [11] used forward feature selection for fast training by ignoring the reweighting scheme in AdaBoost. Another technique based upon the statistics of the weighted input data was used in [12] for even faster training. KLBoost was proposed in Liu and Shum [13] to train a strong classifier. The weak classifiers of KLBoost are based upon histogram divergence of linear features. Notice that in KLBoost, the classifier design is separated from the feature selection process. Bourdev and Brandt [14] developed the soft cascade to reduce the complexity of cascade design and training. The idea was further improved in multiexit boosted classifiers [15]. Cascade classifiers were applied not only to boosting based classifiers, but also to support vector machines (SVMs) [5]. In this work, we propose an improved learning algorithm for face detection, dubbed boosted greedy sparse linear discriminant analysis (BGSLDA). A preliminary version of this work appeared in Paisitkriangkrai *et al.* [16]. In this paper, we have included more discussion and experimental results with various features on both faces and humans.

Manuscript received January 20, 2010; revised May 10, 2010, June 15, 2010; accepted June 15, 2010. Date of publication July 01, 2010; date of current version December 17, 2010. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Winkler.

C. Shen is with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia. He is also with the Australian National University, Canberra, ACT 0200, Australia (e-mail: chunhua.shen@nicta.com.au).

S. Paisitkriangkrai is with the University of Adelaide, Adelaide, SA 5005, Australia (e-mail: paul.paisitkriangkrai@adelaide.edu.au).

J. Zhang is with NICTA, Neville Roach Laboratory, Kensington, NSW 2052, Australia. They are also with the University of New South Wales, Sydney, NSW 2052, Australia (e-mail: jian.zhang@nicta.com.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2010.2055880

One issue that contributes to the efficacy of the system comes from the use of AdaBoost for training cascade nodes. AdaBoost is a forward stage-wise additive modeling with the weighted exponential loss function [17]–[19]. The algorithm combines an ensemble of weak classifiers to produce a final strong classifier with high classification accuracy. AdaBoost chooses a small subset of weak classifiers and assigns them with proper coefficients. The linear combination of weak classifiers can be interpreted as a decision hyper-plane in the weak classifier space. The proposed BGSLDA differs from the original AdaBoost in the following aspects. Instead of selecting decision stumps with minimal weighted error as in AdaBoost, the proposed algorithm finds a new weak learner that maximizes the class-separability criterion. As a result, the coefficients of selected weak classifiers are updated repetitively during the learning process according to this criterion. It seems that the class-separability criterion better considers the asymmetric characteristic of the detection problem.

Our technique differs from [11] in the following aspects. Wu *et al.* [11] proposed the concept of linear asymmetric classifier (LAC) by addressing the *asymmetric node learning* goal in the cascade framework. Unlike our work where the features are selected based upon the linear discriminant analysis (LDA) criterion, Wu *et al.* select features using the AdaBoost/AsymBoost algorithm [11]. Given the selected features, Wu *et al.* then build an optimal linear classifier for the node learning goal using LAC or LDA. Note that similar techniques have also been applied in neural networks. In [20], a nonlinear adaptive feed-forward layered network with linear output units has been introduced. The input data is nonlinearly transformed into a space in which classes can be separated more easily. As our experiments demonstrate, in terms of feature selection, the proposed BGSLDA method is superior than AdaBoost and AsymBoost for object detection.

The key contributions of this work are as follows.

1) We introduce GSLDA as an alternative approach for training face detectors. Similar results are obtained compared with Viola and Jones' approach.
2) We propose a new algorithm, BGSLDA, which combines the sample reweighting schemes typically used in boosting into GSLDA. Experiments show that BGSLDA can achieve better detection performances.
3) Wu *et al.* [11] showed that feature selection and classifier training techniques can have different objective functions (in other words, the two processes can be separated) in the context of training a visual detector. Our proposed GSLDA based approach provides new evidence to confirm this observation.
4) Our results confirm that it is beneficial to consider the highly skewed data distribution when training a detector. LDA's learning criterion already incorporates this imbalanced data information. Hence, it is better than standard AdaBoost's exponential loss for training an object detector.

The remaining parts of the paper are structured as follows. In Section II-A, the GSLDA algorithm is introduced as an alternative learning technique to object detection problems. We then discuss how LDA incorporates imbalanced data information when training a classifier in Section II-B. Then, in Sections II-C and II-D, the proposed BGSLDA algorithm is described and the training time complexity is discussed. Experimental results are shown in Section III and the paper is concluded in Section IV.

II. Algorithms

In this section, we present alternative techniques to AdaBoost for object detection. We start with a short explanation of the concept of GSLDA [21]. Next, we show that like AsymBoost [9], LDA is better at handling asymmetric data than AdaBoost. We also propose a new algorithm that makes use of sample reweighting schemes commonly used in AdaBoost to select a subset of relevant features for training the GSLDA classifier. Finally, we analyze the training time complexity of the proposed methods.

A. GSLDA

LDA can be cast as a generalized eigenvalue decomposition. Given a pair of symmetric matrices corresponding to the between-class (S_b) and within-class covariance matrices (S_w), one maximizes a class-separability criterion defined by the generalized Rayleigh quotient

$$\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^\top S_b \boldsymbol{w}}{\boldsymbol{w}^\top S_w \boldsymbol{w}}. \tag{1}$$

The optimal solution of a generalized Rayleigh quotient is the eigenvector corresponding to the maximal eigenvalue. The sparse version of LDA is to solve (1) with an additional sparsity constraint

$$\mathbf{card}(\boldsymbol{w}) = k \tag{2}$$

where $\mathbf{card}(\cdot)$ counts the number of nonzero components, *a.k.a.* the ℓ_0 norm. $k \in \mathbb{Z}^+$ is an integer set by the user. Due to this sparsity constraint, the problem becomes nonconvex and NP-hard. In [21], Moghaddam *et al.* presented a technique to compute optimal sparse linear discriminants using the branch and bound approach. Nevertheless, finding the exact globally optimal solution for high-dimensional data is computationally infeasible. The algorithm was extended in [2], with new sparsity bounds and efficient matrix inverse techniques to speed up the computation time by 1000 fold. The technique works by sequentially adding the new variable which yields the maximum eigenvalue (forward selection) until the maximum number of elements are selected or some predefined condition is met. As shown in [2], for two-class problems, the computation can be made very efficient as the only finite eigenvalue $\lambda_{\max}(S_b, S_w)$ can be computed in closed-form as $\boldsymbol{b}^\top S_w^{-1} \boldsymbol{b}$ with $S_b = \boldsymbol{b}\boldsymbol{b}^\top$ because in this case S_b is a rank-one matrix. $\boldsymbol{b}$ is a column vector. Therefore, the computation is mainly determined by the inverse of S_w. When a greedy approach is adopted to sequentially find the suboptimal $\boldsymbol{w}$, a simple rank-one update for computing S_w^{-1} significantly reduces the computation complexity [2]. We have mainly used forward greedy search in this work. For forward greedy search, if l is the current subset of k indices and $m = l \cup i$ for candidate i which is not in l. The new augmented

inverse $(S_w^m)^{-1}$ can be calculated in a fast way by recycling the last step's result $(S_w^l)^{-1}$

$$(S_w^m)^{-1} = \begin{bmatrix} (S_w^l)^{-1} + a_i \boldsymbol{u}_i \boldsymbol{u}_i^\top & -a_i \boldsymbol{u}_i \\ -a_i \boldsymbol{u}_i & a_i \end{bmatrix} \tag{3}$$

where $\boldsymbol{u}_i = (S_w^l)^{-1} S_{w,li}$ with (li) indexing the lth rows and ith column of S_w and $a_i = 1/(S_{w,ii} - S_{w,li}^\top \boldsymbol{u}_i)$ [2], [22].

Note that we have experimented with other sparse linear regression and classification algorithms, e.g., ℓ_1-norm linear SVMs, ℓ_1-norm regularized log-linear models, etc. However, the major drawback of these techniques is that they do not have an *explicit* parameter that controls the number of features to be selected. The tradeoff parameter (regularization parameter) only controls the degree of sparseness. One has to tune this parameter using tedious trial-and-error. Hence, we have decided to apply GSLDA, which makes use of greedy feature selection and the number of features can be predefined. It would be of interest to compare our method with ℓ_1-norm induced sparse models [23].

