Cascade Learning by Optimally Partitioning

Yanwei Pang, *Senior Member, IEEE,* Jiale Cao, and Xuelong Li, *Fellow, IEEE*

***Abstract*—Cascaded AdaBoost classifier is a well-known efficient object detection algorithm. The cascade structure has many parameters to be determined. Most of existing cascade learning algorithms are designed by assigning detection rate and false positive rate to each stage either dynamically or statically. Their objective functions are not directly related to minimum computation cost. These algorithms are not guaranteed to have optimal solution in the sense of minimizing computation cost. On the assumption that a strong classifier is given, in this paper, we propose an optimal cascade learning algorithm (iCascade) which iteratively partitions the strong classifiers into two parts until predefined number of stages are generated. iCascade searches the optimal partition point r_i of each stage by directly minimizing the computation cost of the cascade. Theorems are provided to guarantee the existence of the unique optimal solution. Theorems are also given for the proposed efficient algorithm of searching optimal parameters r_i. Once a new stage is added, the parameter r_i for each stage decreases gradually as iteration proceeds, which we call decreasing phenomenon. Moreover, with the goal of minimizing computation cost, we develop an effective algorithm for setting the optimal threshold of each stage. In addition, we prove in theory why more new weak classifiers in the current stage are required compared to that of the previous stage. Experimental results on face detection and pedestrian detection demonstrate the effectiveness and efficiency of the proposed algorithm.**

***Index Terms*—AdaBoost, cascade learning, classifier design, object detection.**

I. Introduction

THE EFFICIENCY of object detection is determined by the types of features [1], the manner of the features to be extracted, and the structure of the classifiers [2]–[8]. This paper is concentrated on the classifier structure. AdaBoost classifiers with cascade structure have greatly contributed to real-time face detection [9]–[13] and pedestrian detection [14]–[18]. With cascade structure, a large fraction of subwindows can be rejected at early stages with a small number of weak classifiers. Only the subwindows of true positives and those similar to true positives can arrive at later stages. However, how to design an optimal cascade structure is an open problem which is the focus of this paper.

Cascade learning is the process of determining the parameters of a cascade in order to improve the efficiency of AdaBoost classifier. The cascade parameters mainly include the number of stages, the number of weak classifiers in each stage, and the thresholds for each stage. However, most of existing cascade learning methods are not directly formulated as a constrained optimization problem. Though more efficient than the noncascade one, they are not guaranteed to be the best in the sense of maximizing detection efficiency under the acceptable constraints. Usually, there are many hand-crafted parameters which are chosen according to one's intuition and experience. The performance of the cascade AdaBoost relies on one's insight into the cascade structure. As Saberian and Vasconcelos [19] mentioned, the design of a good cascade structure can even take up several weeks. In addition, some useful intuitions are not justified in theory.

To overcome the above problems, we formulate cascade learning as a process of learning the parameters of a cascade by directly minimizing the computation cost with some certain constraints. Instead of jointly seeking the optimal number of weak classifiers of each stage and the optimal threshold for each stage, our method consists of two successive steps.

1) By directly minimizing the computation cost, we find the optimal partition point of each stage by using the conservative thresholds which guarantee 100% detection rate of training positives (see Sections III–V-A and V-B). Based on the optimal partition point of each stage, the number of weak classifiers in each stage can be obtained.
2) Based on the learned optimal partition point of each stage and the single predefined target detection rate (e.g., 0.98), we find the optimal stage thresholds by computing the derivative of the computational cost with respect to detection rate (see Section V-C).

In summary, the contributions and characteristics of this paper are as follows.

1) We transform the strong classifier of regular AdaBoost into an optimal cascade structure. That is, the result of regular AdaBoost is the input of our cascade learning algorithm. In the sense of detection rate and rejection rate, we use cascade AdaBoost to approach its noncascade one (i.e., regular AdaBoost) with minimized computation cost.
2) The objective function of our method is just the computation cost of a cascade. In contrast, most of the existing algorithms are designed by empirically assigning detection rate and false positive rate to each stage either

Manuscript received August 21, 2015; revised April 10, 2016 and June 6, 2016; accepted July 25, 2016. Date of publication September 12, 2016; date of current version November 15, 2017. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2014CB340400, and in part by the National Natural Science Foundation of China under Grant 61632081. This paper was recommended by Associate Editor Q. Ji.

Y. Pang and J. Cao are with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: pyw@tju.edu.cn; connor@tju.edu.cn).

X. Li is with the Center for Optical Imagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelong_li@opt.ac.cn).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This includes a PDF file that contains the proof of several theorems of the paper. The total size of the file is 70 KB.

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

Digital Object Identifier 10.1109/TCYB.2016.2601438

Algorithm 1 Two-Stage Cascade

1: if $H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) \leq t$,
2: then $l(\mathbf{x}) = -1$,
3: else (i.e., $H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) > t$)
4: if $H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) \leq t$
5: $l(\mathrm{x}) = -1$,
6: else (i.e., $H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) > t$)
7: $l(\mathbf{x}) = 1$.

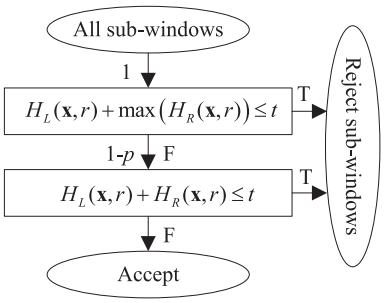


Fig. 1. Proposed method: two-stage cascade AdaBoost with a given r.

dynamically or statically. Existence and uniqueness of the optimal solution are analytically proved.

3) To design a two-stage cascade structure, we propose to partition the strong classifier $H(\mathbf{x})$, a combination of weak classifiers $h_1, \ldots, h_T$, into left part $H_L(\mathbf{x}, r_1)$ and right part $H_R(\mathbf{x}, r_1)$ at partition point r_1 (see Algorithm 1 and Fig. 1). The optimal partition point r_1 is found by minimizing the objective function $f_1(r)$ which stands for the computation cost of the cascade classifier. We theoretically (i.e., Theorem 1) prove that $f_1(r)$ has a unique solution. We further give a rough estimation (i.e., Theorem 2) of the optimal solution.
4) To design a three-stage cascade structure, we propose to further the partition right classifier $H_R(\mathbf{x}, r_1)$ into two parts at another partition point r_2. The partition iteratively continues (see Fig. 4). This algorithm is not globally optimal if r_1 is fixed while r_2 is considered as a variable. To obtain global optimization, we further jointly model the computation cost $f(r_1, r_2)$ with variables r_1 and r_2. We prove that $f(r_1, r_2)$ has a unique minimum solution (see Theorem 7). An iterative optimization algorithm (i.e., Algorithm 2) is proposed to find the optimal solution. Theoretical analysis (i.e., Theorems 9–12) is given to support that r_1 decreases in each iteration where r_2 is fixed and r_2 decreases in each iteration where r_1 is fixed. We call it decreasing phenomenon. Such globally optimal three-stage cascade learning algorithm can be easily generalized to multistage one (i.e., Algorithm 3).
5) Moreover, we contribute to learn the optimal threshold t_i of each stage classifier for minimizing computation cost f_S of the cascaded classifier. We prove that the computation cost decreases with the increase of stage threshold t_i (i.e., Theorem 13). Based on this theorem, we develop an effective threshold learning algorithm (i.e., Algorithm 4) whose core is optimally increasing t_i. We call the proposed algorithm (i.e., Algorithms 3, 4, and the procedure in Fig. 9) iCascade.

II. Related Work

Most of existing cascade learning algorithms can be called DF-guided (where "DF" stands for detection rate and false positive rate) method pioneered by Viola and Jones [20]. In the learning step, DF-guided method selects weak classifiers step by step until predefined minimum acceptable detection rate and maximum acceptable false positive rate are satisfied. We call this method VJCascade [20].

