Linear Asymmetric Classifier for Cascade Detectors

Jianxin Wu WUJX@CC.GATECH.EDU
Matthew D. Mullin MDMULLIN@CC.GATECH.EDU
James M. Rehg REHG@CC.GATECH.EDU
GVU Center, Georgia Institute of Technology, GA 30332 USA

Abstract

The detection of faces in images is fundamentally a rare event detection problem. Cascade classifiers provide an efficient computational solution, by leveraging the asymmetry in the distribution of faces vs. non-faces. Training a cascade classifier in turn requires a solution for the following subproblems: Design a classifier for each node in the cascade with very high detection rate but only moderate false positive rate. While there are a few strategies in the literature for indirectly addressing this asymmetric node learning goal, none of them are based on a satisfactory theoretical framework. We present a mathematical characterization of the node-learning problem and describe an effective closed form approximation to the optimal solution, which we call the Linear Asymmetric Classifier (LAC). We first use AdaBoost or AsymBoost to select features, and use LAC to learn a linear discriminant function to achieve the node learning goal. Experimental results on face detection show that LAC can improve the detection performance in comparison to standard methods. We also show that Fisher Discriminant Analysis on the features selected by AdaBoost yields better performance than AdaBoost itself.

1. Introduction

When classification methods are applied to computer vision problems, asymmetries are usually inherent in the classification task. As a canonical example of successful applications of visual classification methods, object detection/recognition contains many asymmetries. For example, in face detection, the positive training set contains only faces, but the negative training set should include images of millions of different scenes that may appear in the background. In order to distinguish faces from all other objects, it is necessary to collect millions of negative training samples but only thousands of face images. This is due to the rare event nature of the problem. When designing the classifier, the learning goals in the two classes are asymmetric, too. We may allow 5% missed faces. However, in order to dodge the flood of negative samples, the false positive rate must be very low, e.g. 10^{-7}. These asymmetries are difficult because conventional classification techniques (such as AdaBoost (Schapire et al., 1998)) are designed for the symmetric case.

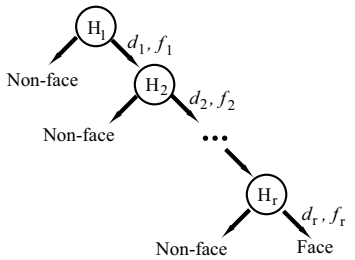


Figure 1. Illustration of a cascade with r nodes.

One approach to such asymmetric problems is to use a stage-wise rejection procedure (Baker & Nayar, 1996; Elad et al., 2002; Fleuret & Geman, 2001; Heisele et al., 2001; Romdhani et al., 2001; Viola & Jones, 2004). Viola and Jones (2004) built a cascade classifier that detected faces at video rate. Instead of designing a complex monolithic classifier, a cascade of simpler classifiers was used. In the cascade architecture illustrated in figure 1, an input patch is classified as face only if it passes the tests in all the nodes. The early nodes quickly reject most of the input patches and the testing speed is greatly improved. In the cascade framework, each node only has to deal with part of the negative training data, which addresses the inherent asymmetry in the two sample populations.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

Assuming that different nodes make independent errors, the detection and false positive rates of a cascade are

$$D = \prod_{i=1}^{r} d_i, F = \prod_{i=1}^{r} f_i \quad (1)$$

where d_i and f_i are the detection and false positive rates of the i-th node, respectively. Eq. (1) suggests the following **node learning goal**: *For every node, design a classifier H with very high (e.g. 99.9%) detection rate and only moderate (e.g. 50%) false positive rate.* Within this learning goal, we no longer need to use an imbalanced data set.

In (Viola & Jones, 2004), the node classifier H is an AdaBoost classifier. However, AdaBoost minimizes the symmetric error rate, which is not guaranteed to achieve the node learning goal. Viola and Jones (2002) proposed an intuitive remedy for this problem. Their AsymBoost method assigns larger costs to false negatives than false positives. Although AsymBoost has better performance than AdaBoost, it addresses the node learning goal indirectly and is still not guaranteed to optimize it. The node learning goal was also implicitly addressed in many other stage-wise rejection methods.

In this paper, we explicitly express the node learning goal as a constrained optimization problem and solve it directly. In general there is no closed-form solution. We show that under some reasonable simplifying assumptions, we can derive a closed-form approximation to the optimal solution. We call the resulting classifier a Linear Asymmetric Classifier (LAC). We first use AdaBoost or AsymBoost to select features, and use LAC to learn a linear discriminant function to achieve the node learning goal. Experimental results on both synthetic data set and real world object detection tasks show that LAC can improve the detection performance. We also demonstrate that applying Fisher Discriminant Analysis (FDA) to the weak classifiers selected by AdaBoost can yield better performance than the usual AdaBoost ensemble.