The following paragraph explains how we apply the GSLDA classifier [2], [21] as an alternative feature selection method to classical Viola and Jones' framework [1].

Due to space limit, we omit the explanation of cascade classifiers. Interested readers should refer to [1] and [11] for details. The proposed GSLDA based detection framework is summarized in Algorithm 1. The algorithm operates as follows. The set of selected features is initialized to an empty set. The first step (lines 4–5 in Algorithm 1) is to train weak classifiers, for example, decision stumps on Haar features.[1] For each Haar-like rectangle feature, the threshold that gives the minimal classification error is precomputed and stored in memory. In order to achieve maximum class separation, the output of each decision stump is examined and the decision stump whose output yields the maximum eigenvalue is sequentially added to the list [line 7, step (1)]. The process continues until the predefined condition is met (line 6).

B. LDA on Asymmetric Data

In cascade classifiers, we would prefer to have a classifier that yields high detection rates without introducing many false positives. In the Bayes sense, linear discriminant classifiers are optimum for normal distributions with equal covariance matrices. However, due to its simplicity and robustness, linear discriminant classifiers have shown to perform well not only for normal distributions with unequal covariance matrices but also nonnormal distributions. A linear discriminant classifier can be written as

$$F(\boldsymbol{x}) = \begin{cases} +1, & \text{if } \sum_{t=1}^{n} w_t h_t(\boldsymbol{x}) + \theta \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{4}$$

[1]We introduce nonlinearity into our system by applying decision stumps to raw Haar feature values. By nonlinearly transforming the data, the input can now be separated more easily using simple linear classifiers [20].

Note that any classifiers can be applied here. We also use LDA on covariance features for human detection as described in [8]. For the time being, we focus on decision stumps on Haar-like features. We will give details about covariance features later.

where $h(\cdot)$ defines a function which returns a binary output, $\boldsymbol{x}$ is the input image features and θ is an optimal threshold such that the minimum number of examples are misclassified.

The asymmetric goal for training cascade classifiers can be written as a tradeoff between false acceptance rate ε_1 and false rejection rate ε_2 as

$$r = \varepsilon_1 + \mu \varepsilon_2 \tag{5}$$

where μ is a tradeoff parameter, representing an acceptable false rejection rate at the cost of higher false acceptance rate. Various approaches have been proposed to determine this tradeoff [1], [11], [14], [24]. The objective of LDA is to maximize the projected between-class covariance matrix (the distance between the mean of two classes) and minimize the within-class covariance matrix. The choice of weight coefficients, $\boldsymbol{w}$, which satisfies the LDA objective is guaranteed to achieve this goal. Having large projected mean difference and small projected class variance indicates that the data can be separated more easily and, hence, the asymmetric goal can also be achieved more easily. On the other hand, AdaBoost minimizes the symmetric exponential loss function that does not guarantee high detection rates with few false positives [9]. The selected features are, therefore, no longer optimal for the task of rejecting negative samples.

Another way to think of this is that AdaBoost sets initial positive and negative sample weights to $0.5/N_p$ and $0.5/N_n$ (N_p and N_n are the number of positive samples and negative samples). The prior information about the number of samples in each class is encoded only in the initial distribution of sample weights. The information gradually *phased out* during subsequent weak learners' training. In contrast, LDA takes the number of samples in each class into consideration when solving the optimization problem, i.e., the number of samples is used in calculating the between-class covariance matrix (S_b). Hence, S_b is the weighted difference between class mean and sample mean, which writes

$$S_b = \sum_{c_i} N_{c_i} (\mu_{c_i} - \overline{x})(\mu_{c_i} - \overline{x})^\top \tag{6}$$

where $\mu_{c_i} = N_{c_i}^{-1} \sum_{j \in c_i} x_j$; $\overline{x} = N^{-1} \sum_j x_j$; N_{c_i} is the number of samples in class c_i and N is the total number of samples. This extra information minimizes the effect of imbalanced data set.

In order to demonstrate this, we generate an artificial data set similar to the one used in [9]. We learn a classifier consisting of four linear classifiers and the results are shown in Fig. 2. From the figure, we see that the first weak classifier (#1) selected by both algorithms are the same since it is the only linear classifier with minimal error. AdaBoost then reweights the samples and selects the next classifier (#2) which has the smallest weighted error. From the figure, the second weak classifier (#2) introduces more false positives to the final classifier. Since most positive samples are correctly classified, the positive samples' weights are close to zero. AdaBoost selects the next classifier (#3) which classifies all samples as negative. Therefore, it is clear that all but the first weak classifier learned by AdaBoost are poor because

Algorithm 1 The training procedure for building a cascade of GSLDA object detector.

Input:

- A positive training set and a negative training set;
- A set of Haar-like rectangle features $h_1, h_2, \cdots$;
- $D_{\min}$: minimum acceptable detection rate per cascade level;
- $F_{\max}$: maximum acceptable false positive rate per cascade level;
- F_{target}: target overall false positive rate;

Initialize: $i = 0$; $D_i = 1$; $F_i = 1$;
while $F_{\text{target}} < F_i$ **do**
  $i = i + 1$; $f_i = 1$;
  foreach *feature* **do**
    Train a weak classifier (*e.g.*, a decision stump parameterized by a threshold θ) with the smallest error on the training set;
  while $f_i > F_{\max}$ **do**
    1. Add the best weak classifier (*e.g.*, decision stump) that yields $\lambda_{\max}(S_b, S_w)$;
    2. Lower a linear classifier threshold, θ in (4), such that $D_{\min}$ holds;
    3. Update f_i using this classifier threshold;
  $D_{i+1} = D_i \times D_{\min}$; $F_{i+1} = F_i \times f_i$; and remove correctly classified negative samples from the training set;
  if $F_{\text{target}} < F_i$ **then**
    Evaluate the current cascade classifier on the negative images and add misclassified samples into the negative training set;

Output:

- A cascade of classifiers for each cascade level $i = 1, \cdots$;
- Final training accuracy: F_i and D_i;

it tries to balance positive and negative errors. The final combination of these classifiers are not able to produce high detection rates without introducing many false positives. In contrast to AdaBoost, GSLDA selects the second and third weak classifier (#2, #3) based upon the maximum class separation criterion. Only the linear classifier whose outputs yield the maximum distance between two classes is selected. As a result, the selected linear classifiers introduce much less number of false positives (Fig. 2).

Viola and Jones [9] pointed out the limitation of AdaBoost in the context of highly skewed data distribution and proposed a new variant of AdaBoost called AsymBoost which is experimentally shown to give a performance improvement over conventional boosting. In brief, the sample weights are updated before each round of boosting with the extra exponential term which causes the algorithm to gradually pay more attention to positive samples in each round of boosting. Our scheme based upon LDA's class-separability can be considered as an alternative classifier to AsymBoost that also takes asymmetry information into consideration.

C. BGSLDA

Before we introduce the concept of BGSLDA, we present a brief explanation of boosting algorithms. Boosting is one of the most popular learning algorithms. It was originally designed for classification problems. It combines the output of many weak classifiers to produce a single strong learner. A weak classifier is defined as a classifier with classification accuracy on the training set greater than random guessing. There exist many variants of boosting algorithms, e.g., AdaBoost (minimizing the exponential loss), GentleBoost (fitting regression function by weighted least square methods), LogitBoost (minimizing the logistic regression cost function) [25], LPBoost (minimizing the hinge loss) [26], [27], etc. All of them rely on sample reweighting and weighted majority voting. One of the widely used boosting algorithm is AdaBoost [28]. AdaBoost is a greedy algorithm that constructs an additive combination of weak classifiers such that the exponential loss

$$L(y, F(\boldsymbol{x})) = \exp(-yF(\boldsymbol{x}))$$

is minimized. Here $\boldsymbol{x}$ is the labeled training examples and y is its label $y \in \{-1, +1\}$; $F(\boldsymbol{x})$ is the final decision function where

its sign predicts the class label. Each training sample receives a weight u_i that determines its significance for training the next weak classifier. In each boosting iteration, the value of weak classifier's weight coefficients, α_t, is computed and the sample weights are updated according to the exponential rule (7). AdaBoost then selects a new hypothesis $h(\cdot)$ that best classifies updated training samples with minimal weighted classification error e. The final decision rule $H(\cdot)$ is a sign of the linear combination of the selected weak classifiers weighted by their coefficients α_t. The classifier decision is given by

$$H(\boldsymbol{x}) = \mathbf{sign}(F(\boldsymbol{x})) = \operatorname{sign}\left(\sum_{t=1}^{N_w} \alpha_t h_t(\boldsymbol{x})\right)$$

where α_t is a weight coefficient; $h_t(\cdot)$ is a weak learner and N_w is the number of weak classifiers.