Variants of VJCascade have been proposed to select and organize weak classifiers. BoostChain [21] improves VJCascade by reusing the ensemble score from previous stages to enhance the performance of the current stage. Brubaker *et al.* [9] called such a technique BoostChain recycling. Similar to BoostChain, Soft-Cascade also allows for monotonic accumulation of information as the classifier is evaluated [22]. In multiexit AdaBoost [23], node classifier also shares overlapping sets of weak classifiers. FloatBoost [24] as well as BoostChain uses DF-guided strategy to design the cascade. But different from VJCascade, FloatBoost uses backtrack mechanism to eliminate the less useful or even detrimental weak classifiers. Wu *et al.* [25] employed forward feature selection algorithm to greedily select features. Wang *et al.* [26] developed an asymmetric learning algorithm for both feature selection and ensemble classifier learning. FisherBoost [27] uses column generation technique to implement totally-corrective boosting algorithm. To decrease the training burden caused by the large number of negative samples and over-complete features (e.g., Haar-like features), some algorithms use only a random subset of the feature pool [9], [22].

Endeavor has also been devoted to adjust the thresholds of stages in the cascade which is also called the thresholds of node classifiers (also known as stage classifiers). On the assumption that a full cascade has been trained by VJCascade algorithm, Luo [28] proposed to jointly optimize the setting of the thresholding parameters of all the node classifiers within the cascade. WaldBoost algorithm utilizes an adaptation of Wald's sequential probability ratio test to set stage thresholds [29]. Brubaker *et al.* [9], [30] proposed a linear program algorithm to select weak classifiers and threshold of a node classifier.

Though most of existing methods are DF-guided, computation-cost guided (i.e., CC-guided) methods were also developed. Chen and Yuille [31] gave a criterion for designing a time-efficient cascade that explicitly takes into account the time complexity of tests including the time for preprocessing. They designed a greedy algorithm to minimize the criterion. But each stage in this method is constrained to detect all positive examples, which leads it to miss opportunity to improve detection efficiency [32]. The loss function of Cronus cascade learning algorithm is a tradeoff between accuracy (training error) and computation cost [32]. Cost-sensitive tree of classifiers (CSTCs) combines regularized training error

and computation cost into a loss function [33]. Compared to VJCascade-like method, CSTC is suitable for balancing time cost and accuracy [33].

In contrast to the above methods, the objective function (i.e., loss function) of our method is just the computation cost and only a single target detection rate is assigned. In addition, global solution instead of local one can be obtained in our method by learning the optimal partition point of each stage and the optimal threshold for each stage.

III. Proposed Method: Two-Stage Cascade

In this section, we describe the proposed two-stage cascade which is the foundation of our multistage cascade. Please note in Sections III–V-A and V-B, the thresholds for each stage are set to guarantee 100% detection rate of training positives.

A. Testing Stage

In our method, cascade AdaBoost is considered as an estimation of regular AdaBoost. A good cascade structure can achieve the same detection accuracy as regular AdaBoost with small computation cost. Therefore, we begin with describing the form of the strong classifier of regular AdaBoost.

Let $H(\mathbf{x})$ be the strong classifier obtained by an AdaBoost algorithm. The strong classifier $H(\mathbf{x})$ is composed of T weak classifiers $h_i(\mathbf{x})$ with their weights α_i

$$H(\mathbf{x}) = \sum\nolimits_{i=1}^{T} \alpha_i h_i(\mathbf{x}). \tag{1}$$

Generally, the weights of the weak classifiers satisfy

$$\alpha_1 > 0,\ \alpha_2 > 0, \ldots,\ \alpha_T > 0 \tag{2}$$

and

$$\sum\nolimits_{i=1}^{T} \alpha_i = 1. \tag{3}$$

Equation (3) holds when the weights are normalized.

Let $\hat{l}(\mathbf{x}) \in \{1, -1\}$ be the predicted class label of the detection window $\mathbf{x}$. The decision rule of the strong classifier $H(\mathbf{x})$ is

$$\hat{l}(\mathbf{x}) = \begin{cases} 1, & \text{if } H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}) > t \\ -1, & \text{if } H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}) \le t \end{cases} \tag{4}$$

where the threshold $t = 0$ if $h_i(\mathbf{x}) \in \{1, -1\}$ and $t = 0.5$ if $h_i(\mathbf{x}) \in \{0, 1\}$. In practice, other values of t can be used for balancing the detection rate and false positive rate.

In two-stage cascade, there is only one stage in which a small number r (e.g., 10) of weak classifiers are combined for classification. The core of the proposed two-stage cascade is to determine an optimal r which divides the strong classifier $H(\mathbf{x})$ into left part $H_L(\mathbf{x}, r)$ and right part $H_R(\mathbf{x}, r)$

$$H(\mathbf{x}, r) = H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) \tag{5}$$

$$H_L(\mathbf{x}, r) = \sum\nolimits_{i=1}^{r} \alpha_i h_i(\mathbf{x}) \tag{6}$$

$$H_R(\mathbf{x}, r) = \sum\nolimits_{i=r+1}^{T} \alpha_i h_i(\mathbf{x}). \tag{7}$$

To reject true negative subwindows with less computation cost, we propose to use the maximum of $H_R(\mathbf{x}, r)$ to approximate the value of $H_R(\mathbf{x}, r)$

$$\max H_R(\mathbf{x}, r) = \max_{\mathbf{x}} \left(\sum_{i=r+1}^{T} \alpha_i h_i(\mathbf{x}) \right) = \sum_{i=r+1}^{T} \alpha_i. \tag{8}$$

We denote the maximum by $\max H_R(\mathbf{x}, r)$. With $\max H_R(\mathbf{x}, r)$, it is guaranteed that all the negative subwindows can be correctly rejected if the following inequality holds:

$$H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) \le t. \tag{9}$$

That is, some subwindows can be rejected by using merely $H_L(\mathbf{x}, r)$ and $\max H_R(\mathbf{x})$ instead of both $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$. Thus, computation cost is significantly reduced.

The rest subwindows not satisfying (9) have to be classified using both $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ (i.e., the strong classifier). If the sum of $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ is not larger than t, that is

$$H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) = H(\mathbf{x}) \le t \tag{10}$$

then the subwindow $\mathbf{x}$ can be finally classified as negative subwindow. Otherwise (i.e., the sum is larger than t), it is classified as positive subwindow. The algorithm of two-stage cascade is given in Algorithm 1. Equivalently, the flow-chart is shown in Fig. 1. In Fig. 1 and Algorithm 1, $\max H_R(\mathbf{x}, r)$ can be moved to the right side and then t-$\max H_R(\mathbf{x}, r)$ represents the new threshold. In this section, the threshold are set to guarantee 100% detection rate. The issue of how to set the threshold for a given target detection rate is addressed in Section V-C.

B. Training Stage: How to Select Optimal r

In Fig. 1, it is assumed that r and $\max H_R(\mathbf{x}, r)$ are given. In this section, we describe how to choose an optimal r. $\max H_R(\mathbf{x}, r)$ can be easily computed from training samples once r is given. In the training stage of cascade learning, it is assumed that the strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ is obtained by a regular AdaBoost algorithm with technology of bootstrap.

Given r, a p fraction of true negative subwindows can be rejected by using left classifier $H_L(\mathbf{x})$ [i.e., (9)]. The fraction p is called rejection rate and defined by

$$p(r, t) = \frac{\sum_{\mathbf{x}} I\left(\sum_{i=1}^{r} \alpha_i h_i(\mathbf{x}) + \max H_R(\mathbf{x}, r) \le t\right)}{\sum_{\mathbf{x}} I(l(\mathbf{x}) = -1)} \tag{11}$$

where I(condition) is 1 if the condition is satisfied and 0 otherwise, and $l(\mathbf{x})$ stands for the ground truth label of $\mathbf{x}$. $\sum_{\mathbf{x}} I(l(\mathbf{x}) = -1)$ is the number of all true negative subwindows. The rejection rate $p(r, t)$ in (11) is a function of r and t. However, because Sections III–V-B are not involved in computing the optimal value of the threshold t for the target detection rate, t is set to guarantee 100% detection rate of training positives and can be omitted in expressing $p(r, t)$. That is, $p(r)$ instead of $p(r, t)$ is used in Sections III–V-B.