The rest of this paper is organized as follows. In section 2 we define the node learning goal mathematically and derive the LAC. Section 3 describes using AdaBoost as a feature selection method so that LAC can be applied to linearly non-separable problems. We discuss the relationship between the proposed LAC method to other methods in section 4. Experimental results on synthetic data set and face detection tasks are presented in section 5. Section 6 concludes the paper.

2. The Linear Asymmetric Classifier

We first formalize the node learning goal. Let $\mathbf{x} \sim (\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ denote that $\mathbf{x}$ is drawn from a distribution with mean $\bar{\mathbf{x}}$ and covariance matrix $\Sigma_{\mathbf{x}}$. Note that we do not assume any specific form of the distribution. The only assumption is that its mean vector and covariance matrix are either known *a priori* or can be estimated from samples. We are dealing with binary classification problems with two classes $\mathbf{x} \sim (\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}), \mathbf{y} \sim (\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$, which are fixed but unknown. Samples drawn from $\mathbf{x}$ are labelled as positive and samples from $\mathbf{y}$ are labelled as negative. We consider linear classifiers $H = (\mathbf{a}, b)$:

$$H(\mathbf{z}) = \begin{cases} +1 & \text{if } \mathbf{a}^T\mathbf{z} \geq b \\ -1 & \text{if } \mathbf{a}^T\mathbf{z} < b \end{cases}.$$

Then the node learning goal is expressed as

$$\max_{\alpha, \mathbf{a} \neq 0, b} \alpha \text{ s.t. } \begin{array}{c} \Pr_{\mathbf{x} \sim (\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})} \left\{ \mathbf{a}^T\mathbf{x} \geq b \right\} = \alpha \\ \Pr_{\mathbf{y} \sim (\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})} \left\{ \mathbf{a}^T\mathbf{y} \leq b \right\} = \beta \end{array}. \quad (2)$$

In general this problem has no closed-form solution. In this section, we will develop an approximate solution for it. Empirical results showed that it is effective to set $\beta = 0.5$ for all cascade nodes. Thus, we will give a closed-form (approximate) solution when $\beta = 0.5$.

Note that an AdaBoost classifier is a linear combination of weak classifiers:

$$H(\mathbf{x}) = \text{sgn}(\Sigma_{t=1}^{T} a_t h_t(\mathbf{x}) - b) = \text{sgn}(\mathbf{a}^T\mathbf{h}(\mathbf{x}) - b)$$

in which sgn is the sign function and $\mathbf{h}(\mathbf{x})$ is the vector of weak classifiers' outputs. Thus $H(\mathbf{x})$ is a linear classifier in the feature space defined by $\mathbf{h}(\mathbf{x})$. However, there is no guarantee that the $(\mathbf{a}, b)$ selected by AdaBoost will satisfy Eq. (2) for a given choice of β. We seek a linear discriminant $(\mathbf{a}, b)$ which maximizes the node learning goal in Eq. (2).

Let $\mathbf{x_a}$ denote the normalized distribution of $\mathbf{a}^T\mathbf{x}$ ($\mathbf{x}$ projected onto the direction of $\mathbf{a}$), i.e.

$$\mathbf{x_a} = \frac{\mathbf{a}^T\mathbf{x} - \mathbf{a}^T\bar{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}, \quad (3)$$

obviously we have $\mathbf{x_a} \sim (0, 1)$. Let $\Phi_{\mathbf{x,a}}$ denotes the cumulative distribution function (cdf) of $\mathbf{x_a}$, i.e.

$$\Phi_{\mathbf{x,a}}(b) = \Pr\{\mathbf{x_a} \leq b\}. \quad (4)$$

$\mathbf{y_a}$ and $\Phi_{\mathbf{y,a}}$ are defined similarly as

$$\mathbf{y_a} = \frac{\mathbf{a}^T\mathbf{y} - \mathbf{a}^T\bar{\mathbf{y}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}}, \quad \Phi_{\mathbf{y,a}}(b) = \Pr\{\mathbf{y_a} \leq b\}.$$

Using (4), the second constraint in Eq. (2) can be written as

$$\begin{aligned}\beta &= \Pr\left\{\mathbf{a}^T\mathbf{y} \le b\right\} = \Pr\left\{\frac{\mathbf{a}^T\mathbf{y}-\mathbf{a}^T\bar{\mathbf{y}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}} \le \frac{b-\mathbf{a}^T\bar{\mathbf{y}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}\right\} \\ &= \Phi_{\mathbf{y},\mathbf{a}}\left(\frac{b-\mathbf{a}^T\bar{\mathbf{y}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}\right),\end{aligned}$$

thus we have

$$b = \mathbf{a}^T\bar{\mathbf{y}} + \Phi^{-1}_{\mathbf{y},\mathbf{a}}(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}} \tag{5}$$

where $\Phi^{-1}_{\mathbf{y},\mathbf{a}}$ is the inverse function of $\Phi_{\mathbf{y},\mathbf{a}}$. Similarly, the first constraint in Eq. (2) can be written as

$$1-\alpha = \Phi_{\mathbf{x},\mathbf{a}}\left(\frac{b-\mathbf{a}^T\bar{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}\right)$$

