

A binary decision tree implementation of a boosted strong classifier

S. Kevin Zhou

Siemens Corporate Research, Integrated Data Systems Department
755 College Road East, Princeton, NJ 08540
Email: {kzhou}@scr.siemens.com

Abstract

Viola and Jones [1] proposed the influential rapid object detection algorithm. They used AdaBoost to select from a large pool a set of simple features and constructed a strong classifier of the form $\{\sum_j \alpha_j h_j(x) \geq \theta\}$ where each $h_j(x)$ is a binary weak classifier based on a simple feature. In this paper, we construct, using statistical detection theory, a binary decision tree from the strong classifier of the above form. Each node of the decision tree is just a weak classifier and the knowledge of the coefficients α_j is no longer needed. Also, the binary tree has a lot of early exits. As a result, we achieve an automatic speedup that always makes the rapid Viola and Jones algorithm rapider.

1 Introduction

Viola and Jones [1] proposed the influential rapid object detection algorithm. There are three key contributions of the Viola and Jones algorithm. The first is to use the integral image that enables fast computation of Harr-like simple features [2]. The second is to invoke the AdaBoost algorithm [3] to select a small set of crucial features from a large set. The final contribution is to train a cascade of strong classifiers that eliminates negative examples as quickly as possible.

A strong classifier is constructed based on weak classifiers. A weak classifier $h_j(x)$, consisting of a simple feature $f_j(x)$, a threshold θ_j , and a parity p_j indicating the direction of the inequality sign, produces a binary decision

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The final strong classifier is $h(x)$ of the following form

$$h(x) = \begin{cases} 1 & \text{if } g(x) = \sum_{j=1}^J \alpha_j h_j(x) \geq \frac{1}{2} \sum_{j=1}^J \alpha_j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Following the spirit of the Viola and Jones algorithm [1], many variants and extensions have been proposed in the literature. Naturally, the Viola and Jones algorithm [1] can be modified in each of the three contributions.

- *Modifying the feature set.* Since the work of Viola and Jones [1], various new types of Harr-like have been proposed: diagonal features [4, 5], rotated features [6], center-surrounded features [6]. In the article of Zhang *et al.* [7], the Gabor wavelets features were used for face recognition. Lienhart *et al.* [8] even used small CART features.
- *Modifying the AdaBoost algorithm.* In Voila and Jones [9], the asymmetric AdaBoosting algorithm was proposed to handle the unbalanced nature of the positive and negative training sets. In the work by Li and Zhang [10], the FloatBoost algorithm was used as a substitute of AdaBoost to achieve an even lower classification error. The effect of three versions of AdaBoost (Discrete AdaBoost, Real AdaBoost, Gentle AdaBoost) were compared by Lienhart *et al.* in [8].
- *Modifying the cascade structure.* The cascade structure can be regarded as a degenerate decision tree. Full decision tree was trained by Li and Zhang [10] to perform multiview face detection more efficiently. A similar detection tree approach were found in the work of Lienhart *et al.* [11]. Also, in [12], Sun *et al.* trained the cascade while taking in account perturbation bias.

However, the core idea of using the strong classifier that consists of weak classifiers remains intact. In this paper, we examine the strong classifier in detail. In particular, we investigate the computational speed of the strong classifiers themselves, assuming that they have been trained by some means.

1.1 A general form of the strong classifier

We focus on a very general form of the strong classifier that arises from the AdaBoost algorithm. It is worth emphasizing that our analysis is not only applicable to the Viola and Jones detection algorithm, and can be used in a wide range of applications where the AdaBoost algorithm is used.

- The weak classifier can be of an arbitrary form.

$$h_j(x) = \begin{cases} T & \text{if certain condition holds} \\ F & \text{otherwise} \end{cases} \quad (3)$$

In the above, the weak classifier produces a binary decision of true (+1) or false (0).

- The strong classifier threshold is adjustable for various purposes [1]. We denote the arbitrary threshold as θ . As mentioned above, usually the strong classifier threshold obtained from an AdaBoost algorithm is set to $\frac{1}{2} \sum_{j=1}^J \alpha_j$.
- Because of the additive nature of the weighted sum $g(x) = \sum_{j=1}^J \alpha_j h_j(x)$, without loss of generality, we can always pre-sort the weights α_i such that $\alpha_1 > \dots > \alpha_J > 0$. We will show that the tie case such as $\alpha_j = \alpha_{j+1}$ is a special case of our treatment and can be easily handled by merging two cases. The fact that $\alpha_j > 0$ arises from the AdaBoost algorithm. According to the AdaBoost algorithm [1], $\alpha_j = \log \frac{1-e_j}{e_j}$ where e_j is the probability of error. Usually, e_j is smaller than 1/2 and hence $\alpha_j > 0$. If $e_j \geq 1/2$, there is no need for further boosting because adding this feature only degrades the final performance.

