

Boosting Nested Cascade Detector for Multi-View Face Detection

Chang HUANG¹, Haizhou AI¹, Bo WU¹ and Shihong LAO²

¹Computer Science and Technology Department, Tsinghua University, Beijing, 100084, China

²Sensing Technology Laboratory, Omron Corporation

E-mail: ahz@mail.tsinghua.edu.cn

Abstract

In this paper, a novel nested cascade detector for multi-view face detection is presented. This nested cascade is learned by Schapire and Singer's improved boosting algorithms that use real-valued confidence-rated weak classifiers [1], where we use confidence-rated Look-Up-Table (LUT) weak classifiers based on Haar features. Experiments show the system performance is significantly improved compared with previous methods.

1. Introduction

Multi-view face detection (MVFD) is used to detect frontal faces in images that with ± 90 -degree rotation-out-of-plane (ROP) pose changes. Since the introduction of boosted cascade face detector by Viola and Jones [2], there have been many related works, such as frontal face rotation invariant [3] and MVFD [4]. Although frontal face detection has achieved a very high performance that meet the requirement of practical applications, MVFD is still an unsolved problem for real applications.

Existing work on MVFD includes Schneiderman et al.'s work [5] based on Bayesian decision rule and Li et al.'s [4] pyramid-structured detector which is the first real-time MVFD system. In this paper, we propose a novel method for MVFD based on Schapire and Singer's improved boosting algorithm [1] that uses confidence-rated weak classifiers. It is called *Real Adaboost* in order to distinguish from *Discrete Adaboost*, that is, the original Adaboost algorithm [6] adopted in [2], which uses Boolean weak classifiers. The main contributions of this paper are: 1) A weak classifier using Look-Up-Table (LUT) of the Haar feature is proposed and Real Adaboost is employed to learn boosted classifiers. 2) A novel nested cascade detector is introduced to achieve significant higher performance compared with previous methods.

The rest of this paper is organized as follow: Section 2 presents the nested cascade detector, Section 3 introduces the MVFD, Section 4 gives experimental results, and Section 5 gives the conclusion.

2. Nested Cascade Detector

Based on Viola and Jones' cascade structure [2], we have developed a variation called nested cascade shown in Figure 1. It mainly consists of four components: Haar feature, weak classifier, strong classifier, and the novel nested classifier. The input of a weak classifier is a Haar feature value $f(\mathbf{x})$, which can be fast calculated by the integral image [2], and the output is a continuous value $h(\mathbf{x})$. A strong classifier $H(\mathbf{x})$ takes $h(\mathbf{x})$, the outputs of weak classifiers, as its inputs, linearly combines them and outputs an arbitral confidence value $conf(\mathbf{x})$, by which the positive samples can be separated from the negative ones accurately. These strong classifiers are connected one by one to form a cascade, each of which is a layer. The nested classifier functions as a weak classifier, but differs from other weak classifiers in that it takes the confidence value $conf(\mathbf{x})$ of the previous layer as its input rather than a Haar feature value. With the help of nested classifiers, more effective information for classification can be inherited by successive layers. A sample is classified as positive one if and only if it has passed every layer of the cascade.

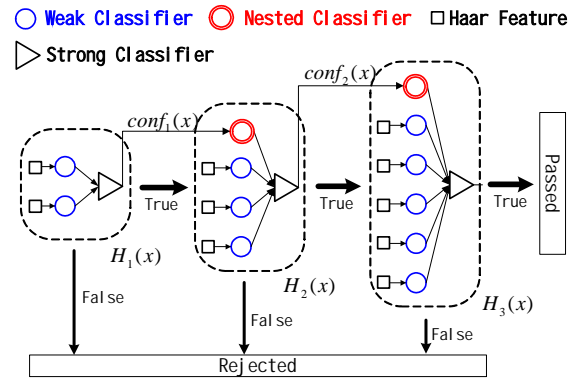


Figure 1. Nested cascade detector

There are basically three problems involved in this nested cascade detector: 1) how to train weak classifiers; 2) how to linearly combine them into a strong classifier; 3) how to make the nested classifier workable.