Obviously, the fraction of negative subwindows classified by using both left and right classifiers is $1 - p$. The criterion for choosing r is to minimize the overall computation cost f consisting of the cost f^L of computing $H_L(\mathbf{x}, r)$ in (9) and the cost f^R of computing both $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ in (10).

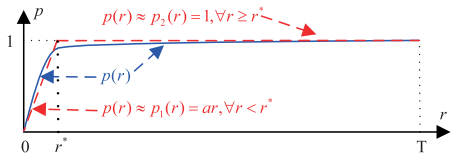


Fig. 2. Representative form of function $p(r)$ which can be simplified as combination of two linear functions: $p_1(r) = ar$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$.

Suppose that all the weak classifiers have the same computation complexity. Then the computation cost is determined by the number of weak classifiers. A fact is that f_1^L grows with p and r

$$f_1^L(r, p) = p(r)(r + c) \tag{12}$$

and f_1^R grows with $1 - p$ and T

$$f_1^R(r, p) = (1 - p(r))(T + 2c). \tag{13}$$

In (12) and (13), c is the computation cost of checking that either inequality (9) or inequality (10) holds. Usually, the computation cost C of a weak classifier is bigger than c. Let $C = 1$, then $c < 1$. Note that c is not involved in computing $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$. Though c is relatively small compared with C, it is not negligible for the case, where the weak classifier (e.g., aggregated channel features (ACF) [34]) is relatively simple.

The goal is to minimize the following object function:

$$f_1(r, p) = f_1^L(r, p) + f_1^R(r, p) = p(r)(r + c) + (1 - p(r))(T + 2c). \tag{14}$$

To solve this optimization problem, it is necessary to reveal the relationship between r and p. The parameters r and p are correlated and the correlation can be expressed as a function $p(r, \max H_R(\mathbf{x}, r))$.

As $\max(H_R(\mathbf{x}, r))$ (its upper bound is $\sum_{i=r+1}^{T} \alpha_i$) decreases with r, a larger number of negative subwindows will be rejected by (9). It is straightforward that the fraction p of negative subwindows satisfying (9) grows with r. Experimental results also show that p monotonically increases with r. The relationship between p and r is nonlinear. Fig. 2 illustrates a typical trend of how p varies with r. It can be seen that p grows quickly from 0 to the value (e.g., 0.99) close to 1 when r changes from 1 to a small value r^* (e.g., 10). But p becomes stable when r is larger than r^*. The reason is that the first r^* weak classifiers h_i play much more important role than the rest weak classifiers.

Mathematically, r^* is defined as the minimum r which satisfies $p(r) \approx 1$ or equivalently $1 - p(r) \leq \varepsilon$ with ε being a small number (e.g., 0.01)

$$r^* = \arg\min_r \{r | 1 - p(r) \leq \varepsilon\}. \tag{15}$$

We call r^* the saturation point of $p(r)$.

Though $p(r)$ is in fact a high-order curve, it can be well modeled by combining two linear functions: $p_1(r) = ar$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$ (see Fig. 2). Because T is a large number, $r^* \ll T$ holds.

It is reasonably assumed that the function $p(r)$ satisfies the following conditions:

$$p(r_1) < p(r_2), \text{ if } r_1 < r_2 \tag{16}$$
$$p'(r_1) > p'(r_2) \geq 0, \text{ if } r_1 < r_2 \tag{17}$$
$$p(T) = 1 \tag{18}$$
$$p(0) = 0 \tag{19}$$
$$p'(T) = 0 \tag{20}$$
$$p'(0) \gg 0 \tag{21}$$
$$p'(1) \gg 0. \tag{22}$$

Inequality (16) states the monotonicity of $p(r)$. Inequality (17) tells that the slope of $p(r)$ decreases with r. Equation (20) shows that the slope is zero at $r = T$ while it is extremely large at $r = 1$. It is noted that (16)–(22) will be used as assumption of the theorems of the proposed methods.

According to Fig. 2, $p(r)$ has the following properties:

$$r^* \ll T, \text{ as } T \text{ is a large number} \tag{23}$$
$$p(r) \approx p_1(r) = ar, \text{ if } r < r^* \tag{24}$$
$$p(r) \approx p_2(r) = 1, \text{ if } r \geq r^* \tag{25}$$

which will be used as assumption of Theorem 2.

After each pairs of (r, p) are known, the value of f_1 can be obtained. Theorem 1 tells that there exists a unique minimization solution.

Theorem 1: $f_1(r) = p(r)(r + c) + (1 - p(r))(T + 2c)$ has a unique minimum solution r_1. Moreover, $f_1(r)$ monotonically decreases with r until $r = r_1$ and then increases with r.

Proof: We first prove the existence of the minimum solution and then give the evidence of the uniqueness of the minimum solution.

Existence: For the sake of notation simplicity, we omit p and write $f_1(r, p)$ as $f_1(r)$:

$$f_1(r) = p(r)(r + c) + (1 - p(r))(T + 2c)$$

The derivative of $f_1(r)$ is

$$f_1'(r) = p'(r)(r - T - c) + p(r).$$

Consider the value of the derivative $f_1'(r)$ when r approaches 0

$$\lim_{r \to 0} f_1'(r) = f_1'(0) = p'(0)(0 - c - T) + p(0). \tag{26}$$

Because $p(0) = 0$ [i.e., (19)] and $p'(0) \gg 0$ [i.e., (21)]. Therefore, it holds

$$\lim_{r \to 0} f_1'(r) = f_1'(0) = -p'(0)(T + c) < 0. \tag{27}$$

Now, consider the value of the derivative $f_1'(r)$ when r approaches T

$$\lim_{r \to T} f_1'(r) = f_1'(T) = p(T) - p'(T)c \approx 1 - 0 > 0. \tag{28}$$

Because $\lim_{r \to 0} f_1'(r) < 0$, $\lim_{r \to T} f_1'(r) > 0$, and $f_1'(r)$ is continuous function, there exists a $r_1 \in [1, T]$ such that $f_1'(r_1) = 0$. The r_1 is at least a local minimum, which shall be the global minimum if the local minimum is unique.

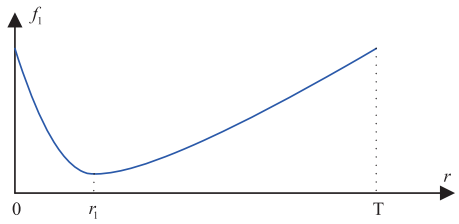


Fig. 3. Representative form of $f_1(r)$.

Uniqueness (Proof by Contradiction): Suppose that there are two local minimums r_1 and r_2 with $r_1 < r_2$. Then it holds that $f_1'(r_1) - f_1'(r_2) = 0$.

Now, investigate the value of $f_1'(r_1) - f_1'(r_2) = [p(r_1) - p(r_2)] - [p'(r_1)(T + c - r_1) - p'(r_2)(T + c - r_2)]$ if $r_1 < r_2$ is true

$$\begin{aligned} &\because r_1 < r_2 \\ &\therefore p'(r_1) > p'(r_2) > 0,\ T + c - r_1 > T + c - r_2 > 0 \\ &\quad p(r_1) < p(r_2) \\ &\therefore f'(r_1) - f'(r_2) < 0. \end{aligned}$$

This contradicts with $f'(r_1) - f'(r_2) = 0$, Therefore, $r_1 < r_2$ is wrong.

Similarly, we can prove that $r_1 > r_2$ is wrong. Consequently, $r_1 = r_2$ is true, meaning a unique solution. ■

Fig. 3 shows a representative form of $f_1(r)$ where a unique minimum solution exists.