In the previous section, we have introduced the concept of GSLDA in the domain of object detection. However, decision stumps used in GSLDA algorithm are learned *only once* to save computation time. In other words, once learned, an optimal threshold, which gives smallest classification error on the training set, remains unchanged during GSLDA training. This speeds up the training process as also shown in forward feature selection of [11]. However, it limits the number of decision stumps available for the GSLDA classifier to choose from. As a result, the GSLDA algorithm fails to perform at its best. In order to achieve the best performance from the GSLDA classifier, we propose to extend decision stumps used in GSLDA training with sample reweighting techniques used in boosting methods. In other words, each training sample receives a weight and the new set of decision stumps are trained according to these sample weights. We call the new classifier Boosted GSLDA (in short, BGSLDA). The BGSLDA cascade learning algorithm is shown in Algorithm 2. Since the BGSLDA based object detection framework has the same input/output as GSLDA based detection framework (Algorithm 1), we replace lines 2–10 in Algorithm 1 with Algorithm 2.

In Algorithm 2, the criterion used to select the best decision stump is similar to the one applied in step (1) in Algorithm 1. Step (3) in Algorithm 2 is introduced in order to speed up the GSLDA training process. By saying that, we remove decision stumps with a weighted error larger than $e_k + \varepsilon$ where $e_k = (1)/(2) - (1)/(2)\beta_k$, ε is an arbitrarily small constant, $\beta_k = \max\left(\sum_{i=1}^{N} u_i y_i h_t(x_i)\right)$, N is the number of samples, y_i is the class label of sample x_i and $h_t(x_i)$ is the prediction of the training data x_i using weak classifier h_t. The condition used here has connection with the dual constraint of the soft margin LPBoost [26]. The dual objective of LPBoost minimizes β subject to the constraints

$$\sum_{i=1}^{N} u_i y_i h_t(x_i) \leq \beta, \quad \forall t,$$

and

$$\sum_{i=1}^{N} u_i = 1, 0 \leq u_i \leq \text{const}, \quad \forall i.$$

As a result, the sample weights u_i is the most pessimistic one. We choose decision stumps with a weighted error smaller than $e_k + \varepsilon$. These decision stumps are the ones that perform best under the most pessimistic condition.

Given the set of decision stumps, GSLDA selects the stump that results in maximum class separation [step (4)]. The sample weights can be updated using different boosting algorithms [step (5)]. In our experiments, we use AdaBoost [1] reweighting scheme (BGSLDA - scheme 1)

$$u_i^{(t+1)} = \frac{u_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z^{(t+1)}} \tag{7}$$

with

$$Z^{(t+1)} = \sum_i u_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)).$$

Here $\alpha_t = 0.5\log((1-e_t)/(e_t))$, e_t is the weighted error, $Z^{(t+1)}$ is a normalization factor chosen such that u_i will be a probability distribution. We also use AsymBoost [9] reweighting scheme (BGSLDA—scheme 2)

$$u_i^{(t+1)} = \frac{u_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)) \exp(y_i \log\sqrt{k})}{Z^{(t+1)}} \tag{8}$$

with

$$Z^{(t+1)} = \sum_i u_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)) \exp(y_i \log\sqrt{k}).$$

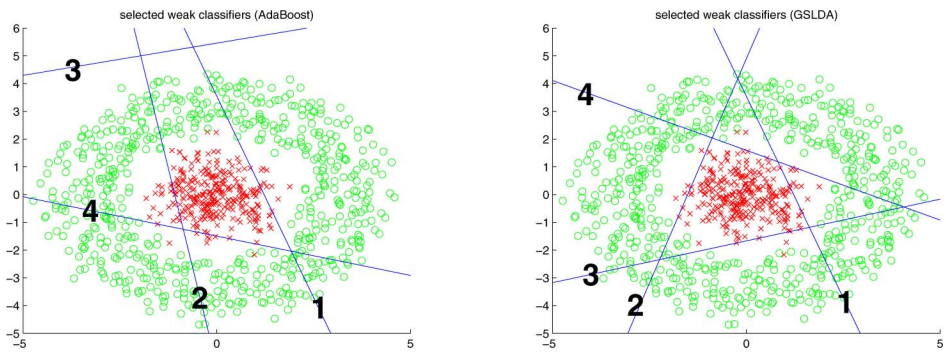
D. Training Time Complexity of BGSLDA

In order to analyze the complexity of the proposed system, we need to analyze the complexity of boosting and GSLDA training. Let the number of training samples in each cascade layer be N. For boosting, finding the optimal threshold of each feature needs $O(N\log N)$. Assume that the size of the feature set is M and the number of weak classifiers to be selected is T. The time complexity for training boosting classifier is $O(MTN\log N)$. The time complexity for GSLDA forward pass is $O(NMT + MT^3)$. $O(N)$ is the time complexity for finding mean and variance of each features. $O(T^2)$ is the time complexity for calculating correlation for each feature. Since, we have M features and the number of weak classifiers to be selected is T, the total time complexity for GSLDA is $O(NMT + MT^3)$. Hence, the total time complexity is $O(\underbrace{MTN\log N}_{\text{weak classifier}} + \underbrace{NMT + MT^3}_{\text{GSLDA}})$. When $N \gg T$, most of the computation is spent on training weak classifiers. On the other hand, when T is large, most of the computation time is spent on GSLDA calculation (finding the feature that maximizes class-separability criterion). For cascaded structure, The value of T can be set to be small, i.e., the maximum number of weak classifiers in each cascade node. For face detection using Haar-like features with cascade classifiers [1], N is 4916, M is 160 000 (24×24 pixels patch) and T is usually less than 200.

For fast AdaBoost training of Haar-like rectangle features, we apply the precomputing technique similar to [11].

III. Experiments

This section is organized as follows. The data sets used in this experiment, including how the performance is analyzed, are



(a) (b)

Fig. 2. Two examples on a toy data set. (a) AdaBoost. (b) GSLDA (forward pass). X's and O's represent positive and negative samples, respectively. Weak classifiers are plotted as lines. The number on the line indicates the order in which weak classifiers are selected. AdaBoost selects weak classifiers for attempting to balance weighted positive and negative error. Notice that AdaBoost's third weak classifier classifies all samples as negative due to the very small positive sample weights. In contrast, GSLDA selects weak classifiers based upon the maximum class separation criterion. We see that four weak classifiers of GSLDA model the positives well and most of the negative are rejected.

Algorithm 2 The training algorithm for building a cascade of BGSLDA object detector.

while $F_{\text{target}} < F_i$ **do**
  $i = i + 1$;
  $f_i = 1$;
  while $f_i > F_{\max}$ **do**
    1. Normalize sample weights $\boldsymbol{u}$;
    2. Train a weak classifier (*e.g.*, a decision stump parameterized by a threshold θ) with the smallest weighted error on the training set;
    3. Remove those weak classifiers with weighted error larger than $e_k + \varepsilon$ (section II-C);
    4. Add the best weak classifier (*e.g.*, decision stump) that yields $\lambda_{\max}(S_b, S_w)$;
    5. Update sample weights $\boldsymbol{u}$ in the AdaBoost manner (Eq. (7)) or AsymBoost manner (Eq. (8));
    6. Lower a linear classifier threshold, θ in (4), such that $D_{\min}$ holds;
    7. Update f_i using this classifier threshold;
  $D_{i+1} = D_i \times D_{\min}$;
  $F_{i+1} = F_i \times f_i$; and and remove correctly classified negative samples from the training set;
  if $F_{\text{target}} < F_i$ **then**
    Evaluate the current cascade classifier on the negative images and add misclassified samples into the negative training set;

described. Experiments and the parameters used are then discussed. Finally, experimental results and analysis of different techniques are presented.