2.1 Learning a LUT-type Weak Classifier under Bhattacharyya Criterion

In Viola and Jones' cascade detector [2], the basic information for classification is from the Haar feature value $f(\mathbf{x})$, in which a weak classifier is defined as a

threshold-type function with Boolean-valued output $h(\mathbf{x})$ as Figure 2(a), indicating whether \mathbf{x} is positive or negative. The formal expression of a threshold-type function is $h(\mathbf{x}) = \text{sign}[f(\mathbf{x})-b]$, where b is the threshold. However, in-class differences of samples are ignored, that is to say, samples \mathbf{x}_i with different $f(\mathbf{x}_i)$ may have the same $h(\mathbf{x}_i)$. Further more, it should be noted that the threshold-type function is an optimum decision only when the positive and negative samples are linearly separable or follow a single-peak distribution such as Gaussian, otherwise it is rather a weak decision in complicated situations such as multi-Gaussian.

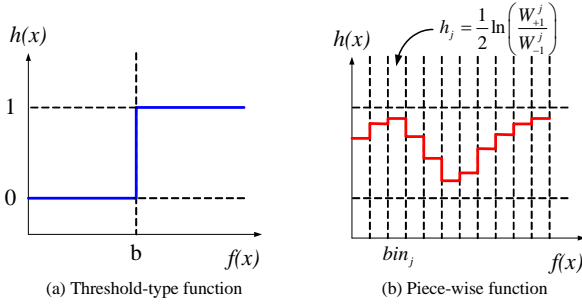


Figure 2. Two types of weak classifiers

By expanding Boolean value to real value and dividing feature space into many sub-regions with equal width, we develop a piece-wise function to approximate complex distribution of training samples (see Figure 2(b)). If $f(\mathbf{x})$ has been normalized to $[0, 1]$, the range is divided into n sub-regions: $\text{bin}_j = [j/n, (j+1)/n)$, $j = 0, \dots, n-1$. For simplicity, all samples falling into sub-region j will get the same output $h^{(j)}$.

Given the characteristic function

$$B_n^{(j)}(u) = \begin{cases} 1 & u \in [j/n, (j+1)/n) \\ 0 & u \notin [j/n, (j+1)/n) \end{cases}, j = 0, \dots, n-1.$$

The piece-wise function is formally expressed as

$$h(\mathbf{x}) = \sum_{j=0}^{n-1} h^{(j)} B_n^{(j)}(f(\mathbf{x})). \quad (1)$$

Spontaneously, $h(\mathbf{x})$ is implemented with the Look-Up Table (LUT) method. Then two consequent questions should be answered: how to find an appropriate Haar feature and how to calculate each $h^{(j)}$ stored in a LUT. The first one is solved under the Bhattacharyya criterion defined below, and the second is left to Section 2.2.

The Bhattacharyya distance [7] is an important criterion characterising the separability of two random variables based on their probabilistic distributions. It is defined as:

$$J_B = -\ln \int \sqrt{p(x|\omega_{+1})p(x|\omega_{-1})} dx \quad (2)$$

In our problem, $p(x|\omega_{+1})$ stands for the posterior probability of positive samples, while $p(x|\omega_{-1})$ stands for that of negative ones. The training error is hold on by an

upper bound $P_e \leq \sqrt{P(\omega_{+1})P(\omega_{-1})} \exp(-J_B)$, where P_e is the probabilistic error [7].

Within the LUT method, the Bhattacharyya distance takes a discrete form as

$$J_B = -\ln \sum_{j=0}^{n-1} \sqrt{p(f(\mathbf{x}) \in \text{bin}_j | \omega_{+1})p(f(\mathbf{x}) \in \text{bin}_j | \omega_{-1})} \quad (3)$$

In the next section, we will see that the Bhattacharyya distance is the best criterion for selecting features in Real Adaboost algorithm because of the coherence between J_B and the object function of Real Adaboost.