Theorem 2: Let r^* be a saturation point of $p(r)$ [see (15)] and assume that $p(r)$ can be modeled by combining $p_1(r) = ar$ where $r < r^*$ with $p_2(r) = 1$ where $r \geq r^*$ (see Fig. 2 for illustration). Then the saturation point r^* is the optimal minimum solution $r_1 = \arg\min_r f_1(r)$.

Proof: See the supplementary materials. ■

Theorem 2 can be intuitively explained as follows. According to the assumption expressed as (25), no samples are rejected after the saturation point, so it is straightforward that the saturation point corresponds to the optimal minimum solution.

IV. Proposed Method: Local-Minimum-Based Multistage Cascade

In this section, we extend two-stage cascade learning to three-stage and multistage cascade learning.

A. Testing Stage

From (5), one can see that two-stage cascade is obtained by splitting $H(\mathbf{x}, r)$ into $H_L(\mathbf{x}, r_1)$ and $H_R(\mathbf{x}, r_1)$ where r_1 is the optimal r [i.e., $r_1 = \arg\min_r f_1(r)$]. We add a superscript "1" to H_L and H_R so that one explicitly knows that $H_L^1(\mathbf{x}, r_1)$ and $H_R^1(\mathbf{x}, r_1)$ correspond to stage 1. Multistage cascade is obtained by iteratively splitting the right classifier H_R^i.

As shown in Fig. 4, the subwindows not rejected by stage 1 are fed to stage 2. The second stage is obtained by further dividing the right classifier $H_R^1(\mathbf{x}, r_1)$ into two parts at the partition point r_2 ($r_2 > r_1$)

$$H_R^1(\mathbf{x}, r_1) = H_L^2(\mathbf{x}, r_2) + H_R^2(\mathbf{x}, r_2) \tag{29}$$

$$H_L^2(\mathbf{x}, r_2) = \sum_{i=r_1+1}^{r_2} \alpha_i h_i(\mathbf{x}) \tag{30}$$

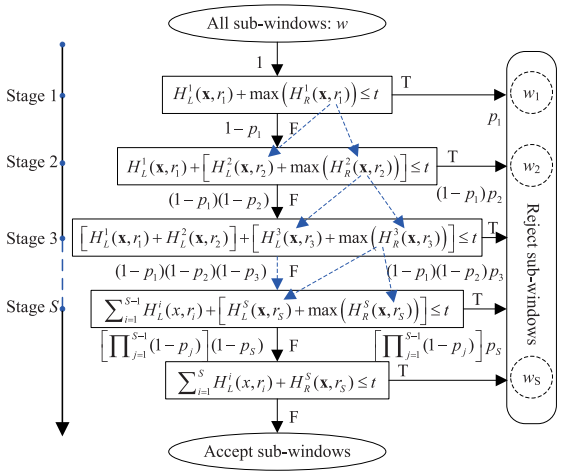


Fig. 4. Block diagram of local-minimum-based multistage cascade AdaBoost.

$$H_R^2(\mathbf{x}, r_2) = \sum_{i=r_2+1}^{T} \alpha_i h_i(\mathbf{x}). \tag{31}$$

In stage 2, the subwindows are rejected if the following inequality holds:

$$H_L^1(\mathbf{x}, r_1) + \left[H_L^2(\mathbf{x}, r_2) + \max\left(H_R^2(\mathbf{x}, r_2)\right)\right] \leq t. \tag{32}$$

Equation (32) is equivalent to

$$H_L(\mathbf{x}, r_2) + \max(H_R(\mathbf{x}, r_2)) \leq t \tag{33}$$

because

$$H_L(\mathbf{x}, r_2) = H_L^1(\mathbf{x}, r_1) + H_L^2(\mathbf{x}, r_2). \tag{34}$$

But (33) is more time-consuming than (32) because r_2 ($r_2 > r_1$) weak classifiers are used to compute $H_L(\mathbf{x}, r_2)$ in (33) whereas in (32) $H_L^1(\mathbf{x}, r_1)$ has been computed in stage 1 and $H_L^2(\mathbf{x}, r_2)$ can be efficiently computed using as small as $r_2 - r_1$ weak classifiers where $H_L^1(\mathbf{x}, r_1)$ can be reused in stage 2.

Analogously, the right classifier in stage $i - 1$ can be represented by the left and right classifiers in stage i

$$H_R^{i-1}(\mathbf{x}, r_{i-1}) = H_L^i(\mathbf{x}, r_i) + H_R^i(\mathbf{x}, r_i). \tag{35}$$

The block diagram of the multistage cascade is shown in Fig. 4, where the rejection rate p_i is the ratio of subwindows rejected in stage i. In stage 1, p_1 fraction of subwindows is directly rejected and $1 - p_1$ fraction of subwindows is fed to stage 2. Among the $1 - p_1$ fraction of subwindows, p_2 fraction is rejected by stage 2, and $1 - p_2$ fraction is considered as positive-class candidates and therefore fed to stage 3. This means that $(1 - p_1)(1 - p_2)$ fraction of total w subwindows is to be classified by stage 3. p_1 is defined as (11). Because p_i in stage i is dependent on p_{i-1} in stage $i - 1$ ($i > 1$), we explicitly express p_i as $p(r_i|r_{i-1})$ when necessary. Specifically, the rejection rate $p(r_i|r_{i-1})$ is defined as

$$p\left(r_i \middle| r_{i-1}\right) = \frac{I\left(\sum_{k=1}^{r_i} \alpha_k h_k(\mathbf{x}) + \max(H_R(\mathbf{x}, r_i)) \leq t\right)}{I\left(\sum_{k=1}^{r_{i-1}} \alpha_k h_k(\mathbf{x}) + \max(H_R(\mathbf{x}, r_{i-1})) > t\right)}. \tag{36}$$

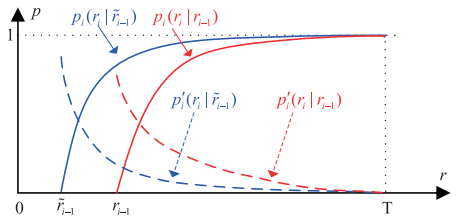


Fig. 5. Form of $p_i(r_i|r_{i-1})$ and its properties. If $\widetilde{r}_{i-1} < r_{i-1}$, then $p_i(r_i|\widetilde{r}_{i-1}) > p_i(r_i|r_{i-1})$ and $p_i'(r_i|\widetilde{r}_{i-1}) < p_i'(r_i|r_{i-1})$.

Fig. 5 shows two representative curves of $p(r_i|r_{i-1})$. The properties of $p(r_i|r_{i-1})$ are summarized as follows:

$$p'\left(r_i\middle|r_{i-1}\right) \geq 0 \tag{37}$$

$$\lim_{r_i \to r_{i-1}} p\left(r_i\middle|r_{i-1}\right) = 0 \tag{38}$$

$$\lim_{r_i \to T} p\left(r_i\middle|r_{i-1}\right) = 1 \tag{39}$$

$$p\left(r_i\middle|\widetilde{r}_{i-1}\right) > p\left(r_i\middle|r_{i-1}\right), \text{ if } \widetilde{r}_{i-1} < r_{i-1} \tag{40}$$

$$p'\left(r_i\middle|\widetilde{r}_{i-1}\right) < p'\left(r_i\middle|r_{i-1}\right), \text{ if } \widetilde{r}_{i-1} < r_{i-1}. \tag{41}$$

We give a theoretical guarantee (i.e., Theorem 3) that adding a stage results in reduction of computation cost if the certain conditions are satisfied.