Using Eq. (5) to eliminate b and we obtain

$$1-\alpha = \Phi_{\mathbf{x},\mathbf{a}}\left(\frac{\mathbf{a}^T(\bar{\mathbf{y}}-\bar{\mathbf{x}}) + \Phi^{-1}_{\mathbf{y},\mathbf{a}}(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}\right).$$

Thus the constrained optimization problem (2) is equivalent to

$$\min_{\mathbf{a}\neq 0} \Phi_{\mathbf{x},\mathbf{a}}\left(\frac{\mathbf{a}^T(\bar{\mathbf{y}}-\bar{\mathbf{x}}) + \Phi^{-1}_{\mathbf{y},\mathbf{a}}(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}\right). \tag{6}$$

In Eq. (6), $\Phi_{\mathbf{x},\mathbf{a}}$ and $\Phi^{-1}_{\mathbf{y},\mathbf{a}}$ depend on the distributions of $\mathbf{x}$ and $\mathbf{y}$, in addition to the projection direction $\mathbf{a}$. Because we have no knowledge of these distributions, we cannot solve Eq. (6) analytically. We need to make some approximations to simplify it.

First, let us give a bound for Φ and Φ^{-1}. Let $Z \sim (0,1)$, applying the one-tailed version of the Chebyshev inequality, for $z > 0$ we get

$$\begin{aligned}\Phi(z) &= \Pr\{Z \le z\} = 1 - \Pr\{Z \ge z\} \\ &\ge 1 - \tfrac{1}{1+z^2} = \tfrac{z^2}{1+z^2}.\end{aligned} \tag{7}$$

Since Φ^{-1} is increasing, we have

$$\Phi^{-1}(\Phi(z)) = z \ge \Phi^{-1}\left(\frac{z^2}{1+z^2}\right),$$

which gives us the following bound:

$$\Phi^{-1}(\beta) \le \kappa(\beta), \text{ where } \kappa(\beta) = \sqrt{\frac{\beta}{1-\beta}} \tag{8}$$

From the definition, it is obvious that $\mathbf{x}_\mathbf{a} \sim (0,1)$. Thus, instead of minimizing $\Phi_{\mathbf{x},\mathbf{a}}(z)$ in Eq. (6), we can instead minimize its upper bound $\kappa(z)$. Furthermore, since $\kappa(z)$ is an increasing function, it is equivalent to minimizing z. Thus, we can approximately solve Eq. (6) by solving

$$\min_{\mathbf{a}\neq 0} \frac{\mathbf{a}^T(\bar{\mathbf{y}}-\bar{\mathbf{x}}) + \Phi^{-1}_{\mathbf{y},\mathbf{a}}(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}, \tag{9}$$

or, equivalently,

$$\max_{\mathbf{a}\neq 0} \frac{\mathbf{a}^T(\bar{\mathbf{x}}-\bar{\mathbf{y}}) - \Phi^{-1}_{\mathbf{y},\mathbf{a}}(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}. \tag{10}$$

Second, we assume that the median value of the distribution $\mathbf{y}_\mathbf{a}$ is close to its mean. This assumption is true for all symmetric distributions and is reasonable for many others. Under this assumption, we have $\Phi^{-1}_{\mathbf{y},\mathbf{a}}(0.5) \approx 0$. Thus for $\beta = 0.5$ (which is used in the cascade framework), Eq. (10) can be further approximated by

$$\max_{\mathbf{a}\neq 0} \frac{\mathbf{a}^T(\bar{\mathbf{x}}-\bar{\mathbf{y}})}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}. \tag{11}$$

In object detection tasks, the object class is often compact. When $\mathbf{x}$ can indeed be modelled by a normal distribution, $\mathbf{x}_\mathbf{a} \sim N(0,1)$ and $\Phi_{\mathbf{x},\mathbf{a}}$ does not depend on either $\mathbf{x}$ or $\mathbf{a}$. In this case, Eq. (10) is exactly equivalent to equations (2) and (6). If in addition we can assume that $\mathbf{y}$ is symmetric, we have $\Phi^{-1}_{\mathbf{y},\mathbf{a}}(0.5) = 0$. Thus, Eq. (11) is exactly equivalent to the node learning goal in Eq. (2). We call the linear discriminant function determined by Eq. (11) the Linear Asymmetric Classifier (LAC) and use it in the cascade learning framework.

The form of Eq. (11) is similar to the Fisher Discriminant Analysis (FDA), which can be written as:

$$\max_{\mathbf{a}\neq 0} \frac{\mathbf{a}^T(\bar{\mathbf{x}}-\bar{\mathbf{y}})}{\sqrt{\mathbf{a}^T(\Sigma_\mathbf{x}+\Sigma_\mathbf{y})\mathbf{a}}}. \tag{12}$$

This analogy automatically gives us the solution to Eq. (11) as:

$$\mathbf{a}^* = \Sigma_\mathbf{x}^{-1}(\bar{\mathbf{x}}-\bar{\mathbf{y}}), b^* = \mathbf{a}^{*T}\bar{\mathbf{y}}, \tag{13}$$

under the assumption that $\Sigma_\mathbf{x}$ is positive definite.