Now, the strong classifier has a very general form

$$h(x) = \begin{cases} T & \text{if } g(x) = \sum_{j=1}^J \alpha_j h_j(x) \geq \theta \\ F & \text{otherwise} \end{cases} \quad (4)$$

A further generalization is to allow the functions $h_j(x)$ to output discrete values (not necessarily just a binary decision of true and false). However, this is not considered here.

1.2 Proposed approach

Calling upon the classical detection theory [13, 14], we will show that the strong classifier of the above form (4) is equivalent to a binary decision tree. To be more exact, given a strong classifier, we are able to construct a binary decision tree that has the same detection and false alarm rates. To have an intuition how it works, by simply treating the weak classifiers $h_j(x)$ as boolean inputs, one can arrive at a boolean output for $h(x)$. In other words, we can easily construct a lookup table for a strong classifier. It turns out that the lookup table can be summarized by a binary decision tree.

The binary decision tree possesses three properties. First, each node of the binary decision tree is just a weak classifier and its binary decision guides what to proceed. The second property is that, once the tree is constructed, the knowledge of the coefficients α_j can be thrown away. So, there is no need to compute multiplications like $\alpha_j h_j(x)$ and sum up the terms $\alpha_j h_j(x)$. Third, the binary tree has a lot of early exits, which implies that only the weak classifiers before exit need evaluation. The above three properties guarantee an automatic speedup that always makes the *rapid* Viola and Jones algorithm even *rapider*.

The binary decision tree corresponding to the strong classifier should not be confused with the cascade of the strong classifiers since the latter is also treated as a degenerate binary decision tree with each node being a strong classifier. Our intention is to replace the strong classifier at each stage of the cascade with a binary decision tree, not the cascade structure. After replacing the strong classifiers at all stages, the cascade of strong classifiers can be regarded as a true binary decision tree with each node being a weak classifier. Also, the binary decision tree structure derived from the strong classifier does not prevent us from using a decision tree for the weak classifier as in [15]. In this case, we have a true decision tree corresponding to the boosted strong classifier.

In [16], Grossmann directly learned a decision tree that can be analyzed in the framework of Adaboost. He also observed that the tree structure reduces computational cost. We here propose a technique that converts a boosted strong classifier to a binary decision tree that improves computational speed as well.

1.3 Paper organization

Sections 2 and 3 study the strong classifier with two weak classifiers and three weak classifiers, respectively. Many insights are derived from the detailed discussion. Section 4 addresses a general strong classifier with more than three weak classifiers. Section 5 presents the experiments.

1.4 Notation

To simplify our analysis, we assume that all weak classifiers are statistically independent¹. Further, we assume that each weak classifier $h_j(x)$ has its own detection rate $(1 - a_j)$ and false alarm rate b_j . Table 1 summarizes the rates associated with the weak classifier $h_j(x)$, where \mathcal{P} denotes the positive set and \mathcal{N} the negative set.

	$p(h_j(x) = T)$	$p(h_j(x) = F)$
$x \in \mathcal{P}$	$1 - a_j$	a_j
$x \in \mathcal{N}$	b_j	$1 - b_j$

Table 1. Rates for the weak classifier $h_j(x)$.

We use the notation $\langle \mathcal{S}_J \rangle$ to denote a strong classifier that combines J weak classifiers or simply refer to it as the strong classifier $\langle \mathcal{S}_J \rangle$. We are often interested in conducting a case study. We use the notation $\langle \mathcal{S}_{J,n} \rangle$ to denote the n^{th} case of the strong classifier $\langle \mathcal{S}_J \rangle$, or simply refer to it as the strong classifier $\langle \mathcal{S}_{J,n} \rangle$.

Sometime we isolate, from a ‘mother’ strong classifier, a ‘child’ strong classifier that combines a subset of weak classifiers that belongs to the ‘mother’ strong classifier. Therefore, there is a need to specify the weak classifiers used by the ‘child’ strong classifier. We use the notation $\langle \mathcal{S}_{j,n} |_{\{\text{weak classifier IDs}\}} \rangle$. For example, the strong classifier $\langle \mathcal{S}_{2,3} |_{\{4,5\}} \rangle$ means the 3rd case of the ‘child’ strong classifier using two weak classifiers, the 4th and 5th ones, belonging to the ‘mother’ strong classifier.

2 Strong classifier $\langle \mathcal{S}_2 \rangle$

We start from a simple case of combining $J = 2$ weak classifiers with $\alpha_1 > \alpha_2 > 0$. The possible $g(x)$ values are given in Table 2. Note that $\theta_{11} > \theta_{10} > \theta_{01} > \theta_{00}$. There are five cases of interests.

$h_1(x)$	$h_2(x)$	$g(x)$
T	T	$\theta_{11} = \alpha_1 + \alpha_2$
T	F	$\theta_{10} = \alpha_1$
F	T	$\theta_{01} = \alpha_2$
F	F	$\theta_{00} = 0$

Table 2. Possible $g(x)$ values of combining $J = 2$ weak classifiers.