Theorem 3: Let $r_1, \ldots, r_S$ define an $S+1$ stage cascade structure whose computation cost is $f_S(r_1, \ldots, r_S)$

$$f_S(r_1, \ldots, r_S) = \sum_{i=1}^{S} f_i^L(r_1, \ldots, r_i) + f_S^R(r_S) \tag{42}$$

where

$$f_i^L = \left[\prod_{j=1}^{i-1}\left(1 - p_j\left(r_j\middle|r_{j-1}\right)\right)\right] p_i(r_i|r_{i-1})(r_i + ic) \tag{43}$$

$$f_S^R = \left[\prod_{j=1}^{S}\left(1 - p_j\left(r_j\middle|r_{j-1}\right)\right)\right] (T + (S+1)c). \tag{44}$$

Let $r_1, \ldots, r_{S-1}$ define an S stage cascade structure whose computation cost is $f_{S-1}(r_1, \ldots, r_{S-1})$

$$f_{S-1}(r_1, \ldots, r_{S-1}) = \sum_{i=1}^{S-1} f_i^L(r_1, \ldots, r_i) + f_{S-1}^R(r_{S-1}). \tag{45}$$

If $p_S(r_S|r_{S-1}) > c/(T + c - r_S)$, then we have

$$f_{S-1}(r_1, \ldots, r_{S-1}) > f_S(r_1, \ldots, r_{S-1}, r_S). \tag{46}$$

Proof:

$$\begin{aligned}
&\because f_{S-1}(r_1, \ldots, r_{S-1}) - f_S(r_1, \ldots, r_{S-1}, r_S) \\
&\quad = f_{S-1}^R(r_{S-1}) - f_S^L(r_1, \ldots, r_S) - f_S^R(r_S) \\
&\quad = \left[\prod_{j=1}^{S-1}(1 - p_j)\right]\left[p_S\left(r_S\middle|r_{S-1}\right)(T + c - r_S) - c\right] \\
&\because 1 - p_j > 0, \ p_S > 0 \\
&\therefore \text{if } p_S\left(r_S\middle|r_{S-1}\right) > c/(T + c - r_S), \text{ then } f_{S-1} > f_S.
\end{aligned}$$

■

Note that if the computation cost c is omitted, then $f_{S-1} > f_S$ as long as $p_S(r_S|r_{S-1}) > 0$. In this case, it is optimal that each stage contains a new weak classifier (i.e., the case $S = T$, $r_1 = 1, r_2 = 2, \ldots, r_T = T$). Since $c \neq 0$ in practice, it is necessary to let $S < T$ and find a way to search the optimal values of $r_1, \ldots, r_S$.

B. Training Stage: How to Select Optimal r_i

Now, we describe the training stage of the proposed method.

1) Existence and Uniqueness: Investigating Fig. 4, one can find that the cascade structure is completely determined once $r_1, \ldots, r_S$ are known. Therefore, the main task of the training stage is to find the optimal $r_1, \ldots, r_S$.

The r_1 in stage 1 is obtained by the method in Section III-B. Given r_1, we learn the best r_2

$$r_2 = \arg\min_r f_2(r_1, r) = \arg\min_r f_2(r|r_1). \tag{47}$$

Similar to the proof of Theorem 1, it can be proved that $f_2(r|r_1)$ has a unique solution.

Corollary 1: $f_2(r|r_1)$ has a unique solution.

Proof: See the supplementary materials. ■

More generally, as a corollary of Theorem 1, we have the following theorem.

Theorem 4: $\min_r f_i(r_1, \ldots, r_{i-1}, r) = \min_r f_i(r|r_1, \ldots, r_{i-1})$ has a unique minimum solution $r_i \in [r_{i-1}, T]$. Moreover, $f_i(r|r_1, \ldots, r_{i-1})$ monotonically decreases with r until $r = r_i$ and then increases with r.

Note that $p(0) = 0$ is used in proving Theorem 1 and the term corresponding to $p(0) = 0$ in proving Theorem 4 and Corollary 1 is $p_i(r_{i-1}|r_{i-1})$. $p_i(r_{i-1}|r_{i-1})$ means that the samples accepted by the r_{i-1} weak classifiers can not be rejected by the r_{i-1} weak classifiers themselves.

Theorem 4 implies that $f_i(r|r_1, \ldots, r_{i-1})$ has the similar form as the curve in Fig. 3.

2) Efficient Search: The search range of r_i is (r_{i-1}, T). However, because $f_i(r|r_1, \ldots, r_{i-1})$ monotonically decreases with r until $r = r_i$ and then increases with r. To find the unique minimum solution, r is increased from r_{i-1} with a small step and stops at the value once $f_i(r|r_1, \ldots, r_{i-1})$ no longer decreases. Therefore, the practical range is less than (r_{i-1}, T).

The search range can be further reduced according to the following increasing phenomenon.

Theorem 5: If $r_i = \arg\min_{r_{i-1}<r<T} f_i(r|r_1, \ldots, r_{i-1})$, $r_{i-1} = \arg\min_{r_{i-2}<r<T} f_{i-1}(r|r_1, \ldots, r_{i-2})$, $r_{i+1} = \arg\min_{r_i<r<T} f_{i+1}(r|r_1, \ldots, r_i)$, and $2r_i - r_{i-1} < T$, then it holds

$$r_i - r_{i-1} \leq r_{i+1} - r_i \tag{48}$$

$$r_{i+1} \geq 2r_i - r_{i-1}. \tag{49}$$

We define $r_0 = 0$, so we have

$$r_2 \geq 2r_1. \tag{50}$$

Proof: See the supplementary materials. ■

Fig. 6 illustrates the nature of Theorem 5, where the objective functions (solid curves) and estimated optimal solutions at saturation points are shown. The dashed curves are

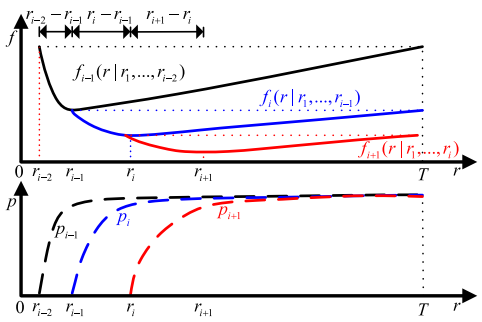


Fig. 6. Illustration of Theorem 5. The solid curves are computational cost and the dashed ones are rejection rates.

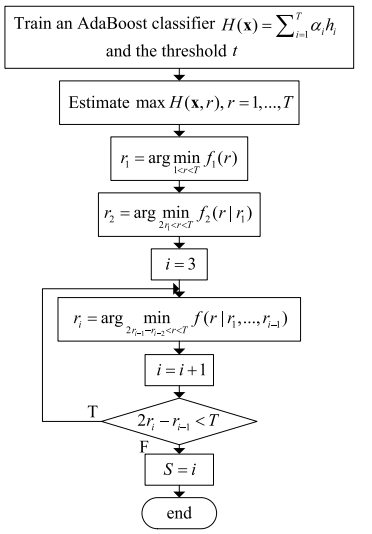


Fig. 7. Local-minimum-based multistage cascade learning.

the rejection rates. The relationship of r_{i-1}, r_i, and r_{i+1} is $r_{i-1}-r_{i-2} < r_i-r_{i-1} < r_{i+1}-r_i$ (i.e., $\Delta r_{i-1} < \Delta r_i < \Delta r_{i+1}$).

According to Theorem 5, if $r_i = \arg\min_{r_{i-1}<r<T} f_i(r|r_1,\ldots,r_{i-1})$ and $r_{i-1} = \arg\min_{r_{i-2}<r<T} f_i(r|r_1,\ldots,r_{i-2})$ are already known, then the search range for r_{i+1} will be reduced to $[r_i + (r_i - r_{i-1}), T]$ (i.e., $[2r_i - r_{i-1}, T]$) where $r_i - r_{i-1}$ is called increasing step.

The training process is shown in Fig. 7, where the increasing phenomenon is used for efficient minimization.