2.2 Learning a Strong Classifier with Real Adaboost

Real Adaboost [1] linearly combines many weak hypotheses into a highly accurate one like other boosting algorithms. However, it employs real-valued confidence-rated weak classifiers rather than discrete predictors (e.g. threshold-type functions used in naive boosting [2]). The LUT-type weak classifier introduced in Section 2.1 meets the very requirement of Real Adaboost.

Given a data set $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, where $\mathbf{x} \in \mathcal{X}$ and $y_i \in \{-1, +1\}$, a Haar feature pool F and the number of weak classifiers to be selected T .

- Initialize the sample distribution $D_1(i) = 1/m$.
- For $t = 1, \dots, T$
 - Select a Haar feature $f(\mathbf{x})$ in F .
 - Train a weak classifier $h_t(\mathbf{x})$ based on $f(\mathbf{x})$.
 - Update the sample distribution

$$D_{t+1}(i) = \frac{D_t(i) \exp[-y_i h_t(\mathbf{x}_i)]}{Z_t} \quad (4)$$

where Z_t is a normalization factor.

- The final strong classifier is

$$H(\mathbf{x}) = \text{sign} \left[\sum_{t=1}^T h_t(\mathbf{x}) - b \right], \quad (5)$$

where b is a threshold whose default is zero.

- The confidence of $H(\mathbf{x})$ is defined as

$$\text{conf}(\mathbf{x}) = \sum_{t=1}^T h_t(\mathbf{x}) - b \quad (6)$$

Figure 3. The Real Adaboost algorithm

Figure 3 presents a general framework of Real Adaboost. However it tells us neither how to select a feature $f(\mathbf{x})$ nor how to train a weak classifier $h(\mathbf{x})$. As illustrated in [1], the algorithm is embodied as follow.

The object function in Real Adaboost is to greedily minimize the normalization factor Z_t in equation (4)

$$\arg \min_{h_t(\mathbf{x})} Z_t = \arg \min_{h_t(\mathbf{x})} \left(\sum_{i: h_t(\mathbf{x}_i) \neq y_i} D_t(i) \exp(-y_i h_t(\mathbf{x}_i)) \right) \quad (7)$$

$$\text{Let } W_b^{(j)} = \Pr_{i \sim D}(f(\mathbf{x}_i) \in \text{bin}_j \wedge y_i = b) \quad (8)$$

where $b = \pm 1$, denoting positive or negative. It is optimized when:

$$h^{(j)} = \arg \min_{h^{(j)}} Z_t = \frac{1}{2} \ln \left(\frac{W_{+1}^{(j)}}{W_{-1}^{(j)}} \right) \quad (9)$$

Then

$$Z_t = 2 \sum_{j=0}^{n-1} \sqrt{W_{+1}^{(j)} W_{-1}^{(j)}} \quad (10)$$

By comparing equation (3) and equation (10), under the distribution D_t , we have $\arg \min Z_t = \arg \max J_B$. Therefore, according to equation (7), the object function of Real Adaboost algorithm, the Bhattacharyya distance is in fact the best criterion for selecting features.

In conclusion, we state that in order to train a best LUT-type weak classifier under a distribution D_t , first a Haar feature $f(\mathbf{x})$ which gets the maximum J_B should be selected, and then based on this $f(\mathbf{x})$, each $h^{(j)}$ in a LUT-type weak classifier $h(\mathbf{x})$ should be calculated according to equation (9). After several rounds of iteration, selected weak classifiers are linearly combined into a strong arbitral classifier (equation 5), while the confidence value (equation 6) is inherited by the coming layers by means of the nested cascade method discussed in the next Section.