¹ It should be noted that we assumed the statistical independence among the weak classifiers for simplicity of illustration. The assumption of independence is only needed for calculating the detection and false alarm rates of the strong classifier. Therefore our case study analysis presented above still holds even when the weak classifiers are dependent. One evidence is that the ROC curve for strong classifier $\langle \mathcal{S}_2 \rangle$ always consists of five points, whether the two weak classifiers are independent or not.

2.1 Case study

Below, we examine each case separately. The detection and false alarm rates for all cases are listed in Table 3. The binary decision trees for all cases are shown in Figure 1.

$\langle \mathcal{S}_{2,1} \rangle$	$p(h(x) = T)$	$p(h(x) = F)$
$x \in \mathcal{P}$	0	1
$x \in \mathcal{N}$	0	1
$\langle \mathcal{S}_{2,2} \rangle$	$p(h(x) = T)$	$p(h(x) = F)$
$x \in \mathcal{P}$	$(1 - a_1)(1 - a_2)$	$a_1 + a_2 - a_1 a_2$
$x \in \mathcal{N}$	$b_1 b_2$	$1 - b_1 b_2$
$\langle \mathcal{S}_{2,3} \rangle$	$p(h(x) = T)$	$p(h(x) = F)$
$x \in \mathcal{P}$	$1 - a_1$	a_1
$x \in \mathcal{N}$	b_1	$1 - b_1$
$\langle \mathcal{S}_{2,4} \rangle$	$p(h(x) = T)$	$p(h(x) = F)$
$x \in \mathcal{P}$	$1 - a_1 a_2$	$a_1 a_2$
$x \in \mathcal{N}$	$b_1 + b_2 - b_1 b_2$	$(1 - b_1)(1 - b_2)$
$\langle \mathcal{S}_{2,5} \rangle$	$p(h(x) = T)$	$p(h(x) = F)$
$x \in \mathcal{P}$	1	0
$x \in \mathcal{N}$	1	0

Table 3. Rates for all five cases of the strong classifier $\langle \mathcal{S}_2 \rangle$.

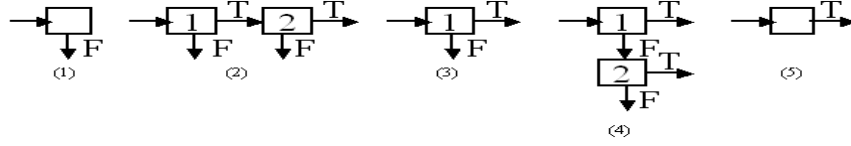


Fig. 1. The binary decision trees for all five cases of the strong classifier $\langle \mathcal{S}_2 \rangle$.

- strong classifier $\langle \mathcal{S}_{2,1} \rangle$: $\theta_{11} < \theta$. This is an all-fail case, i.e., the strong classifier fails all possible x .
- strong classifier $\langle \mathcal{S}_{2,2} \rangle$: $\theta_{10} < \theta \leq \theta_{11}$. The only way to pass the strong classifier is to pass both weak classifiers. Thus, the corresponding decision tree is just a cascade of two weak classifiers. Only when the first weak classifier is passed is the second weak classifier triggered.
- strong classifier $\langle \mathcal{S}_{2,3} \rangle$: $\theta_{01} < \theta \leq \theta_{10}$. Passing the strong classifier is equivalent to passing the first weak classifier and the second weak classifier is completely useless! Hence, we automatically save the computation of evaluating the second weak classifier.

- strong classifier $\langle \mathcal{S}_{2,4} \rangle$: $\theta_{00} < \theta \leq \theta_{01}$. The only way to fail the strong classifier is to fail both weak classifiers. Thus, the corresponding decision tree is also a cascade of two weak classifiers. However, only when the first weak classifier is failed is the second weak classifier triggered.
- strong classifier $\langle \mathcal{S}_{2,5} \rangle$: $\theta < \theta_{00}$. This is an all-pass case, i.e., the strong classifier passes all possible x .

The above 5 cases can be summarized using a receiver operating characteristic (ROC) curve plotted in Figure 2 (with $a_1 = 0.2$, $a_2 = 0.3$, $b_1 = 0.4$, $b_2 = 0.5$). The ROC curve consists of a set of discrete points. As the detection rate increases, the false alarm rate increases too.

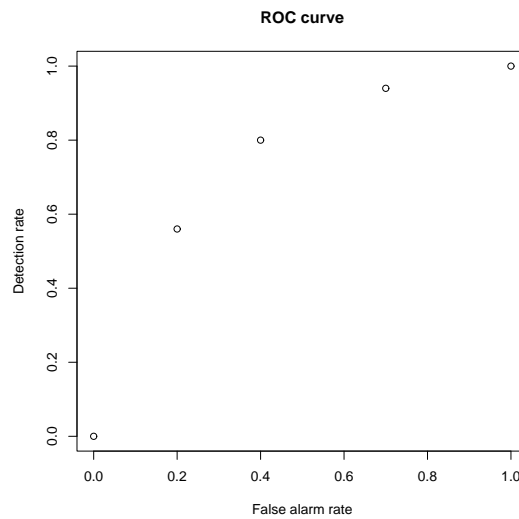


Fig. 2. A typical ROC curve for the strong classifier $\langle \mathcal{S}_2 \rangle$.