V. Proposed Method: Joint-Minimum-Based Multistage Cascade

A. Existence and Uniqueness of Jointly Optimal Solution

The method in Section IV is a greedy optimization algorithm because it seeks an optimal r_S on the condition that $(r_1,\ldots,r_{S-1})$ are known and fixed. The objective function is $f_S(r|r_1,\ldots,r_{S-1})$. In this section, we give an algorithm for jointly seeking the optimal $(r_1,\ldots,r_S)$ that globally minimizes the objective function $f(r_1,\ldots,r_S)$ instead of $f_S(r|r_1,\ldots,r_{S-1})$. That is, the goal of joint optimization is to find $(r_1^*,\ldots,r_S^*) = \arg\min_{r_1,\ldots,r_S} f_S(r_1,\ldots,r_S)$.

For the sake of clarity, we start with establishing a globally optimal three-stage cascade structure. The globally optimal cascade structure with more than three stages will be extended from the three-stage one.

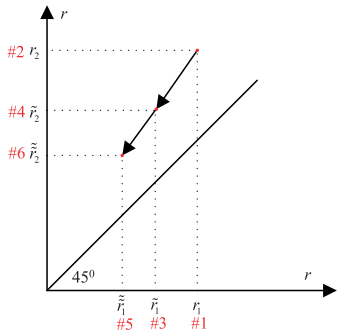


Fig. 8. Illustration of the intermediate values obtained by Algorithm 2. r_1 and r_2 are the outputs of initialization. The sequence of the stage parameters are updated in the following turn: $r_1 \to r_2 \to \tilde{r}_1 \to \tilde{r}_2 \to \tilde{\tilde{r}}_1 \to \tilde{\tilde{r}}_2$ with $\tilde{\tilde{r}}_1 < \tilde{r}_1 < r_1$ and $\tilde{\tilde{r}}_2 < \tilde{r}_2 < r_2$.

The goal of jointly optimal three-stage cascade learning aims at finding $(r_1^*, r_2^*) = \arg\min_{r_1,r_2} f_2(r_1, r_2)$.

Obviously, if both $f_2(r|r_1) = f_2(r_1, r)$ and $f_2(r|r_2) = f_2(r, r_2)$ have unique minimization solutions, then $f_2(r_1, r_2)$ has unique minimization solutions. $f_2(r|r_1)$ means the objective function of a three-stage cascade where the parameter r_1 of stage 1 is known and the parameter r of stage 2 is a unknown variable. $f_2(r|r_2)$ stands for the situation, where the parameter r_2 of stage 2 is known and the parameter r of stage 1 is a unknown variable. The theorems related to the joint optimization are as follows.

Theorem 6: $\min_r f_2(r, r_2) = \min_r f_2(r|r_2)$ has a unique minimum solution r_1.

Proof: See the supplementary materials. ■

Theorem 6 tells that if the information of stage 2 is given, then one can find an optimal parameter r for stage 1 so that the computation cost f_2 of the final three-stage cascade is minimized.

Theorem 7: $f_2(r_1, r_2)$ has a unique minimum solution (r_1^*, r_2^*).

Proof: See the supplementary materials. ■

It is straightforward to generalize Theorem 7 to the following theorem.

Theorem 8: $f_i(r_1,\ldots,r_i)$ has a unique minimum solution $(r_1^*,\ldots,r_i^*)$.

B. How to Search the Jointly Optimal Solutions

1) Algorithm: Theorem 8 guarantees the existence and uniqueness of jointly optimizing the stages of a cascade. In this section, we give algorithms (i.e., Algorithms 2 and 3) for searching the solution and then justify the algorithms in theory. We start with the algorithm for optimizing a three-stage cascade and then generalize it to multistage one.

The task of jointly optimizing a three-stage cascade can be expressed as $(r_1^*, r_2^*) = \arg\min_{r_1,r_2} f_2(r_1, r_2)$. The idea of our optimization method is shown in Algorithm 2.

The proposed Algorithm 2 is an alternative optimization procedure. In the initialization step, the solution r_1 of the two-stage cascade learning is searched in the largest range

Algorithm 2 Globally Optimal Three-Stage Cascade Learning

Input:

Strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$, its threshold t;
A set of true negative subwindows$\{\mathbf{x}|l(\mathbf{x}) = -1\}$;

Output:

$(r_1^*, r_2^*) = \arg\min_{r_1,r_2} f_2(r_1, r_2)$;

1: **Initialization**
2: Search the optimal solution r_1 of $f_1(r)$ for stage 1 in the range of $(1, T)$: $r_1 = \arg\min_{1<r<T} f_1(r)$.
3: Given r_1, search the optimal solution r_2 of $f_2(r|r_1)$ for stage 2 in the range of $[2r_1, T)$: $r_2 = \arg\min_{2r_1 \le r < T} f_2(r|r_1)$. See Theorem 5 for the reason of $r \ge 2r_1$.
4: **repeat**
5: Given r_2, search the optimal solution $\tilde{r}_1$ of $f_2(r|r_2)$ in the range of $1 < r \le r_1$ for stage 1: $\tilde{r}_1 = \arg\min_{1<r\le r_1} f_2(r|r_2)$. Note that $\tilde{r}_1 < r_1$ (see Theorem 9). An efficient search strategy is decreasing r from r_1 step by step until $f_2(r|r_2)$ does not decrease. $f \leftarrow f_2(r|r_2)$.
6: Given $\tilde{r}_1$, search the optimal solution $\tilde{r}_2$ of $f_2(r|\tilde{r}_1)$ for stage 2 in the range of $\tilde{r}_1 < r \le r_2$: $\tilde{r}_2 = \arg\min_{\tilde{r}_1<r\le r_2} f_2(r|\tilde{r}_1)$. Note that $\tilde{r}_2 \le r_2$ (see Theorem 12). An efficient search strategy is decreasing r from r_2 step by step until $f_2(r|\tilde{r}_1)$ does not decrease. $\tilde{f} \leftarrow f_2(r|\tilde{r}_1)$.
7: Update $r_1 \leftarrow \tilde{r}_1$, $r_2 \leftarrow \tilde{r}_2$.
8: **until** $f - \tilde{f} \le \mu$
9: **return** $r_1^* \leftarrow \tilde{r}_1$, $r_2^* \leftarrow \tilde{r}_2$.

$1 < r < T : r_1 = \arg\min_{1<r<T} f_1(r)$. The value of r_1 is shown in Fig. 8, where "#1" means that r_1 is obtained first. The obtained r_1 is used as the upper bound of the searching range for the better solution $\tilde{r}_1$ in line 5 of Algorithm 2. After r_1 is given, line 3 of Algorithm 2 searches the optimal solution r_2 of $f_2(r|r_1)$ for stage 2 in the range of $2r_1 \le r < T : r_2 = \arg\min_{2r_1\le r<T} f_2(r|r_1)$. Based on (50), the search range starts from $2r_1$. The value of r_2 is shown in Fig. 8, where "#2" means that r_2 is the second value obtained by Algorithm 2.

In line 5 of Algorithm 2, r_2 is given and the task is to search the optimal solution $\tilde{r}_1$ of $f_2(r|r_2)$ in the range of $1 < r \le r_1$ for stage 1: $\tilde{r}_1 = \arg\min_{1<r\le r_1} f_2(r|r_2)$. Because $r_1 \ll T$, the search range $1 < r \le r_1$ is much smaller than the one (i.e., $1 < r < T$) in line 2. Theorem 9 guarantees $\tilde{r}_1 \le r_1$ for the first round of iteration. $\tilde{r}_1$ is the third value obtained by Algorithm 2 which is shown near "#3" in Fig. 8. Experimental results and intuitive analysis show that the absolute distance $|\tilde{r}_1 - r_1|$ from $\tilde{r}_1$ to r_1 is much smaller than the absolute distance $|1 - \tilde{r}_1|$ from 1 to $\tilde{r}_1$. Therefore, the search strategy of decreasing r from r_1 step by step until $f_2(r|r_2)$ does not decrease is more efficient than the one of increasing r from 1 step by step until $f_2(r|r_2)$ does not decrease.