A. Face Detection With the GSLDA Classifier

Due to its efficiency, Haar-like rectangle features [1] have become a popular choice as image features in the context of face detection. Similar to the work in [1], the weak learning algorithm known as decision stumps and Haar-like rectangle features are used here due to their simplicity and efficiency. The following experiments compare AdaBoost, FloatBoost (AdaBoost with backtrack mechanism) [10] and GSLDA learning algorithms in their performances in the domain of face detection.

1) Performances on Single-Node Classifiers: This experiment compares single strong classifier learned using AdaBoost, FloatBoost and GSLDA algorithms in their classification performance. The data sets consist of three training sets and two test sets. Each training set contains 2000 face examples and 2000 nonface examples (Table I). The data set consists of 10 000

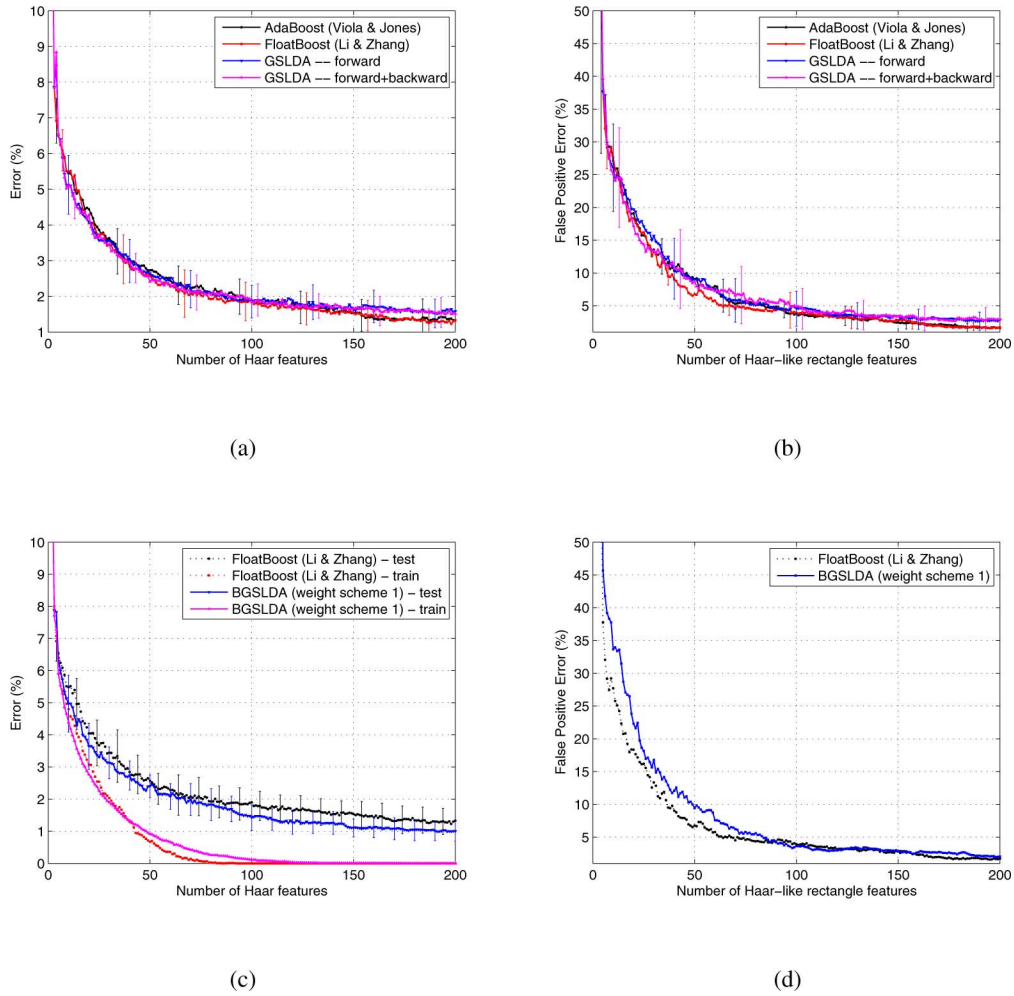


Fig. 3. See test for details (best viewed in color). (a) Comparison of test error rates between GSLDA, AdaBoost, and FloatBoost. (b) Comparison of false alarm rates on the test data set between GSLDA, AdaBoost, and FloatBoost. The detection rate on the validation face set is fixed at 99%. (c) Comparison of train and test error rates between BGSLDA (scheme 1) and FloatBoost. (d) Comparison of false alarm rates on the test set between BGSLDA (scheme 1) and FloatBoost.

TABLE I
SIZE OF TRAINING AND TEST SETS USED ON THE SINGLE NODE CLASSIFIER

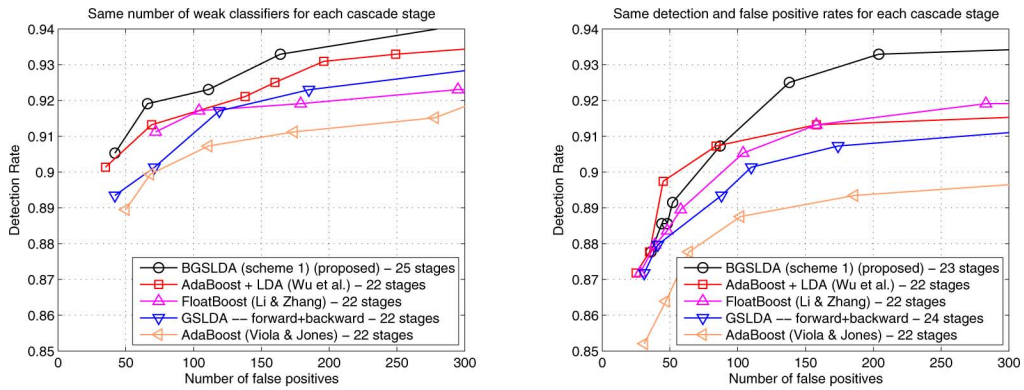
#	data splits	faces/split	non-faces/split
Train	3	2000	2000
Test	2	2000	2000

mirrored faces. The faces were cropped and rescaled to images of size 24×24 pixels. For nonface examples, we randomly selected 10 000 random nonface patches from nonface images obtained from the internet.

For each experiment, three different classifiers are generated, each by selecting two out of the three training sets and the remaining training set for validation. The performance is measured by two different curves: the test error rate and the classifier learning goal (the false alarm error rate on test set given that the detection rate on the validation set is fixed at 99%). A 95% confidence interval of the true mean error rate is given by the t-distribution. In this experiment, we test two different approaches of GSLDA: forward-pass GSLDA and dual-pass (forward+backward) GSLDA.[2]

The results are shown in Fig. 3. The following observations can be made from these curves. Having the same number of learned Haar-like rectangle features, GSLDA achieves a comparable error rate to AdaBoost or FloatBoost on test sets [Fig. 3(a)]. GSLDA seems to perform slightly better with less number of Haar-like features (less than 100) while AdaBoost and FloatBoost seems to perform slightly better with more Haar-like features (more than 100). However, all classifiers perform almost similarly within 95% confidence interval of the true error rate. This indicates that features selected using GSLDA classifiers are as meaningful as features selected using the AdaBoost and FloatBoost classifiers. From the curve, GSLDA with bidirectional search yields better results than GSLDA with forward search only. Fig. 3(b) shows the false positive error rate on the test set. From the figure, GSLDA, AdaBoost and FloatBoost achieve a comparable false positive

[2]Dual-pass GSLDA performs a backward elimination after the latest weak classifier is added by forward-pass GSLDA. The process removes those previously added weak classifiers which have little help in separating positive class from negative class.