3. AdaBoost as a Feature Selector

Since most visual classification tasks are not linearly separable, we need to inject some non-linearity into the linear asymmetric classifier. We used the 'rectangle features' proposed in (Viola & Jones, 2004). A rectangle feature value is the inner product of the input with a mask $\mathbf{m}_k$. However, because of the special form of the chosen masks, rectangle features can be evaluated extremely quickly using a data structure

called the 'integral image'. For details on integral images and rectangle features, please refer to (Viola & Jones, 2004). Coupled with a threshold, a rectangle feature plays the role of a weak classifier:

$$h_k(\mathbf{x}) = \text{sgn}(\mathbf{x}^T\mathbf{m}_k - \tau_k). \tag{14}$$

Viola and Jones used AdaBoost to select weak classifiers (or equivalently, rectangle features.) The vector of weak classifiers' outputs $\mathbf{h}(\mathbf{x})$ is used as features. The AdaBoost ensemble classifier is $H(\mathbf{x}) = \text{sgn}\left(\mathbf{a}^T\mathbf{h}(\mathbf{x}) - b\right)$.

We also use $\mathbf{h}(\mathbf{x})$ as features in LAC. The experimental results in (Viola & Jones, 2004) showed that AdaBoost could effectively select discriminative rectangle features. However, AdaBoost is not designed for the node learning goal. In our proposed method, we use AdaBoost to select features, but use LAC to learn the linear discriminant function $(\mathbf{a}, b)$ to optimize the node learning goal. The complete algorithm is described in table 1.

When we want to apply FDA instead of LAC, we replace Eq. (13) with the following FDA solution:

$$\mathbf{a} = \left(\Sigma_\mathbf{x} + \Sigma_\mathbf{y}\right)^{-1}\left(\bar{\mathbf{x}} - \bar{\mathbf{y}}\right). \tag{15}$$

It is tempting to use the integral feature values $\mathbf{z}^T\mathbf{m}_k - \tau_k$ directly as features in LAC or FDA. However, since $\mathbf{z}^T\mathbf{m}_t - \tau_t$ is a linear function of the input $\mathbf{z}$, $H(\mathbf{z}) = \text{sgn}\left(\sum_{t=1}^T \mathbf{a}_t(\mathbf{z}^T\mathbf{m}_t - \tau_t) - b\right)$ is still a linear discriminant function and will not work for problems that are not linearly separable. The sgn function in (14) introduces the necessary non-linearity into the features.

Both AdaBoost and LAC have the same form $H(\mathbf{z}) = \text{sgn}\left(\mathbf{a}^T\mathbf{h}(\mathbf{z}) - b\right)$ and they share the same feature vector $\mathbf{h}(\mathbf{z})$. The only difference between these two classifiers are parameters of the linear discriminant $(\mathbf{a}, b)$. In AdaBoost, a_i is chosen in step i of the AdaBoost procedure to minimize a margin-based cost function. This is a greedy procedure and a_i is never changed after its value is determined. Furthermore, AdaBoost does not take into account the fact that the two classes are asymmetric. The linear asymmetric classifier, on the contrary, is a global procedure to seek the optimal vector $\mathbf{a}$ which optimizes the asymmetric loss in Eq. (2).

Viola and Jones (2002) proposed AsymBoost to accommodate the asymmetry. In AsymBoost, the weight updating rule increases the weights of positive examples after applying the standard AdaBoost updating rule in each round. This updating strategy causes the algorithm to gradually pay more attention to positive samples. However, the resulting linear discriminant $(\mathbf{a}, b)$ is determined in the same way as ordinary AdaBoost.

4. Comparison to Previous Work

Learning imbalanced data sets has attracted many researchers recently (see (Weiss, 2004) for an overview). One common strategy is to use sampling. Various sampling methods have been used, e.g. under-sampling, over-sampling, and SMOTE (Chawla et al., 2002). Variants of existing algorithms, e.g. AdaBoos (Ting, 2000), were also proposed to deal with imbalanced data sets. However, for problems such as face detection and network intrusion detection (Fan et al., 2000), the negative class is so huge that these methods are either too computationally expensive or hard to get representative samples. Rejection-based classifiers were often used in these problems. So we pay close attention to rejection-based classifiers in this section.

There are other rejection-based methods that are similar to the form of Eq. (11) used in cascade classifiers. However, the node learning goal was not explicitly defined and solved in these methods. We will examine the relationship between the proposed LAC and other related classifiers.