2.3 Nested Weak Classifier

The nested weak classifier as a component is used to enhance Viola et al's cascade structure [2]. We observe that in their bootstrap training process of each layer, after a strong classifier $H(\mathbf{x})$ is learned, only samples with $H(\mathbf{x})=1$ are collected to be the training data of the coming layer. In this way, the neighbouring layers in cascade are correlated. However, the correlation is rather loose since all the samples that passed the previous layer are treated the same in the coming layer due to Boolean nature of $H(\mathbf{x})$. The obvious disadvantage is that the knowledge acquired during the current training has not been more efficiently inherited by the next layer.

Practically, newly collected samples are still separated well by confidence value as Figure 4. It can be seen that after resampling the newly collected samples have a similar distribution to that before resampling. Derived from the LUT-type weak classifier introduced in Section 2.1, a novel nested weak classifier $h_{nested}(conf(\mathbf{x}))$ is defined based on $conf(\mathbf{x})$. Experimentally, $conf(\mathbf{x})$ has a much larger Bhattacharyya distance (equation 3) than a Haar feature $f(\mathbf{x})$, so the nested weak classifier is a "strong" one, which is placed as the first component in each layer except for the first layer (Figure 1).

With the help of nested weak classifiers, the confidence value is inherited by the coming layer as:

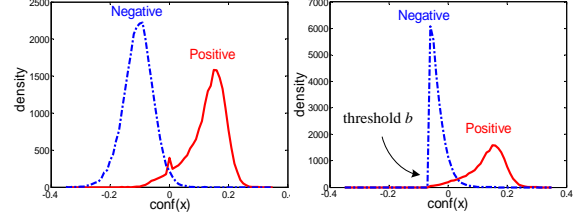
$$conf_n(\mathbf{x}) = h_{nested}(conf_{n-1}(\mathbf{x})) + \sum_{t=1}^T h_t^{(n)}(\mathbf{x}) - b \quad (11)$$

where $h_t^{(n)}(\mathbf{x})$ is the t -th weak classifier of layer n .

Meanwhile the strong classifier is redefined as:

$$H_n(\mathbf{x}) = \text{sign}(conf_n(\mathbf{x})) \quad (12)$$

The notable enhancement made by nested weak classifiers will be illustrated in Section 4.



(a) Samples before resampling (b) Samples after resampling
Figure 4. Confidence value before and after resampling

3. Multi-View Face Detector

According to the ROP angle, multi-view faces are divided into five categories: full left profile, half left profile, frontal, half right profile and full right profile. For each category, a detector is trained.

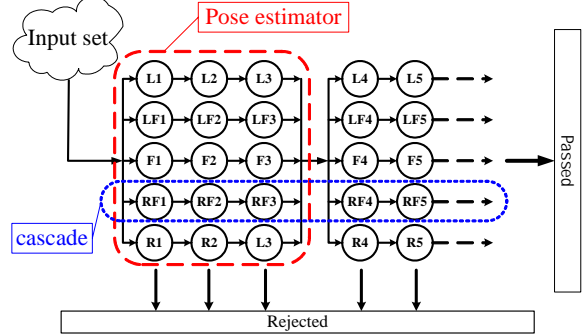


Figure 5. Pose estimator

A pose estimation (PE) technique is employed for acceleration as shown in Figure 5, in which first a few, say 3, layers of all those cascades are used for a pose estimator to select a particular cascade to continue. The pose estimation function is defined as:

$$pose(\mathbf{x}) = \arg \max_{1 \leq i \leq 5} [conf_3^{(i)}(\mathbf{x})] \quad (13)$$

where $conf_3^{(i)}(\mathbf{x})$ represents the confidence value of the third layer in the i -th cascade (equation 11).

4. Experimental Results

With about 20,000 frontal face samples, we have trained a detector that consists of 16 layers with 756 Haar features, while Viola's cascade detector [2] consists of 32 layers with 4,297 Haar features. It takes about 30 ms to process a 320×240 image for this frontal face detector on a 1.4 GHz Athlon PC. The experimental results are given in Table 1 and the ROC curve is given in Figure 6(a).