(a) (b)

Fig. 4. Comparison of ROC curves on the MIT+CMU face test set. (a) With the same number of weak classifiers in each cascade stage on AdaBoost and its variants. (b) With 99.5% detection rate and 50% false positive rate in each cascade stage on AdaBoost and its variants. BGSLDA (scheme 1) corresponds to the GSLDA classifier with decision stumps being reweighted using the AdaBoost scheme.

error rate on the test set. Similar to [10], FloatBoost has a slightly lower error rate than AdaBoost.

2) Performances on Cascades of Strong Classifiers: In this experiment, we used 5000 mirrored faces from previous experiments. The nonface samples used in each cascade layer are collected from false positives of the previous stages of the cascade (bootstrapping). The cascade training algorithm terminates when there are not enough negative samples to bootstrap. For fair evaluation, we trained both techniques with the same number of weak classifiers in each cascade. Note that since dual-pass GSLDA (forward+backward search) yields better solutions than the forward search in the previous experiment, we use the dual-pass GSLDA classifier to train a cascade of face detectors.

We tested our face detectors on the low resolution faces data set, MIT+CMU frontal face test set. The complete set contains 130 images with 507 frontal faces. In this experiment, we set the scaling factor to 1.2 and window shifting step to 1 pixel. The technique used for merging overlapping windows is similar to [1]. Detections are considered true or false positives based upon the area of overlap with ground truth bounding boxes. To be considered a correct detection, there must be at least a 50% overlap between the predicted bounding box and ground truth bounding box. Multiple detections of the same face in an image are considered false detections.

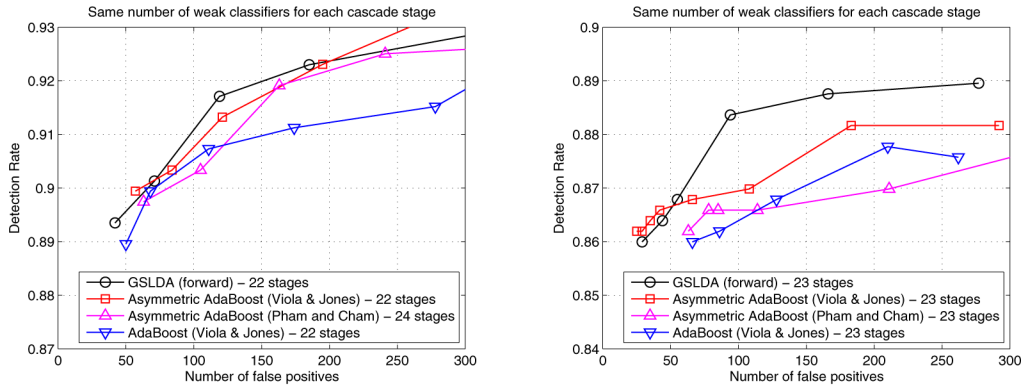
Fig. 4(a) and (b) shows a comparison between the receiver operating characteristic (ROC) curves produced by GSLDA, AdaBoost and FloatBoost. In Fig. 4(a), the number of weak classifiers in each cascade stage is predetermined while in Fig. 4(b), weak classifiers are added to the cascade until the predefined objective is met.

The ROC curves show that the GSLDA method outperforms AdaBoost at all false positive rates. The observation is that by lowering AdaBoost's threshold (in order to achieve high detection rates with moderate false positive rates), the classification performance of AdaBoost is *no longer optimal*. Our findings in this work are consistent with the experimental results presented in [9]–[11].

Wu *et al.* [11] used LDA weights instead of weak classifiers' weights provided by the AdaBoost algorithm. [10] introduced a backtrack mechanism to remove unfavorable weak classifiers (FloatBoost learning). We found the performance of AdaBoost plus LDA, FloatBoost and GSLDA to be similar. Since, we also use Haar-like features and the cascade structure introduced in [1], we can infer that the evaluation time of our GSLDA face detector is similar to that of AdaBoost face detector.

Fig. 5(a) compares the performance of LDA-based classifiers against asymmetric boosting-based classifiers [9], [29]. For [9], we set the asymmetric parameter to be 1.1 using cross-validation. For [29], we trained the classifier in an offline mode and set the asymmetric parameter to be 2 using cross-validation. The number of weak classifiers in each node was set to be the same in all classifiers. Based upon our results, the detection performance of GSLDA is similar to asymmetric AdaBoost. Since, we train each cascade node with equal number of faces and nonfaces, in our experiments, the performance of [29] is very similar to [9]. Training each node with different ratio of training faces and training nonfaces might produce different performance results between the two versions of asymmetric boosting of [29] and [9].

Our next experiment compares the performance of our classifiers on correlation coefficient features. We first calculate region covariance from several image statistics. In this experiment, we used seven statistics; namely pixel location x, pixel location y, image intensity $\mathbf{I}(x, y)$, first-order partial derivative of the intensity in horizontal and vertical direction, $|\mathbf{I}_x|$ and $|\mathbf{I}_y|$, second-order partial derivative of the intensity in horizontal and vertical direction $|\mathbf{I}_{xx}|$ and $|\mathbf{I}_{yy}|$. We then calculate correlation coefficient between any two random variables and use them to replace Haar-like features. We train weak classifiers using decision stumps and learn a cascade classifier. We set the asymmetric parameter for [9] to be 1.1 and for [29] to be 2. The experimental results are shown in Fig. 5(b). As we can see, the performance of GSLDA is similar to AsymBoost [9] when the number of false positives is less than 50. Because we only extract one statistic (correlation between a pair of random



(a) (b)

Fig. 5. Comparison of ROC curves on the MIT+CMU face test set. (a) Using Haar-like features. (b) Using correlation coefficient features.

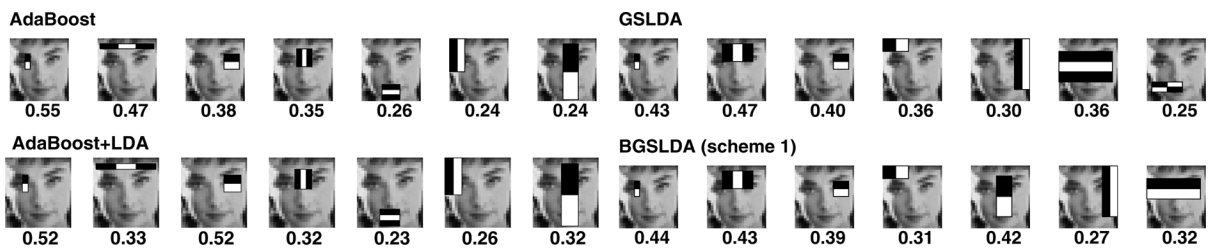


Fig. 6. First seven Haar-like rectangle features selected from the first layer of the cascade. The value below each Haar-like rectangle features indicates the normalized feature weight. AdaBoost and GSLDA have the same Haar-like rectangle features in the first layer. For AdaBoost, the value corresponds to the normalized α where α is computed from $\log((1-e_t)/e_t)$ and e_t is the weighted error. For LDA, the value corresponds to the normalized $\boldsymbol{w}$ such that for input vector $\boldsymbol{x}$ and a class label y, $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$ leads to maximum separation between two classes.

variables) from each image region, the performance is slightly inferior to Haar-like features.

Note that GSLDA not only performs better than AdaBoost but it is also much simpler. The weak classifier learning (decision stumps) is performed only once for the given set of samples (unlike AdaBoost or FloatBoost where weak classifiers have to be retrained in each boosting iteration). GSLDA sequentially selects the decision stump whose output yields the maximum eigenvalue. The process continues until the stopping criteria are met. Note that given the decision stumps selected by GSLDA, any linear classifiers can be used to calculate the weight coefficients. Based upon our experiments, using linear SVM (maximizing the minimum margin) instead of LDA also gives a very similar result to our GSLDA detector. We believe that using one objective criterion for feature selection and another criterion for classifier construction would provide a classifier with more flexibility than using the same criterion to select features and train weight coefficients. This finding was originally advocated in Wu *et al.* [11]. Our results are consistent with the experimental results reported in [11]. This finding opens up many more possibilities in combining various feature selection techniques with many existing classification techniques. We believe that a better and faster object detector can be built with careful design and experiments.