2.2 Discussion

The following issues are worthy of further clarification.

[XOR problem]. The XOR problem means that Table 4 or similar thing holds. The XOR problem might result in a very complicate binary decision tree so that there is no computational saving available. However, this cannot happen to the strong classifier $\langle \mathcal{S}_2 \rangle$ because, if the second row holds, then the $h(x)$ function in the first row must take a true value owing to the additive nature of the $g(x)$ function.

[Tie $\alpha_1 = \alpha_2$]. If $\alpha_1 = \alpha_2$, the strong classifier $\langle \mathcal{S}_{2,3} \rangle$ never exists. We only consider cases 1, 2, 4, and 5.

$h_1(x)$	$h_2(x)$	$h(x)$
T	T	F
T	F	T
F	T	T
F	F	F

Table 4. The XOR problem.

[Strong classifier threshold $\theta = \frac{1}{2}(\alpha_1 + \alpha_2)$]. When the strong classifier threshold is $\theta = \frac{1}{2}(\alpha_1 + \alpha_2)$, the strong classifier $\langle \mathcal{S}_{2,3} \rangle$ is applied, which means that the first stage of the Viola and Jones cascade that uses a combination of two weak classifiers can be replaced by one weak classifier. In Viola and Jones [1], the first strong classifier of a cascade of strong classifiers uses two weak classifiers and its strong classifier threshold is adjusted to achieved a near 100% detection rate. We will examine this point next.

[Design issue of the strong classifier $\langle \mathcal{S}_{2,2} \rangle$]. In this strong classifier, the order of the two weak classifiers can be interchanged without affecting the final decision. In practice, we select the weak classifier with lower false alarm rate as the first one because this way the negative example exits the binary decision tree more quickly. For the positive example, both weak classifiers need to be evaluated.

[Design issue of the strong classifier $\langle \mathcal{S}_{2,4} \rangle$]. In this strong classifier, the order of the two weak classifiers can be interchanged without affecting the final decision. In practice, we select the weak classifier with higher detection rate as the first one because this way the positive example exits the binary decision tree more quickly. For the negative example, both weak classifiers need to be evaluated.

[100% detection rate]. Adjusting the strong classifier threshold θ is equivalent to running on the ROC curve. To achieve a 100% detection rate by adjusting θ as suggested in [1], theoretically there are only one feasible way, that is, choosing the classifier $\langle \mathcal{S}_{2,5} \rangle$. However, the false alarm rate is also 100% in this case, indicating that the classifier $\langle \mathcal{S}_{2,5} \rangle$ is useless in practice. Therefore, adjusting the strong classifier threshold in principle is a dangerous practice to achieve 100% detection. The strong classifier $\langle \mathcal{S}_{2,4} \rangle$ is the practical choice with the highest detection rate and moderate false alarm rate.

Ultimately, the detection rate of the strong classifier depends on the detection rates of the weak classifiers. If the weak classifier classifiers all have 100% detection rate, the strong classifiers from $\langle \mathcal{S}_{2,2} \rangle$ to $\langle \mathcal{S}_{2,4} \rangle$ all have 100% detection rate. In practice, we will choose the strong classifier $\langle \mathcal{S}_{2,2} \rangle$ owing its lower false alarm rate and its early exit for negative examples. Therefore, adjusting the weak classifier thresholds is a more feasible approach to achieving 100% detection rate.

3 Strong classifier $\langle \mathcal{S}_3 \rangle$

We now consider combining $J = 3$ weak classifiers with $\alpha_1 > \alpha_2 > \alpha_3 > 0$. The possible $g(x)$ values are given in Table 5. Note that $\{\theta_{111}, \dots, \theta_{000}\}$ are in a descending order except the order ambiguity between θ_{100} and θ_{011} . If $\alpha_1 > \alpha_2 + \alpha_3$, then $\theta_{100} > \theta_{011}$. There are ten cases of interests.

$h_1(x)$	$h_2(x)$	$h_3(x)$	$g(x)$
T	T	T	$\theta_{111} = \alpha_1 + \alpha_2 + \alpha_3$
T	T	F	$\theta_{110} = \alpha_1 + \alpha_2$
T	F	T	$\theta_{101} = \alpha_1 + \alpha_3$
T	F	F	$\theta_{100} = \alpha_1$
F	T	T	$\theta_{011} = \alpha_2 + \alpha_3$
F	T	F	$\theta_{010} = \alpha_2$
F	F	T	$\theta_{001} = \alpha_3$
F	F	F	$\theta_{000} = 0$

Table 5. Possible $g(x)$ values of combining $J = 3$ weak classifiers.

3.1 Case study

Below, we examine each case separately. Similarly, we can compute the detection and false alarm rates for all cases. The binary decision trees for all cases are shown in Figure 3.