By substituting Eq. (8) into Eq. (10), we get another approximation to Eq. (10):

$$\max_{\mathbf{a}\neq 0} \frac{\mathbf{a}^T\left(\bar{\mathbf{x}} - \bar{\mathbf{y}}\right) - \kappa\left(\beta\right)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}, \tag{16}$$

which is the biased minimax probability machine (BMPM) (Huang et al., 2004). Eq. (16) is a worst case lower bound of our objective function Eq. (10). The solution of a BMPM used fractional programming which is computationally expensive. The BMPM solver in (Huang et al., 2004) often takes thousands of iterations to converge, while the solution in Eq. (13) requires only a single matrix inversion.

Another related objective function comes from the Maximum Rejection Classifier (MRC) (Elad et al., 2002), which can be written:

$$\max_{\mathbf{a}\neq 0} \frac{\left(\mathbf{a}^T\bar{\mathbf{y}} - \mathbf{a}^T\bar{\mathbf{x}}\right)^2 + \mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}. \tag{17}$$

The solution of Eq. (17) requires the solution of a generalized eigenvalue problem. The intuition behind Eq. (17) is to make the overlap between the projections $\mathbf{x_a}$ and $\mathbf{y_a}$ small. The derivation of Eq. (17) in (Elad et al., 2002) treats the two classes equally. Asymmetry in the MRC framework results from the fact that

Table 1. Training the Linear Asymmetric Classifier using AdaBoost features.

1. Given a training set composed of positive examples $\{\mathbf{x}_i\}_{i=1}^{n_\mathbf{x}}$ and negative examples $\{\mathbf{y}_i\}_{i=1}^{n_\mathbf{y}}$, and a set of rectangle features. The output is a classifier with false positive rate $\beta = 0.5$.
2. Use the AdaBoost algorithm to select T weak classifiers $\mathbf{h} = (h_1, h_2, \ldots, h_T)$ where $h_i(\mathbf{z}) = \text{sgn}(\mathbf{z}^T\mathbf{m}_i - \tau_i)$.
3. Build a feature vector $\mathbf{h}(\mathbf{z}) = (h_1(\mathbf{z}), h_2(\mathbf{z}), \ldots, h_T(\mathbf{z}))$ for each training example.
4. Estimate the mean and covariance matrix for every class: $\bar{\mathbf{x}} = \frac{1}{n_\mathbf{x}}\sum_{i=1}^{n_\mathbf{x}}\mathbf{h}(\mathbf{x}_i)$, $\bar{\mathbf{y}} = \frac{1}{n_\mathbf{y}}\sum_{i=1}^{n_\mathbf{y}}\mathbf{h}(\mathbf{y}_i)$, $\Sigma_\mathbf{x} = \frac{1}{n_\mathbf{x}}\sum_{i=1}^{n_\mathbf{x}}(\mathbf{h}(\mathbf{x}_i) - \bar{\mathbf{x}})(\mathbf{h}(\mathbf{x}_i) - \bar{\mathbf{x}})^T$, $\Sigma_\mathbf{y} = \frac{1}{n_\mathbf{y}}\sum_{i=1}^{n_\mathbf{y}}(\mathbf{h}(\mathbf{y}_i) - \bar{\mathbf{y}})(\mathbf{h}(\mathbf{y}_i) - \bar{\mathbf{y}})^T$.
5. Applying Eq. (13) to get: $\mathbf{a} = \Sigma_\mathbf{x}^{-1}(\bar{\mathbf{x}} - \bar{\mathbf{y}}), b = \mathbf{a}^T\bar{\mathbf{y}}$
6. The output is a classifier $H(\mathbf{z}) = \text{sgn}\left(\mathbf{a}^T\mathbf{h}(\mathbf{z}) - b\right)$

the two classes have different prior probabilities with $P(\mathbf{x}) \ll P(\mathbf{y})$. However, the effect of the prior on $\mathbf{y}$ is reduced quickly as the stage-wise rejection process continues. After a few rejections, $P(\mathbf{x})$ is not negligible any more in comparison to $P(\mathbf{y})$. Under such conditions, Eq. (17) is not an appropriate objective function.

A final comparison can be made between LAC and FDA. FDA and LAC both have their own merits and drawbacks. We have shown that when $\mathbf{x}$ is normal, $\mathbf{y}$ is symmetric, and $\beta = 0.5$, LAC is indeed the optimal solution to the node learning goal. However, when these assumptions are broken, LAC may be suboptimal. The intuition in FDA is to maximize the (normalized) separation between the two class means. It does not minimize the error rate or the node learning goal. The advantage of FDA is that it does not have constraints – performance will be reasonably good if the class means are far apart. If we assume that $\mathbf{x}$ and $\mathbf{y}$ have equal covariance matrices, LAC is equivalent to FDA.

There are other works that are not directly related to LAC, but are related to the cascade framework. The cascade framework has been applied to other applications, e.g. wiry object detection (Carmichael & Hebert, 2003). There were also many improvements to the cascade detector. For example, alternative boosting algorithms, such as FloatBoost (Li et al., 2002), have been used to replace AdaBoost.