Haar-like rectangle features selected in the first cascade layer of different classifiers are shown in Fig. 6. Note that all classifiers select Haar-like features that cover the area around the eyes and forehead. Table II compares the two cascade classifiers in terms of the number of weak classifiers and the average number of Haar-like rectangle features evaluated per detection window. Comparing GSLDA with AdaBoost, GSLDA has more weak classifiers and takes longer time to evaluate than AdaBoost.

Unlike in AdaBoost, where training samples are reweighed in each boosting iteration, GSLDA does not update sample weights. In our algorithm, we only learn weak classifiers (e.g., decision stumps) with the smallest error on the training set once, i.e., for decision stump, the threshold is learned once in the beginning. Once learned, the threshold parameter remains unchanged. This is different from AdaBoost where the threshold parameter is relearned so that the weak classifier would yield minimal weighted misclassification error. Hence, the number of decision stumps available for training GSLDA is much smaller than the number of decision stumps used in training AdaBoost classifiers. In other words, AdaBoost can choose a more powerful/meaningful decision stump during each boosting iteration. Interestingly, GSLDA outperforms AdaBoost. This indicates that the classifier trained to maximize class separation might be more suitable in the domain where the distribution of positive and negative samples is highly skewed. In the next section, we conduct experiments on BGSLDA.

B. Face Detection With BGSLDA Classifiers

The following experiments compare BGSLDA and different boosting learning algorithms for face detection. BGSLDA (weight scheme 1) corresponds to GSLDA classifier with decision stumps being reweighted using the AdaBoost scheme while BGSLDA (weight scheme 2) corresponds to GSLDA with decision stumps being reweighted using the AsymBoost

TABLE II
COMPARISON OF NUMBER OF WEAK CLASSIFIERS. THE NUMBER OF CASCADE STAGES AND TOTAL WEAK CLASSIFIERS WERE OBTAINED FROM THE CLASSIFIERS TRAINED TO ACHIEVE A DETECTION RATE OF 99.5% AND THE MAXIMUM FALSE POSITIVE RATE OF 50% IN EACH CASCADE LAYER. THE AVERAGE NUMBER OF HAAR-LIKE RECTANGLES EVALUATED WAS OBTAINED FROM EVALUATING THE TRAINED CLASSIFIERS ON MIT+CMU FACE TEST SET. DUAL-PASS CLASSIFIERS (FORWARD+BACKWARD) E.G. FLOATBOOST, GSLDA (DUAL-PASS) TAKE LONGER TIME TO TRAIN THAN ONE-PASS CLASSIFIERS

Method	Training time	Number of stages	Number of weak classifiers	Average number of Haar features evaluated
AdaBoost [1]	3 hours	22	1771	23.9
FloatBoost [10]	3+ hours	22	1532	23.3
AdaBoost+LDA [11]	3 hours	22	1436	22.3
GSLDA	16+ hours	24	2985	36.0
BGSLDA (scheme 1)	16+ hours	23	1696	24.2
AsymBoost [9]	3 hours	22	1650	22.6
AsymBoost+LDA [11]	3 hours	22	1542	21.5
BGSLDA (scheme 2)	16+ hours	23	1621	24.9

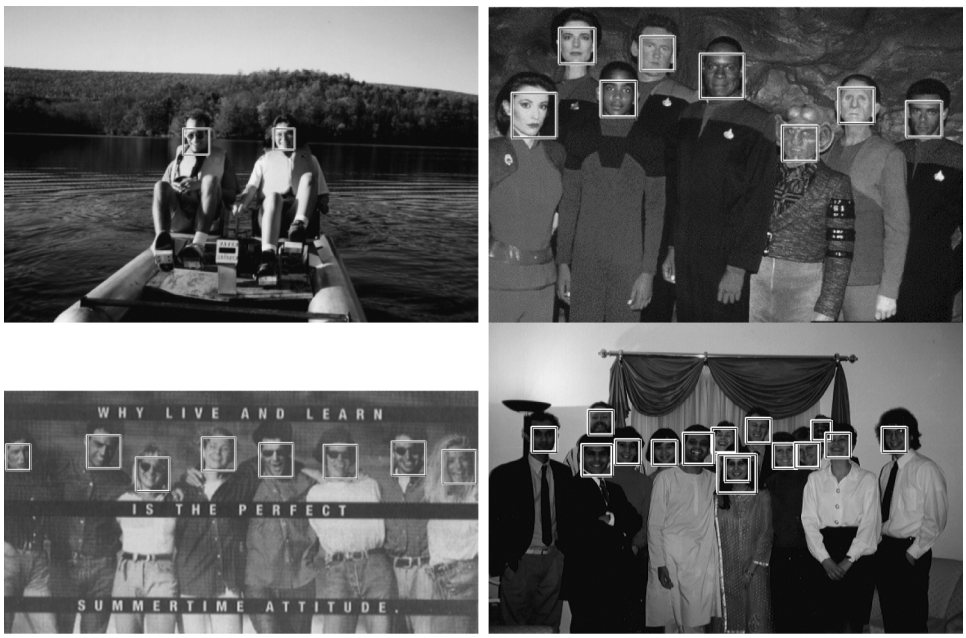
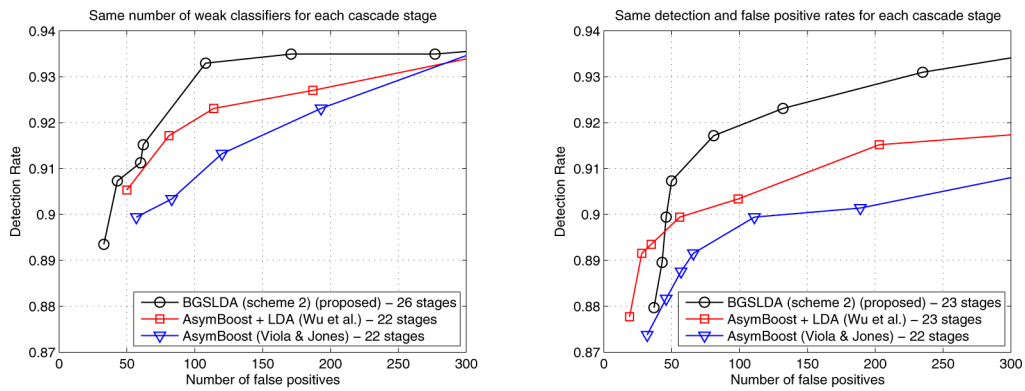


Fig. 7. Face detection examples using the BGSLDA (scheme 1) detector on the MIT+CMU test data set. We set the scaling factor to 1.2 and window shifting step to 1 pixel. The technique used for merging overlapping windows is similar to [1].

scheme (for highly skewed sample distributions). AsymBoost used in this experiment is from [9]. However, any asymmetric boosting approach can be applied here, e.g., [27], [30].

1) Performances on Single Node Classifiers: The experimental setup is similar to the one described in previous section. The results are shown in Fig. 3. The following conclusions can be made from Fig. 3(c). Given the same number of weak classifiers, BGSLDA always achieves a lower generalization error rate than FloatBoost. However, in terms of training error, FloatBoost achieves a lower training error rate than BGSLDA. This may be explained as FloatBoost has a faster convergence rate than BGSLDA. From the figure, FloatBoost only achieves lower training error rate than BGSLDA when the number of Haar-like rectangle features is larger than 50. Fig. 3(d) shows the false alarm error rate. The false positive error rates of both classifiers are very similar.