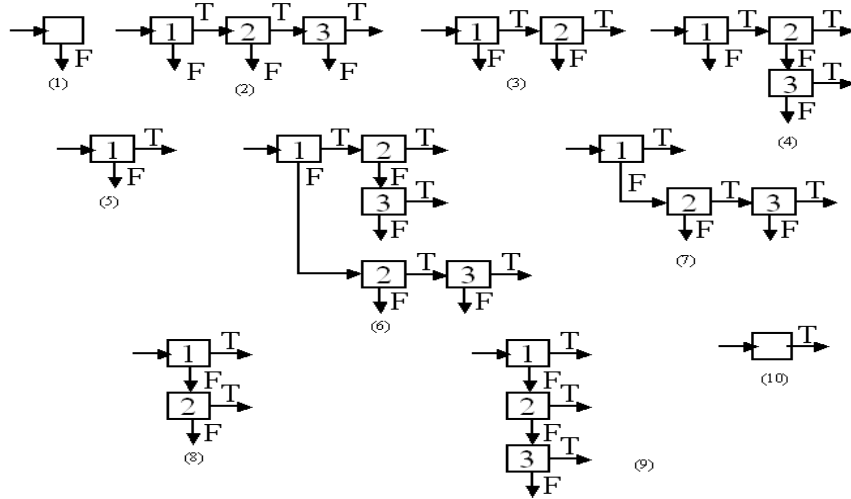


Fig. 3. The binary decision trees for all 10 cases of the strong classifier $\langle \mathcal{S}_3 \rangle$.

- strong classifier $\langle \mathcal{S}_{3,1} \rangle$: $\theta_{111} < \theta$. This is an all-fail case, i.e., the strong classifier fails all possible x .
- strong classifier $\langle \mathcal{S}_{3,2} \rangle$: $\theta_{110} < \theta \leq \theta_{111}$. The only way to pass the strong classifier is to pass all three weak classifiers. The strong classifier $\langle \mathcal{S}_{3,2} \rangle$ is a cascade of three weak classifiers or a cascade of the first weak classifier and the strong classifier $\langle \mathcal{S}_{2,2} |_{\{2,3\}} \rangle$.

- strong classifier $\langle \mathcal{S}_{3,3} \rangle$: $\theta_{101} < \theta \leq \theta_{110}$. The way to pass the strong classifier is to pass the first two weak classifiers. This reduces to the strong classifier $\langle \mathcal{S}_{2,2} |_{\{1,2\}} \rangle$. An alternative way is to view the strong classifier $\langle \mathcal{S}_{3,3} \rangle$ as a cascade of the first weak classifier and the strong classifier $\langle \mathcal{S}_{2,3} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,4} \rangle$: $\max(\theta_{100}, \theta_{011}) < \theta \leq \theta_{101}$. The way to pass the strong classifier is to first pass the first weak classifier and then the strong classifier $\langle \mathcal{S}_{2,4} |_{\{2,3\}} \rangle$. Therefore, the strong classifier $\langle \mathcal{S}_{3,2} \rangle$ is a cascade of the first weak classifier and the strong classifier $\langle \mathcal{S}_{2,4} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,5} \rangle$: $\min(\theta_{100}, \theta_{011}) < \theta \leq \max(\theta_{100}, \theta_{011})$ and $\theta_{100} = \max(\theta_{100}, \theta_{011})$. The way to pass the strong classifier is to simply pass the first weak classifier. Also, the strong classifier $\langle \mathcal{S}_{3,5} \rangle$ can be viewed as a cascade of the first weak classifier and the strong classifier $\langle \mathcal{S}_{2,5} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,6} \rangle$: $\min(\theta_{100}, \theta_{011}) < \theta \leq \max(\theta_{100}, \theta_{011})$ and $\theta_{011} = \max(\theta_{100}, \theta_{011})$. There are two ways to pass the strong classifier. The first way is to pass the strong classifier $\langle \mathcal{S}_{3,4} \rangle$. The second is to first fail in the first weak classifier (this is allowed!) but then pass the strong classifier $\langle \mathcal{S}_{2,2} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,7} \rangle$: $\theta_{010} < \theta \leq \min(\theta_{100}, \theta_{011})$. There are two ways to pass the strong classifier. The first way is to pass the first weak classifier. The second is to first fail in the first weak classifier but then pass the strong classifier $\langle \mathcal{S}_{2,2} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,8} \rangle$: $\theta_{001} < \theta \leq \theta_{010}$. There are two ways to pass the strong classifier. The first way is to pass the first weak classifier. The second is to first fail in the first weak classifier but then pass the second weak classifier or the strong classifier $\langle \mathcal{S}_{2,3} |_{\{2,3\}} \rangle$.
- *strong classifier* $\langle \mathcal{S}_{3,9} \rangle$: $\theta_{000} < \theta \leq \theta_{001}$. The only way to fail the strong classifier is to fail all three weak classifiers. In an alternative perspective, there are two ways to pass the strong classifier. The first way is to pass the first weak classifier. The second is to first fail in the first weak classifier but then pass the strong classifier $\langle \mathcal{S}_{2,4} |_{\{2,3\}} \rangle$.
- strong classifier $\langle \mathcal{S}_{3,10} \rangle$: $\theta < \theta_{000}$. This is an all-pass case, i.e., the strong classifier passes all possible x .