5. Experimental Results

We tested the performance of the linear asymmetric classifier on both a synthetic data set and a real world face detection task. In the synthetic data set, LAC is compared against BMPM, MRC and FDA. For detection of faces, the cascade framework is used. A detailed algorithm for cascade training and background data bootstrapping is given in table 2. Two feature selectors are used: AdaBoost and AsymBoost. We compare three different ways to determine the linear discriminant $(\mathbf{a}, b)$ after the features are selected. The first method is to use the weights $\mathbf{a}$ and threshold b found by AdaBoost or AsymBoost; the second method uses LAC; the third method uses FDA. We use "X+Y" to denote the methods used in experiment, e.g. AdaBoost+LAC means that the features are selected by AdaBoost and the linear discriminant function is trained by LAC. So, there are 6 different methods compared in our experiments.

Table 2. New cascade learning algorithm based on LAC.

1. Given a set of positive examples P, an initial set of negative examples N, and a database of bootstrapping negative examples D. Given a false positive rate goal β for the cascade. The output is a cascade classifier $H = (H_1, H_2, \ldots, H_r)$.
2. i=0;
3. While the current cascade's false positive rate is bigger than β
   (a) i=i+1;
   (b) Use LAC to train a classifier H_i with P and N. Add H_i to the cascade H;
   (c) $N' = \emptyset$. Run H_i on D, add false positives (those classified as positive by H_i) to N', until $|N'| = |N|$;
   (d) $N \leftarrow N'$;

5.1. Results on Synthetic Data Set

The synthetic data set was generated using the following steps. Three distributions are created as: $\mathbf{d}_i = A_i\mathbf{c}_i + \mathbf{m}_i, i = 1, 2, 3$, where $\mathbf{c}_i \sim N(0, I), \mathbf{m}_i \sim N(0, 0.1I)$, and the elements of A are drawn randomly from a uniform distribution in $[0, 1]$. The positive examples $\mathbf{x}$ are drawn from $\mathbf{X} = \mathbf{d}_1$, and negative examples $\mathbf{y}$ are drawn from $\mathbf{Y} = \mathbf{d}_2^2 - \mathbf{d}_3^2$. This choice produces a $\mathbf{Y}$ which is not symmetric, and has a reasonable overlap with $\mathbf{X}$. The training and test sets both contain 1000 samples, including 500 positive and 500

negative samples. Four linear discriminant methods are compared. In each method, we determine the projection direction $\mathbf{a}$ using the corresponding method. The threshold b is determined such that on the training set the false positive rate is 50%. For every method, the experiments are repeated 100 times. The averaged test set accuracy on both classes are reported in table 3. On the synthetic data set, LAC works the best while FDA follows closely. Two-tailed paired t-test shows that there is no significant difference between LAC and FDA. Both the difference between LAC and MRC, and the difference between LAC and BMPM are significant, at the 0.01 significance level.

Table 3. Results on synthetic data set.

Classifier	Positive Accuracy	Negative Accuracy
LAC	96.11	50.04
FDA	95.12	49.96
MRC	90.19	49.96
BMPM	87.99	50.26

In cascaded classifiers, the imbalance between classes is absorbed by the cascade structure. In each node of a cascade, balanced training sets are usually used. This is why we used a balanced training set in the above synthetic data set. We also tested the performance of these classifiers on imbalanced training set. Two extra sets of experiments were performed. The training sets still had 500 positive examples, but the negative class had 1000 and 1500 examples, respectively. The examples were drawn from the same distributions as described above. All four classifiers' performances remained approximately the same, despite of the increasing of negative training examples. Thus detailed error rates are not presented for space constraint. Under both imbalance level, LAC performed about the same as FDA, and both LAC and FDA were better than MRC and BMPM.

5.2. Results on Faces

For face detection, our training set contained 5000 example face images and 5000 initial non-face examples, all of size 24x24. We had a set of 4832 face images for validation purposes. We used approximately 2284 million non-face patches to bootstrap the non-face examples between nodes. We used 16233 features sampled uniformly from the entire set of rectangle features. For testing purposes we used the MIT+CMU frontal face test set (Rowley et al., 1998) in all experiments. Although many researchers use automatic procedures to evaluate their algorithm, we decided to manually count the missed faces and false positives.[1] When scanning a test image at different scales, the image is re-scaled repeatedly by a factor of 1.25. Post-processing is similar to (Viola & Jones, 2004).

We trained 6 different cascades, using the two feature selectors and three linear discriminant functions. Each cascade has 21 nodes, except that the AsymBoost+LAC cascade has 22 nodes. In order to make the face detector run at video speed, the first node used only 7 features. We used more features while the node index increases (the last node used 200 features).[2] In every node, the false positive rate goal β is set to be 0.5. We observed several distributions $\mathbf{x_a}$ and $\mathbf{y_a}$ for randomly generated $\mathbf{a}$. Although they are not exactly gaussian, the normal probability plot tests show that they all fit closely to normal distributions. These results showed that for the face detection task, LAC's assumptions were legitimate.