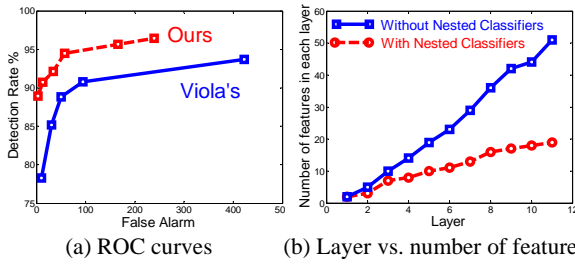


Figure 6. ROC curves & layer vs. number of features

To illustrate the role of nested weak classifiers, an experiment is done in which two cascades are trained respectively in the same conditions except that one cascade employs nested weak classifiers while the other does not. It results in a Figure 6(b) that draws the number of features in each layer of corresponding cascades. It can be seen that the cascade with nested weak classifiers uses much fewer Haar features but achieves the same accuracy as the other one. This indicates that a nested weak classifier serves equally as many Haar-feature weak classifiers as a whole so that the computation load is decreased greatly.

With about 10,000 half profile and 14,000 full profile faces samples, we have trained a half profile detector that consists of 16 layers with 1,078 Haar features and a full profile detector that consists of 16 layers with 871 Haar features respectively. Together with the above frontal face detector, a MVFD system is integrated in which a pose variation of ± 30 degrees of rotation in plane (RIP) is also considered. The experimental results on CMU profile face test set is given in Table 2, some processed images are shown in Figure 7. With PE module, the detector processes a 320×240 image in 110 ms on a 1.4GHz Athlon PC, while without PE, it increases up to 200 ms.

5. Conclusion

We propose a nested cascade structure, which is a novel variation of Viola et al.'s method [2], and train a **Table 1**. Frontal face detection results on CMU+MIT frontal face set (Totally 130 images, 507 faces)

False Alarm Method	3	10	31	34	50	57	95	166	239	422
Ours	89.0%	90.1%	-	92.1%	-	94.5%	-	95.7%	96.5%	-
Viola-Jones	-	78.3%	85.2%	-	88.8%	-	90.8%	-	-	93.7%
Rowley	-	83.2%	86.0%	-	-	-	89.2%	-	-	89.9%

Table 2. Multi-view face detection results on CMU profile face test set (Totally 208 images, 441 faces)

False Alarm Method	8	12	34	89	91	109	221	415	700
Ours									
With PE	79.4%	-	84.8%	-	-	87.8%	89.8%	-	-
Without PE	-	-	84.1%	86.2%	-	-	-	91.3%	-
Schneiderman	-	75.2%	-	-	85.5%	-	-	-	92.7%

MVFD system based on the Real Adaboost algorithm. Our system has achieved a very high performance in both accuracy and speed compared with previous published methods as far as we know. The main contributions are: 1) LUT-type weak classifiers, which are selected under Bhattacharyya criterion, are configured to approximate complicated distributions; they are boosted into strong classifiers with Real Adaboost. 2) A novel nested cascade structure is employed so that the effective information acquired during current layer learning is inherited by the next layer. In this way, a very efficient MVFD system is implemented.

6. Acknowledgements

This work is supported mainly by a grant from OMRON Corporation. It is also supported in part by National Science Foundation of China under grant No.60332010.

7. Reference

- [1] R. E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence-rated Predictions", *Machine Learning*, 37, 1999, 297-336.
- [2] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *CVPR* 2001.
- [3] Shihong Lao, Toshiyuki Kozuru, et al., "A fast 360-degree rotation invariant face detection system", in *ICCV2003 Demo program*.
- [4] S. Z. Li, L. Zhu, Z. Q. Zhang, et al., "Statistical Learning of Multi-View Face Detection". *ECCV* 2002.
- [5] H. Schneiderman and T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars". *CVPR* 2000.
- [6] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm". *The 13-th Conf. on Machine Learning*, 1996, 148-156.
- [7] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Elsevier Science, 2003, 177-179.



Figure 7. Some results on CMU profile set