In line 6 of Algorithm 2, $\tilde{r}_1$ is given and the task is to search the optimal solution $\tilde{r}_2$ of $f_2(r|\tilde{r}_1)$ in the range of $\tilde{r}_1 < r \le r_2$ for stage 2: $\tilde{r}_2 = \arg\min_{\tilde{r}_1<r\le r_2} f_2(r|\tilde{r}_1)$. Because $r_2 < T$, the upper bound of the search range is much smaller than the one (i.e., T) in line 3. Moreover, as iteration continues, the updated r_2 becomes smaller and so the upper bound of search range for $\tilde{r}_2$ becomes smaller too. Theorem 10 guarantees $\tilde{r}_2 \le r_2$. The value of $\tilde{r}_2$ is shown in Fig. 8 which is "#4" obtained by Algorithm 2. Experimental results and intuitive analysis show that the absolute distance $|\tilde{r}_2 - r_2|$ from $\tilde{r}_2$ to r_2 is much smaller than the absolute distance $|\tilde{r}_1 - \tilde{r}_2|$ from $\tilde{r}_1$ to $\tilde{r}_2$; the search strategy of decreasing r from r_2 step by step until $f_2(r|\tilde{r}_1)$ does not decrease is more efficient than the one of increasing r from $\tilde{r}_1$ step by step until $f_2(r|\tilde{r}_1)$ does not decrease.

In the second round of iteration, because $\tilde{r}_2 \le r_2$, the parameter value $\tilde{\tilde{r}}_1$ for stage 1 is obtained and shown in Fig. 8 with a label "#5." According to Theorem 11, it is true that $\tilde{\tilde{r}}_1 \le \tilde{r}_1$. Subsequently, the parameter value $\tilde{\tilde{r}}_2$ for stage 2 is obtained and shown in Fig. 8 with a label "#6." According to Theorem 10, it is true that $\tilde{\tilde{r}}_2 \le \tilde{r}_2$.

The iteration stops if the difference between the value f of objective function in line 5 of Algorithm 2 and the one $\tilde{f}$ in line 6 of Algorithm 2 is equal to or smaller than the threshold $\mu \ge 0$.

Decreasing Phenomenon: Fig. 8 illustrates an interesting phenomenon.

1) Once a new stage 2 is added, the parameter r_1 of stage 1 should be updated by decreasing r_1 to a smaller number $\tilde{r}_1$ so that the computation cost is minimized.
2) Once the number of stages is fixed, the parameter for each stage decreases gradually as iteration proceeds.

2) Justification of the Algorithm: Theorems 9–12 are given to theoretically interpret the so-called decreasing phenomenon and justify Algorithm 2. Theorem 9 implies that the parameter r_1 of stage 1 should be updated by decreasing to a smaller number when the parameter r_2 of stage 2 is fixed.

Theorem 9: $\tilde{r}_1 = \arg\min_r f_2(r|r_2) \le \arg\min_r f_1(r) = r_1$, where $r_2 > r_1$.

Proof: See the supplementary materials. ■

As a lemma of Theorem 9, we have the following theorem.

Theorem 10: If $(r_1^{i*}, \ldots, r_i^{i*}) = \arg\min_{r_1,\ldots,r_i} f_i(r_1, \ldots, r_i)$ and $(r_1^{*(i+1)}, \ldots, r_i^{*(i+1)}, r_{i+1}^{*(i+1)}) = \arg\min_{r_1,\ldots,r_i,r_{i+1}} f_i(r_1, \ldots, r_i, r_{i+1})$, then $r_j^{*(i+1)} \le r_j^{*i}$, $j = 1, \ldots, i$.

As a generalized version of Theorem 9, Theorem 10 tells that once a new stage $i+1$ is added, all the optimal parameters of the existing stages $1, \ldots, i$ should be updated and decreased so that the computation cost is minimized.

Theorem 11: If $\tilde{r}_2 < r_2$, then $\tilde{r}_1 = \arg\min_r f_2(r|\tilde{r}_2) \le \arg\min_r f_2(r|r_2) = r_1$.

Proof: See the supplementary materials. ■

Theorem 12: If $\tilde{r}_1 < r_1$, then $\tilde{r}_2 = \arg\min_{\tilde{r}_1<r<T} f_2(r|\tilde{r}_1) \le \arg\min_{r_1<r<T} f_2(r|r_1) = r_2$.

Proof: See the supplementary materials. ■

Theorems 9, 11, and 12 can be extended to multistage cascade. Correspondingly, decreasing phenomenon can be generalized to generalized decreasing phenomenon and Algorithm 2 can be generalized to Algorithm 3.

Generalized Decreasing Phenomenon: If the alternative optimization Algorithm 3 is used to find the globally optimal solution $(r_1^{*i}, r_2^{*i}, \ldots, r_i^{*i}) = \arg\min_{r_1,\ldots,r_i} f_i(r_1, \ldots, r_i)$, then it holds the following.

1) Once a new stage $i+1$ is added, all the optimal parameters of the existing stages $1, \ldots, i$ are updated and decreased so that the computation cost is minimized.

Algorithm 3 Globally Optimal Multistage Cascade Learning

Input:

Strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$, its threshold t;

A set of true negative subwindows$\{\mathbf{x}|l(\mathbf{x}) = -1\}$;

Output:

$(r_1^*, \ldots, r_S^*) = \arg\min_{r_1,\ldots,r_S} f_S(r_1, ..., r_S)$ where S+1 is the number of stages in the final cascade structure;

1: Search the optimal solution $r_1^{*(1)}$ of $f_1(r)$ for stage 1 in the range of $1 < r < T$: $r_1^{*(1)} = \arg\min_{1<r<T} f_1(r)$. $f \leftarrow f_1(r_1)$;
2: **for** $i = 2$ to S **do**
3: Initialize the upper bound $r_1^u, \ldots, r_{i-1}^u$ of $r_1, \ldots, r_{i-1}$: $r_j^u \leftarrow r_j^{*(i-1)}$ for $j = 1, \ldots, i-1$;
4: Initialize the upper bound r_i^u of r_i by finding $r_i^u = \arg\min_{r_i \geq 2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}} f_i\left(r_i | r_1^{*(i-1)}, \ldots, r_{i-1}^{*(i-1)}\right)$ with the search range $r_i \geq 2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ and $r_0^{*(1)} \triangleq 0$.
5: $f \leftarrow f_i(r_1^{*(i-1)}, \ldots, r_{i-1}^{*(i-1)})$, $\tilde{f} \leftarrow f_i(r_1^u, \ldots, r_i^u)$.
6: **while** $f - \tilde{f} > \varepsilon$ **do**
7: $f \leftarrow \tilde{f}$;
8: **for** $j = 0$ to i **do**
9: $r_j^* = \arg\min_{r_j \leq r_j^u} f_i(r_j | r_k^u, k \neq j)$;
10: **end for**
11: $\tilde{f} \leftarrow f_i(r_1^*, \ldots, r_i^*)$, $r_j^u \leftarrow r_j^*$.
12: **end while**
13: $r_j^{*i} \leftarrow r_j^u$, $j = 1, \ldots, i$;
14: **end for**
15: **return** $r_i^* \leftarrow r_i^{*S}$, $i = 1, \ldots, S$.

2) Once the number of stages is fixed, the parameter for each stage decreases gradually as iteration proceeds.

In Algorithm 3, $\tilde{f}$ is the objective function after a new stage i is added while f is the one before stage i is added. That is, f is the value of objective function when there are $i-1$ stages. According to Theorem 5, when a new stage i is to be added, the optimal solution r_i can be searched by increasing r_i from $2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ instead of $r_{i-1}^{*(i-1)}$. Because $2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ is much larger than $r_{i-1}^{*(i-1)}$, the search efficiency is very high. The iteration in line 6 of Algorithm 3 stops if the difference between f and $\tilde{f}$ is below a threshold $\varepsilon > 0$, which implies that the algorithm arrives at global minimum solution for S stages.