We consider two types of performance measures: node and cascade. The node performance measure is the classifiers' ability to achieve the node learning goal. Given a trained cascade, each node has an associated training set, which is generated by the bootstrapping process (refer to table 1). We collected all such training sets from the 6 trained cascades. Given one such training set, different algorithms are required to achieve the criteria in Eq. (2). Their performance is evaluated using the validation set. The node performance measure is useful because it directly compares the ability of each method to achieve the node learning goal. The cascade performance measure compares the performance of the entire cascade. The performance of a cascade depends on more than the classifier that is used to train the nodes. The background data bootstrapping step and post processing step in face detection also have significant effects on the cascades' performance. The cascade performance measure is evaluated using the MIT+CMU benchmark test set.

The node comparison results are shown in figure 2. We are able to observe the effects of using FDA or LAC to train a linear discriminant function instead of using the values provided by the AdaBoost (or AsymBoost) algorithm. From the results in figure 2, it is obvious that both FDA and LAC can greatly reduce the false negative rates. In figure 2(a), averaged over the 11 nodes shown, AdaBoost+FDA reduces the false negative rates by 31.5% compared to AdaBoost, while

[1] We found that the criterion for automatically finding detection errors in (Lienhart et al., 2002) was too loose. This criterion yielded higher detection rates and lower false positive rates than manual counting.

[2] The source code and demo video is available at `http://www.cc.gatech.edu/~wujx`.

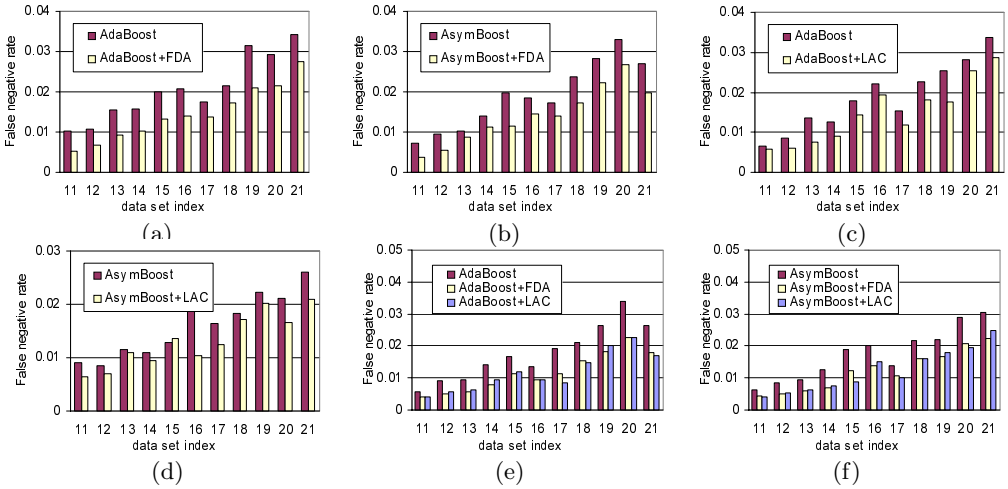


Figure 2. Experiments comparing different linear discriminant functions. The y axis shows the false negative rate when $\beta = 0.5$. In 2(a), training sets are collected from the AdaBoost+FDA cascades' node 11 to 21 (x axis shows the index of the data set). AdaBoost and AdaBoost+FDA are compared using these training sets. Similarly, 2(b)-2(f) used training sets from the AsymBoost+FDA, AdaBoost+LAC, AsymBoost+LAC, AdaBoost, and AsymBoost cascades respectively. We do not show results when the data set index is less than 11, for space constraint.

in figure 2(c) AdaBoost+LAC reduces it by 22.5%. When AsymBoost is used as the feature selector, the reductions are 27.3% and 17.3%, respectively. In figure 2(a) to 2(d), training sets came from FDA or LAC cascades. We also compare node performance when the training sets came from the AdaBoost or AsymBoost cascade. Results are shown in figure 2(e) and 2(f). Both FDA and LAC work better than the original AdaBoost and AsymBoost.