2) Performances on Cascades of Strong Classifiers: The experimental setup and evaluation techniques used here are similar to the one described in Section III-A.1. The results are shown in Fig. 4. Fig. 4(a) shows a comparison between the ROC curves produced by BGSLDA (scheme 1) and FloatBoost trained with the same number of weak classifiers in each cascade. Both ROC curves show that the BGSLDA classifier outperforms both AdaBoost, FloatBoost [10] and AdaBoost plus LDA [11]. Fig. 4(b) shows a comparison between the ROC curves of different classifiers when the number of weak classifiers in each cascade stage is no longer predetermined. At each stage, weak classifiers are added until the predefined objective is met. Again, BGSLDA significantly outperforms other evaluated classifiers. Fig. 7 demonstrates some face detection results on our BGSLDA (scheme 1) detector.



(a) (b)

Fig. 8. Comparison of ROC curves on the MIT+CMU face test set. (a) With the same number of weak classifiers in each cascade stage on AsymBoost and its variants. (b) With 99.5% detection rate and 50 false positive rate in each cascade stage on AsymBoost and its variants. BGSLDA (scheme 2) corresponds to GSLDA with decision stumps being reweighted using the AsymBoost scheme.

TABLE III
BREAKDOWN OF CPU TIME USED IN TRAINING A GSLDA CASCADE CLASSIFIER ON A STANDARD DESKTOP PC

Process	Time
Weak classifier training	1h 20m
GSLDA feature selection	12h 40m
Bootstrapping	1h 50m

In the next experiment, we compare the performance of BGSLDA (scheme 2) with other classifiers using the asymmetric weight updating rule [9]. In other words, the asymmetric multiplier $\exp(\frac{1}{N} y_i \log \sqrt{k})$ is applied to every sample before each round of weak classifier training. The results are shown in Fig. 8. Fig. 8(a) shows a comparison between the ROC curves trained with the same number of weak classifiers in each cascade stage. Fig. 8(b) shows the ROC curves trained with 99.5% detection rate and 50% false positive rate criteria. From both figures, the BGSLDA (scheme 2) classifier outperforms other classifiers evaluated. BGSLDA (scheme 2) also outperforms BGSLDA (scheme 1). This indicates that an asymmetric loss might give a better detection accuracy in object detection where the distributions of positive examples and negative examples are highly imbalanced. Note that the performance gain between BGSLDA (scheme 1) and BGSLDA (scheme 2) is small compared with the performance gain between AdaBoost and AsymBoost. Since, LDA takes the number of samples of each class into consideration when solving the optimization problem, we believe that this reduces the performance gap between BGSLDA (scheme 1) and BGSLDA (scheme 2).

Table II indicates that our BGSLDA (scheme 1) classifier evaluates at a speed comparable to the AdaBoost classifier. However, compared with AdaBoost plus LDA, the performance gain of BGSLDA comes at the slightly higher cost in evaluation time. In terms of cascade training time, on a desktop with an Intel Core™ 2 Duo CPU T7300 with 4GB RAM, the total training time is less than one day. A breakdown of GSLDA training time is given in Table III.

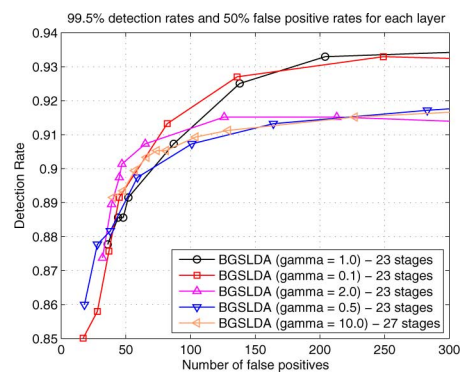


Fig. 9. Comparison of ROC curves with different values of γ in (9).

As mentioned in [31], a more general technique for generating discriminating hyper-planes is to define the total within-class covariance matrix as

$$S_w = \sum_{x_i \in C_1} (x_i - \mu_1)(x_i - \mu_1)^{\mathsf{T}} + \gamma \sum_{x_i \in C_2} (x_i - \mu_2)(x_i - \mu_2)^{\mathsf{T}}, \quad (9)$$

where μ_1 is the mean of class 1 and μ_2 is the mean of class 2. The weighting parameter γ controls the weighted classification error. We have conducted an experiment on BGSLDA (scheme 1) with different value of γ, namely $\gamma \in \{0.1, 0.5, 1.0, 2.0, 10.0\}$. All the other experiment settings remain the same as described in the previous section. The results are shown in Fig. 9. Based upon ROC curves, it can be seen that *all configurations of BGSLDA classifiers outperform AdaBoost at all false positive rates*. Setting $\gamma = 1$ gives the highest detection rates when the number of false positives is larger than 200. Setting $\gamma = 0.5$ performs best when the number of false positives is very small.

Algorithm 3 The algorithm for training multi-dimensional features.

foreach *multi-dimensional feature* **do**

1. Calculate the projection vector with LDA and project the multi-dimensional feature to 1D space;
2. Train decision stump classifiers to find an optimal threshold θ using positive and negative training set;

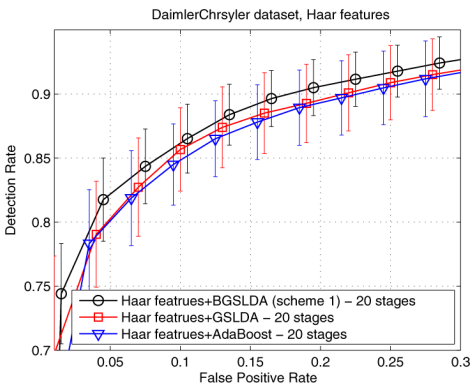


Fig. 10. Pedestrian detection performance comparison on the Daimler-Chrysler pedestrian data sets [32].

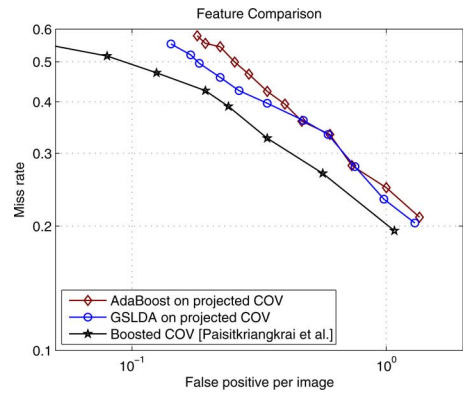


Fig. 11. Pedestrian detection performance comparison of 1-D covariance features (projected covariance) trained using AdaBoost and GSLDA on the INRIA data sets [33].

C. Pedestrian Detection With GSLDA and BGSLDA Classifiers

In this section, we apply the proposed algorithm to pedestrian detection, which is considered a more difficult problem than face detection.

1) Pedestrian Detection on the Daimler-Chrsyler Data Set With Haar-Like Features: In this experiment, we evaluate the performance of our techniques on Daimler-Chrsyler pedestrian data sets [32]. The data sets contain a set of extracted pedestrian and nonpedestrian samples which are scaled to size 18×36 pixels. The data sets consist of three training sets and two test sets. Each training set contains 4800 pedestrian examples and 5000 nonpedestrian examples. Performance on the test sets is analyzed similarly to the techniques described in [32]. For each experiment, three different classifiers are generated. Testing all three classifiers on two test sets yields six different ROC curves. A 95% confidence interval of the true mean detection rate is given by the t-distribution. We conducted three experiments using Haar-like features trained with three different classifiers: AdaBoost, GSLDA and BGSLDA (scheme 1). The experimental setup is similar to the previous experiments.

Fig. 10 shows detection results of different classifiers. Again, the ROC curves show that LDA outperforms AdaBoost at all false positive rates. Clearly these curves are consistent with those on face data sets.

2) Pedestrian Detection on the Inria Data Sets With Covariance Features: We also conduct experiments on INRIA pedestrian data sets. We compare the performance of our method with other state-of-the-art results. The INRIA data set [33] consists of one training set and one test set. The training set contains 2416 mirrored pedestrian examples and 1200 nonpedestrian images. The pedestrian samples were obtained from manually labeling images taken at various time of the days and various locations. The pedestrian samples are mostly in standing position. A border of eight pixels is added to the sample in order to preserve contour information. All samples are scaled to size 64×128 pixels. The test set contains 1176 mirrored pedestrian examples extracted from 288 images and 453 nonpedestrian test images.