3.2 Discussion

The same issues (such as XOR problem, interchanging the order of weak classifiers for specific strong classifiers, 100% detection rate, etc.) for the strong classifier $\langle \mathcal{S}_2 \rangle$ addressed in section 2.2 exists for the strong classifier $\langle \mathcal{S}_3 \rangle$. However, most of them can be similarly treated. Here we address only some newly introduced issues.

[Order ambiguity between θ_{100} and θ_{011}]. The order ambiguity between θ_{100} and θ_{011} arises from the undetermined relationship between α_1 and $\alpha_2 + \alpha_3$. The weight pre-sorting can only automatically fix orders of all possible $g(x)$ values to some extent (in fact sub-exponentially!). In practice, we would prefer the case $\theta_{100} > \theta_{011}$ because it provides a far simpler binary decision tree. However, the experimental results seldom present this. We will return to this point in section 4.

[Recursive tree construction]. As we have mentioned in the case study and illustrated in Figure 3, all the binary decision trees corresponding to the strong classifier

$\langle \mathcal{S}_3 \rangle$ can be constructed as follows: The top node contains the first weak classifier, followed by the strong classifier $\langle \mathcal{S}_2|_{\{2,3\}} \rangle$ (with different case numbers though). In other words, we can construct the tree recursively. This idea applies to the strong classifier with an arbitrary value of J . We will examine this in detail in Section 4.

4 Strong classifier $\langle \mathcal{S}_{J>3} \rangle$

A complete analysis of the strong classifier $\langle \mathcal{S}_{J>3} \rangle$ becomes tedious. Especially when J takes a very big value, the tree could be complex. Below, we present a general comprehension of the asymptotic behavior of the tree corresponding to the strong classifier $\langle \mathcal{S}_{J>3} \rangle$. We also address the same issue from an information theory perspective.

[Number of cases]. As we have seen before, the tree structure is completely determined by the weight coefficients α and the strong classifier threshold θ . A different choices of α and θ might yield a completely different tree. Unfortunately, the number of all possible trees (or cases) grows exponentially with the number of weak classifiers J . It roughly equals to $O(2^{J+1})$. This number can be estimated as follows. If there is no order ambiguity, there are $O(2^J)$ cases (exactly $2^J + 1$ cases). The case with order ambiguity also grows as $O(2^J)$ because the unambiguous cases that can be eliminated by the condition $\alpha_1 > \dots > \alpha_J > 0$ is $o(2^J)$. It should be noted that the case study is introduced for theoretic analysis only. In practice, when only one configuration of α and θ is available, there is no need to store all cases.

[Tree construction]. The tree for the strong classifier $\langle \mathcal{S}_{J,n} \rangle$ can be constructed recursively, as illustrated in Figure 4. First test the first weak classifier. If true, test the next binary decision tree corresponding to the strong classifier $\langle \mathcal{S}_{J-1,n_1} \rangle$ (with case number n_1); if false, test the next binary decision tree corresponding to $\langle \mathcal{S}_{J-1,n_0} \rangle$ (with case number n_0). Therefore, at each node, we only need to remember two case numbers. Similarly, the tree can be constructed backward, starting from the n^{th} weak classifier.

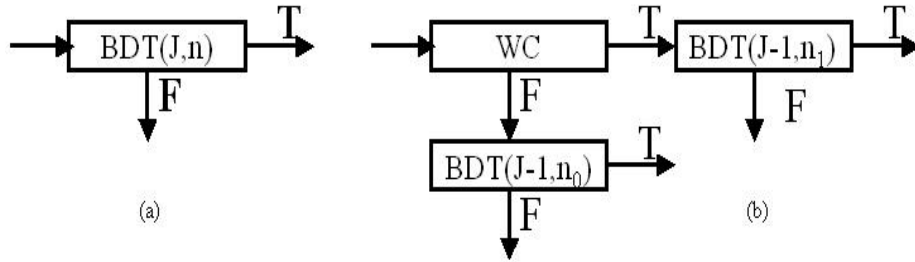


Fig. 4. Recursive construction of the decision tree. (a) Decision tree corresponding to the strong classifier $\langle \mathcal{S}_J \rangle$. (b) The equivalent decision tree constructed using the first weak classifier followed two decision tree corresponding to the strong classifiers $\langle \mathcal{S}_{J-1,n_1} \rangle$ and $\langle \mathcal{S}_{J-1,n_0} \rangle$.

[Tree complexity]. It seems that the constructed tree can be rather complicated because it may exhaust all possible nodes of a binary decision tree. However, in practice,

we found that many nodes disappears because of the additive nature of the strong classifier and there is no XOR problem.