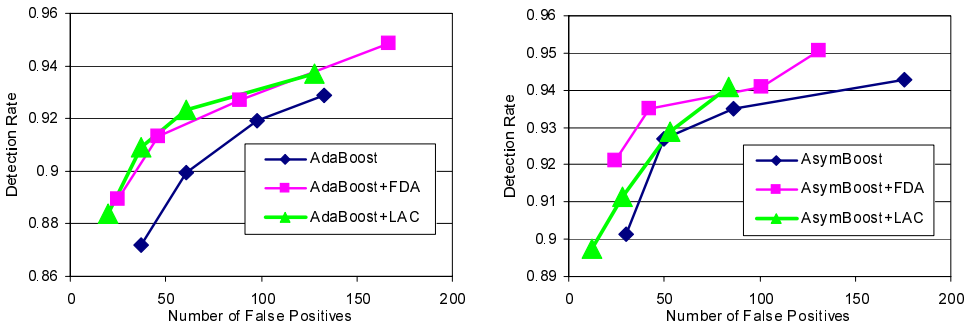
Cascade comparison results are shown in figure 3. In each figure, the marked point with fewest false positives indicates the performance of the entire cascade. We then remove the last node and the result is shown as the next marked point, etc. We do not evaluate the performance between two consecutive marked points and use linear interpolation instead. Figure 3(a) shows the results when AdaBoost is used as the feature selector and figure 3(b) shows results for AsymBoost. Figure 3 shows that both FDA and LAC have significant advantages over AdaBoost or AsymBoost. It coincides well with the node performances in figure 2. However, the error reduction effects of FDA or LAC in figure 2 is more significant than those in figure 3. We conjecture that the background data bootstrapping and post processing step remove part of the error reducing effects.

An interesting observation is that FDA works better than LAC in a few cases. LAC is derived under the assumption that for $\mathbf{y_a}$, its median value should not be far away from its mean value. We use the discrete version of AdaBoost, which means that the features can only take values 0 or 1. We conjecture that this may cause the median assumption problematic in a few cases.

6. Conclusions

The key learning problem in training cascade classifiers is to achieve the node learning goal: design a classifier with very high detection rate and only moderate false positive rate. This node learning goal was implicitly addressed in many previous works, but not based on a satisfactory theoretical framework. We present the first explicit formula of the node learning goal as a constrained optimization problem. Although in general there is no closed-form solution, we propose the Linear Asymmetric Classifier (LAC) as an approximate solution under some reasonable simplifying assumptions. LAC is also computationally efficient, only requiring a single matrix inversion. In object detection tasks, we use AdaBoost or AsymBoost as a feature selection mechanism, i.e. the weak classifiers' outputs are used as features in LAC. Experimental results show that LAC yields better performance than AdaBoost on object detection tasks. These results show that even though AdaBoost selects discriminative features, LAC can find a linear discriminant function that fulfills the node learning goal more effectively. We find that applying Fisher Discriminant Analysis on the AdaBoost features also yields better performance on asymmetric problems than AdaBoost itself.

There are some issues that we will explore in the fu-



(a) (b)

Figure 3. Experiments comparing cascade performances using the MIT+CMU benchmark test set. The x axis is the total number of false positives. The y axis is the detection rate.

ture. We currently use weak classifiers' outputs as features to inject non-linearity into LAC. Kernel methods can be readily applied to LAC as an alternative for introducing non-linearity. We will also use LAC to detect objects other than faces. Since FDA can be generalized to multi-class discriminant analysis, it is possible to extend LAC into a multi-class object detection/recognition method. Finally, LAC may be applied in other machine learning problems with highly imbalanced data sets and/or problems with asymmetric costs.

References

Baker, S., & Nayar, S. (1996). Pattern rejection. *Proc. CVPR* (pp. 544–549).

Carmichael, O., & Hebert, M. (2003). Shape-based recognition of wiry objects. *Proc. CVPR* (pp. II:401–408).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Elad, M., Hel-Or, Y., & Keshet, R. (2002). Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, *23*, 1459–1471.

Fan, W., Lee, W., Stolfo, S. J., & Miller, M. (2000). A multiple model cost-sensitive approach for intrusion detection. *Proc. 11th ECML*.

Fleuret, F., & Geman, D. (2001). Coarse-to-fine face detection. *IJCV*, *41*, 85–107.

Heisele, B., Serre, T., Mukherjee, S., & Poggio, T. (2001). Feature reduction and hierarchy of classifiers for fast object detection in video images. *Proc. CVPR* (pp. II:18–24).

Huang, K., Yang, H., King, I., & Lyu, M. R. (2004). Learning classifiers from imbalanced data based on biased minimax probability machine. *Proc. CVPR* (pp. II:558–563).

Li, S., Zhang, Z., Shum, H., & Zhang, H. (2002). FloatBoost learning for classification. *NIPS 15*. MIT Press.

Lienhart, R., Kuranov, A., & Pisarevsky, V. (2002). *Empirical analysis of detection cascades of boosted classifiers for rapid object detection* (Technical Report). MRL, Intel Labs.

Romdhani, S., Torr, P., Schoelkopf, B., & Blake, A. (2001). Computationally efficient face detection. *Proc. ICCV* (pp. 695–700).

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *20*, 23–38.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 1651–1686.

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. *Proc. ICML* (pp. 983–990).

Viola, P., & Jones, M. (2002). Fast and robust classification using asymmetric AdaBoost and a detector cascade. *NIPS 14*.

Viola, P., & Jones, M. (2004). Robust real-time face detection. *IJCV*, *57*, 137–154.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, *6*, 7–19.