Because Haar-like features perform poorly on theses data sets, we apply covariance features instead of Haar-like features [8], [34]. Note that decision stumps may not be a ideal choice for weak classifiers. To apply decision stumps to multidimensional data, one can usually select one dimension and then train a decision stump on this dimension—the information of other dimensions is totally ignored. To overcome this problem, we apply LDA that projects a multidimensional data onto a 1-D space first. In brief, we stack covariance features and project them onto 1-D space. Decision stumps are then applied as weak classifiers. Our training technique is different from [8]. [8] applied AdaBoost with weighted linear discriminant analysis (WLDA) as weak classifiers. The major drawback of [8] is a slow training time. Since, each training sample is assigned a weight, weak classifiers (WLDA) need to be trained T times, where T is the number of boosting iterations. In this experiment, we only train weak classifiers (LDA) once and store their projected result into a table. Because most of the training time in [8] is used to train WLDA, the new technique requires only $(1/T)$ training time as that of [8]. After we project the multidimensional covariance features onto a 1-D space using LDA, we train decision stumps on these 1-D features. In other words, we replace lines 4 and 5 in Algorithm 1 with Algorithm 3.

In this experiment, we generate a set of over-complete rectangular covariance filters and subsample the over-complete set in order to keep a manageable set for the training phase. The set contains approximately 45 675 covariance filters. In each stage, weak classifiers are added until the predefined objective is met. We set the minimum detection rate to be 99.5% and the maximum false positive rate to be 35% in each stage. The cascade

Fig. 12. Pedestrian detection examples on the INRIA test data sets. The classifier is trained on the INRIA training data sets.

threshold value is then adjusted such that the cascade rejects 50% negative samples on the training sets. Each stage is trained with 2416 pedestrian samples and 2500 nonpedestrian samples. The negative samples used in each stage of the cascades are collected from false positives of the previous stages of the cascades.

Fig. 11 shows a comparison of our experimental results on learning 1-D covariance features using AdaBoost and GSLDA. The ROC curve is generated by thresholding the minimum number of neighbor rectangles that makes up a single pedestrian. From the curve, GSLDA performs slightly better than AdaBoost classifiers. The results seem to be consistent with our results reported earlier on face detection. Some detection results on INRIA test data sets are shown in Fig. 12. Note that multiple scanning windows are merged using the simple technique similar to [8].

IV. Conclusion

In this work, we have proposed an alternative approach in the context of visual object detection. The core of the new framework is GSLDA [2], which aims to maximize the class-separation criterion. On various data sets for face detection and pedestrian detection, we have shown that this technique outperforms AdaBoost when the distribution of positive and negative samples is highly skewed. To further improve the detection result, we have proposed a boosted version GSLDA, which combines boosting reweighting scheme with decision stumps used for training the GSLDA algorithm. Our extensive experimental results indicate that the performance of BGSLDA is better than that of AdaBoost and AsymmBoost at a similar computation cost.

Future work will focus on the search for more efficient weak classifiers and online updating the learned model.

References

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

[2] B. Moghaddam, Y. Weiss, and S. Avidan, "Fast pixel/part selection with sparse eigenvectors," in *Proc. IEEE Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[3] S. Nadimi and B. Bhanu, "Physical models for moving shadow and object detection in video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1079–1087, Aug. 2004.

[4] B. Ni, A. A. Kassim, and S. Winkler, "A hybrid framework for 3D human motion tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1075–1084, Aug. 2008.

[5] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake, "Computationally efficient face detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Vancouver, Canada, 2001, pp. 695–700.

[6] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.

[7] C. Shen, S. Paisitkriangkrai, and J. Zhang, "Face detection from few training examples," in *Proc. IEEE Int. Conf. Image Process.*, San Diego, CA, 2008, pp. 2764–2767.

[8] S. Paisitkriangkrai, C. Shen, and J. Zhang, "Fast pedestrian detection using a cascade of boosted covariance features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1140–1151, Aug 2008.

[9] P. Viola and M. J. Jones, "Fast and robust classification using asymmetric AdaBoost and a detector cascade," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, 2002, pp. 1311–1318.

[10] 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, Sep. 2004.

[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, Mar. 2008.

[12] M. T. Pham and T. J. Cham, "Fast training and selection of Haar features using statistics in boosting-based face detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, 2007, pp. 1–7.

[13] C. Liu and H.-Y. Shum, "Kullback-Leibler boosting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 587–594.

[14] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, 2005, vol. 2, pp. 236–243.

[15] M. T. Pham, V. D. D. Hoang, and T. J. Cham, "Detection with multi-exit asymmetric boosting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, 2008.

[16] S. Paisitkriangkrai, C. Shen, and J. Zhang, "Efficiently training a better visual detector with sparse eigenvectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, 2009, pp. 1129–1136.

[17] 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.

[18] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.* Feb. 25, 2010, 10.1109/TPAMI.2010.47 .

[19] C. Shen and H. Li, "Boosting through optimization of margin distributions," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 659–666, Apr. 2010.

[20] A. R. Webb and D. Lowe, "The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis," *Neural Netw.*, vol. 3, no. 4, pp. 367–375, 1990.

[21] B. Moghaddam, Y. Weiss, and S. Avidan, "Generalized spectral bounds for sparse LDA," in *Proc. Int. Conf. Mach. Learn.*, Pittsburgh, PA, 2006, pp. 641–648.

[22] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

[23] A. Destrero, C. De Mol, F. Odone, and A. Verri, "A sparsity-enforcing method for learning face features," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 188–201, Jan. 2009.

[24] J. Sun, J. M. Rehg, and A. Bobick, "Automatic cascade training with perturbation bias," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Washington, DC, 2004, vol. 2, pp. 276–283.

[25] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[26] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1–3, pp. 225–254, 2002.

[27] J. Leskovec, "Linear programming boosting for uneven datasets," in *Proc. Int. Conf. Mach. Learn.*, Washington, DC, 2003, pp. 456–463.

[28] R. E. Schapire, "Theoretical views of boosting and applications," in *Proc. Int. Conf. Alg. Learn. Theory*, Tokyo, Japan, 1999, pp. 13–25.

[29] M.-T. Pham and T.-J. Cham, "Online learning asymmetric boosted classifiers for object detection," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, Minneapolis, MN, 2007, pp. 1–8.

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

[31] T. Cooke and M. Peake, "The optimal classification using a linear discriminant for two point classes having known mean and covariance," *J. Multivariate Anal.*, vol. 82, no. 2, pp. 379–394, 2002.

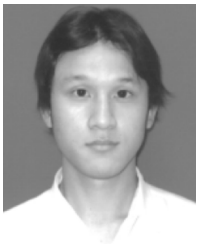
[32] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1863–1868, Nov. 2006.

[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, 2005, vol. 1, pp. 886–893.

[34] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1713–1727, Oct. 2008.

Chunhua Shen received the Ph.D. degree from the School of Computer Science, University of Adelaide, Australia, in 2006.

Since October 2005, he has been a Researcher with the Computer Vision Program, National ICT Australia (NICTA), Canberra Research Laboratory, where he is currently a Senior Researcher. He is also an adjunct Research Fellow at the Australian National University. His main research interests include statistical machine learning and its applications in computer vision and image processing.



Sakrapee Paisitkriangkrai received the B.E. degree in computer engineering, the M.E. degree in biomedical engineering, and the Ph.D. degree from the University of New South Wales, Sydney, Australia, in 2003, and 2010, respectively.

His research interests include pattern recognition, image processing, and machine learning.

Jian Zhang (M'98–SM'04) received the Ph.D. degree in electrical engineering from the University College, University of New South Wales, Australian Defence Force Academy, Australia, in 1997.

He is currently a Principal Researcher at NICTA, Sydney, Australia. He is also a conjoint Associate Professor at the University of New South Wales. His research interests include image/video processing, video surveillance and multimedia content management.

Dr. Zhang is currently an associate editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the *Journal on Image and Video Processing*.