[Information]. In summary, the decision tree is just another way of storing the same amount of information. Therefore, we did not gain or loss any information. To be more specific, the information of the strong classifier $\langle S_J \rangle$ lies in the J weighting coefficients α_j , the strong classifier threshold θ , and the comparison to determine the final classification result, whereas the information of the decision tree lies in the tree structure. Because of the discrete nature of the weak classifier responses (true or false), the binary decision tree encodes more than one configuration of the weights α and the threshold θ . Therefore, the binary decision trees partition the complete information space covering all possible configurations of weights α and strong classifier threshold θ into a finite number of subsets. Also, binary decision tree helps visualize and interpret the results.

(a) Exp. 1, strong classifier 1	Default	Fast exit	Binary decision tree
Detection(train)	97.3%	97.3%	97.3%
False alarm (train)	5.89%	5.89%	5.89%
Duration	64.80 s	40.16 s	36.98 s
Detection (test1)	94.2%	94.2%	94.2%
False alarm (test1)	9.07%	9.07%	9.07%
Duration	26.91 s	16.86 s	15.17 s
False alarm (test2)	16.4%	16.4%	16.4%
(b) Exp. 1, strong classifier 2	Default	Fast exit	binary decision tree
Detection(train)	100%	100%	100%
False alarm (train)	82.4%	82.4%	82.4%
Duration	64.67 s	34.56 s	29.12 s
Detection (test1)	99.96%	99.96%	99.96%
False alarm (test1)	75.8%	75.8%	75.8%
Duration	26.88 s	17.59 s	15.93 s
False alarm (test2)	46.4%	46.4%	46.4%
(c) Exp. 2	Default	Fast exit	Binary decision tree
Detection(train)	99.98%	99.98%	99.98%
False alarm (train)	2.78%	2.78%	2.78%
Duration	991 s	561 s	86.8 s
Detection (test1)	98.0%	98.0%	98.0%
False alarm (test1)	8.41%	8.41%	8.41%
Duration	349 s	253 s	32.5 s
False alarm (test2)	12.3%	12.3%	12.3%

Table 6. Summary of various face detection algorithms. (a) The strong classifier composed of 5 weak classifiers before adjusting the strong classifier threshold. (b) The strong classifier composed of 5 weak classifiers after adjusting the strong classifier threshold. (c) The cascade of 3 strong classifiers.

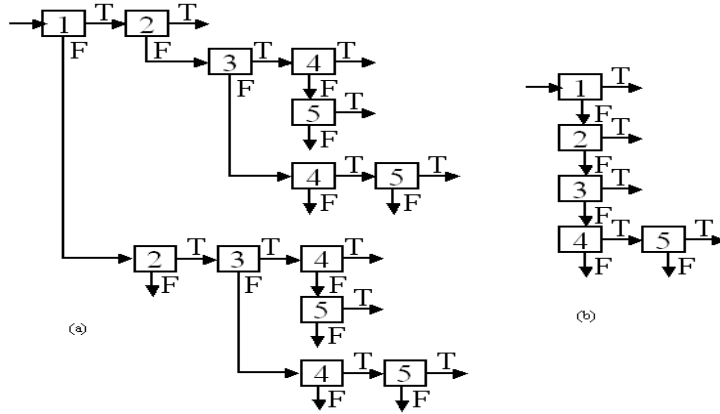


Fig. 5. Two binary decision trees corresponding the first stage strong classifier in the face detector with two different strong classifier thresholds.

5 Classification experiment

We compared the binary decision tree implementation with other two methods. The default method is to first calculate the value $g(x)$ and then compare it with θ . The fast exit algorithm [12] utilized the additive nature of the function $g(x) = \sum_{j=1}^J \alpha_j h_j(x)$. Denote the sum up to k by $g_k(x) = \sum_{j=1}^k \alpha_j h_j(x) = g_{k-1}(x) + \alpha_k h_k(x)$. The fast exit algorithm uses the following two facts:

1. If $g_k(x)$ is already larger than the threshold θ , we can safely declare x as positive.
2. If $g_k(x) + \sum_{j=k+1}^J \alpha_j$ is already smaller than θ , we can safely declare x as negative.

The essence is to check at each weak classifier if we can exit the whole strong classifier quickly.

5.1 The first experiment

In this experiment, we used 3724 frontal face patches of size 24×24 (in fact 1862 images with their mirrors) and 6231 negative examples sampled from natural images to boost a strong classifier consisting of 5 weak classifiers. Some training images are shown in Figure 6. This boosted strong classifier has the following form:

$$2.550h_1(x) + 1.606h_2(x) + 1.288h_3(x) + 1.274h_4(x) + 1.096h_5(x).$$

[Exp. 1, strong classifier 1]. When we take the strong classifier threshold as default, i.e. $\theta = 3.907$, we achieve a detection rate of 97.3% at the cost of a false alarm rate of 5.89% on the training data. The corresponding binary decision tree is shown in Figure 5(a).

[Exp. 1, strong classifier 2]. When adjusting the strong classifier threshold to pass all training positive samples, the threshold is set to $\theta = 1.287$ and the false alarm rate increases to 82.4%! The corresponding binary decision tree is shown in Figure 5(b), which is nothing but the following: to fail the strong classifier is to fail the first 3 weak classifiers and then fail either of the 4th and 5th weak classifiers.

We used the above strong classifier to classify two test datasets. (i) The first testing data set contains 4858 positives (2429 images with their mirrors) and 8569 negatives. This data set is downloaded from the MIT CBCL website [17] and used in [18] for face detection. The original image size is 19×19 and we normalized it to 24×24 using a bicubic interpolation. Sample images in this testing data set are shown in Figure 7. Note that the face images are shifted, rotated, and scaled version of the same object that is not explicitly learned in the training stage. (ii) The second testing data set exhaustively sampled 7740000 negatives from 10 images of size 1042×768 known to be no face inside. Two such images are shown in Figure 8. This is used for testing negative rejection at a large scale.



Fig. 6. Example of positives (the top row) and negatives (the bottom row) of size 24×24 in the training data set.



Fig. 7. Example of positives (the top row) and negatives (the bottom row) of size 24×24 in the first testing data set.

It is obvious from Table 6(a) and (b) that the binary decision tree is the fastest algorithm for the first dataset, almost two times faster than the default method. The fast exit algorithm is more efficient than the default method, only a bit slower than the binary decision tree. However, here we only used one strong classifier. When a cascade of strong classifier is used as shown later, the decision tree implementation is much faster than the fast exit algorithm. As expected, in the terms of the detection and false alarm rates, the binary decision tree yields the exactly same results as the default and fast exit algorithms.



Fig. 8. Two testing images of size 1024×768 in the second testing data set.

5.2 The second experiment

In this experiment, we mimicked a detection scenario and tested the computation of the cascade of strong classifiers. We followed the negative selection to training the strong classifier cascade as in [1]. We used the same 3724 positives and selected negatives from web images whenever needed. For simplicity, we trained a cascade of three strong classifiers with 5, 10, and 200 weak classifiers at each stage. We constructed the binary decision trees for the first two stages and used the fast exit algorithm for the third stage. Table 6(c) compares the performance of different algorithms. This experiment truly manifests the computational advantage of the binary decision tree. It is significantly faster than the other two implementations. It is more than ten times faster than the default method and more than six times faster than the fast exit algorithm!

6 Conclusions

We presented a computational improvement to the rapid Viola and Jones algorithm [1]. The improvement arises from the fact that a binary decision tree is derived from a boosted strong classifier and the tree has a lot of early exits. In addition, each node of the tree is just a weak classifier and the knowledge of the coefficients α_j can be discarded once the tree is constructed. Our experiments demonstrated that the binary decision tree implementation is indeed rapider.

Acknowledgement

The author thanks Binglong Xie for providing the face database used for training.

References

1. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. IEEE Computer Society Conference on Computer Vision and Pattern Recognition **1** (2001) 511–518

2. Papageorgiou, C., Oren, M., Poggio, T.: A general framework for object detection. *IEEE International Conference on Computer Vision* (1998) 555–562
3. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. (*Computation Learning Theory: Eurocols*) 23–27
4. Jones, M., Viola, P.: Face recognition using boosted local features. *IEEE International Conference on Computer Vision* (2003)
5. Jones, M., Viola, P.: Fast multi-view face detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2003)
6. Lienhart, R., Maydt, J.: An extended set of harr-like features for rapid object detection. *IEEE International Conference on Image Processing* **1** (2002) 900–903
7. Zhang, L., Li, S., Qu, Z., Huang, X.: Boosting local feature based classifier for face recognition. *First IEEE International Workshop on Face Processing in Video* (2004)
8. Lienhart, R., Kuranov, A., Pisarevsky, A.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *The 25th Pattern Recognition Symposium* (2003) 297–304
9. Viola, P., Jones, M.: Face and robust classification using asymmetric adaboost and a detector cascade. *Neural Information Processing Systems* **14** (2001)
10. Li, S., Zheng, Z.: Floatboost learning and statistical face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* **26** (2004) 1–12
11. Lienhart, R., Liang, L., Kuranov, A.: A detector tree of boosted classifiers for real-time object detection and tracking. *IEEE International Conference on Multimedia and Expo* (July 2003)
12. Sun, J., Rehg, J.M., Bobick, A.: Automatic cascade training with perturbation bias. In: *CVPR*. (2004)
13. Casella, G., Berger, R.L.: *Statistical Inference*. Duxbury (2002)
14. Poor, H.: *An Introduction to Signal Detection and Estimation*. Springer-Verlag (1994)
15. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. *Annals of statistics* **28** (2000) 337–374
16. Grossmann, E.: Adatree: Boosting a weak classifier into a decision tree. In: *IEEE Workshop Learning in Computer Vision and Pattern Recognition*. (2004)
17. CBCL Face Database #1, MIT Center for Biological and Computational Learning, <http://www.ai.mit.edu/projects/cbcl>.
18. Sung, K., Poggio, T.: Example-based learning for view-based human face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* **20** (1998) 39–51