Fig. 9 shows the classification procedure of the multistage iCascade, where the partition points $(r_1, \ldots, r_S)$ are given by Algorithm 3. The computation cost f_S of iCascade can be estimated by

$$f_S = \sum_{i=1}^{S} (r_i + ic)\left[\prod_{j=1}^{i}\left(1 - p_{j-1}\left(r_{j-1}\right)\right)\right] p_i(r_i) + (T + (S+1)c)\left[\prod_{j=1}^{S+1}\left(1 - p_{j-1}\left(r_{j-1}\right)\right)\right]. \quad (51)$$

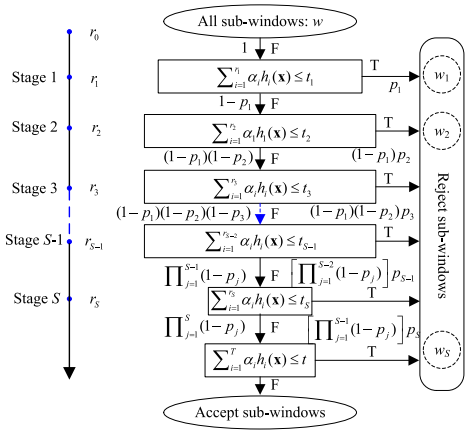


Fig. 9. Classification procedure of the multistage iCascade algorithm.

C. Threshold Learning in iCascade

Once globally optimal partitions $r_1, \ldots, r_S$ for each stage are determined by Algorithm 3, the parameters affecting the computation cost are the thresholds t_i, $i = 1, \ldots, S$. In this section, we give the theorem and algorithm for setting the thresholds $(t_1, \ldots, t_S)$. As stated in Section III-B, the rejection rate $p(r, t)$ is a function of the number r of weak classifiers and the stage threshold t. Because r is fixed, here in after, p is expressed as the function of the threshold t. That is, $p(t)$ is used in Section V-C.

Theorem 13 tells that computation cost f_S monotonically decreases with the increase of t_i and $p_i(t_i)$, $i = 1, \ldots, S$. So computation cost can be reduced by increasing the thresholds under the constraint of minimum-acceptable detection rate.

Theorem 13: f_S monotonically decreases with the increase of t_i and $p_i(t_i)$, $i = 1, \ldots, S$.

Proof: See the supplementary materials. ■

If the detection rate $D = 1$ (i.e., all the positive training samples are correctly classified) is the constraint, then the optimal threshold t_i^* can be expressed as

$$t_i^* = \arg\max t_i, \quad \text{s.t. } d(t_i) = 1, \quad i = 1, \ldots, S \quad (52)$$

which guarantees $D = \prod_{i=1}^{S} d(t_i^*) = 1$. In (52), $d(t_i)$ is the detection rate of stage i defined by

$$d(t_i) = \frac{\sum_{\mathbf{x}} I\left(\sum_{j=1}^{r_i} \alpha_j h_j(\mathbf{x}) > t_i\right)}{\sum_{\mathbf{x}} I(l(\mathbf{x}) = 1)}. \quad (53)$$

It is challenging to choose the optimal thresholds if the expected detection $D < 1$. It is well known that the detection rate D of the system is the product of the detection rate $d(t_i)$ of each stage. A popular way to set $d(t_i)$ is

$$d(t_i) = D^{1/S}, \ i = 1, \ldots, S. \quad (54)$$

However, when the number of stages of iCascade is very large, it holds that $d(t_i) \approx 1$. Such high $d(t_i)$ makes the threshold t_i very small and the corresponding computation cost is very large.

To deal with the above problem, we propose to use Algorithm 4 for threshold learning. The initial thresholds are chosen by (52) guaranteeing the initial detection rate D being 1 and the detection rate of each stage being 1 as well. The

corresponding initial computation cost is denoted by f_S. The main issue is to select which stage to increase its initial threshold by a small step Δt_i. In our algorithm, the derivative f'_S of the computation cost f_S against detection rate D is computed by

$$f'_S(i) \approx \Delta f_S / \Delta D_i \tag{55}$$

where ΔD_i is the variation of the system detection rate. Note that the variation ΔD_i is caused by changing t_i to $t_i + \Delta t_i$ while the thresholds t_k of other stages (i.e., $k \neq i$) remain unchanged.

The stage j with the largest derivative is selected and its threshold t_j is then increased by the small step Δt_j

$$j = \arg\max_i \frac{\Delta f_S}{\Delta D_i} \tag{56}$$

$$t_j \leftarrow t_j + \Delta t_j \tag{57}$$

with the thresholds of the stages (i.e., $i \neq j$) unchanged.

Recompute the computation cost f_S and detection rate D after t_j is updated

$$f_S \leftarrow f_S - \Delta f_S \tag{58}$$

$$D \leftarrow D - \Delta D_j. \tag{59}$$

The step Δt_i is small enough to keep the detection rate D bigger than the target detection rate D_o.

As shown in Algorithm 4, the iteration of choosing the most important stage $j = \arg\max_i \Delta f_S / \Delta D_i$, updating its threshold $t_j \leftarrow t_j + \Delta t_j$ and corresponding computation cost $f_S \leftarrow f_S - \Delta f_S$ and detection rate $D \leftarrow D - \Delta D_j$ runs until the updated detection rate D is equal to or smaller than the expected detection rate D_o.

Algorithm 4 Threshold Learning Algorithm for iCascade

Input:

Expected detection rate D_o;
Positve and negative training samples;
Strong classifiers $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$;

Output:

The optimal thresholds t_i of all the S stages;

1: Initialize the thresholds t_i for each stage by $t_i = \arg\max t_i$, s.t. $d(t_i) = 1$, $i = 1, \ldots, S$ so that the system detection rate $D = 1$;
2: Corresponding to the initial thresholds, the initial computation cost of the system is computed by (51) and denoted by f_S;
3: **repeat**
4: For each stage, compute the approximation of the derivative $\Delta f_S / \Delta D_i$ of the computation cost f_S against detection rate D. The variations Δf_S and ΔD_i are caused by changing t_i to $t_i + \Delta t_i$ while the thresholds t_k of other stages (i.e., $k \neq i$) remain unchanged;
5: From all the S stages, choose the stage j with largest derivative $j = \arg\max_i \Delta f_S / \Delta D_i$. Then increase the threshold t_j of the stage j by a small step Δt_j : $t_j \leftarrow t_j + \Delta t_j$;
6: Update the computation cost f_S and detection rate D: $f_S \leftarrow f_S - \Delta f_S$, $D \leftarrow D - \Delta D_j$;
7: **until** $D \leq D_o$
8: **return** the updated thresholds t_i of all the S stages.

VI. Experimental Results

A. Experimental Setup

The experimental results in Sections VI-B and VI-C are obtained from the MIT-CMU frontal face dataset [20], [35] and the experimental results in Section VI-D are obtained from the Caltech pedestrian dataset [36], [37]. The proposed method is compared with Fixed Cascade [20], Recycling Cascade [9], Recycling and Retracting Cascade [9], and Soft-Cascade [22]. Haar features [20], ACF features [34], and non-neighboring and neighboring features with level 4 decision trees (NNNF-L4) features [18] are employed.

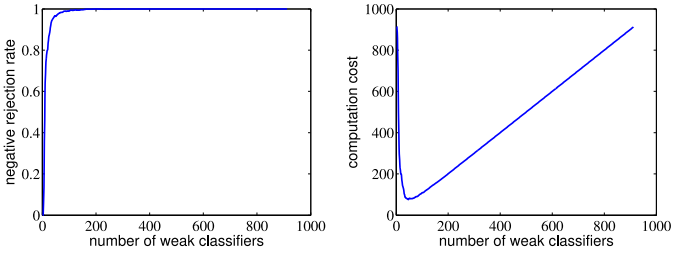
In order to test on the MIT-CMU frontal face dataset, the positive training dataset consists of about 20 000 normalized face images and the negative training dataset contains 5000 nonface images. They are collected from Web sites. The testing dataset consists of 130 gray-scale images containing 507 labeled frontal faces. The Caltech pedestrian dataset consists of 11 videos with the first 6 videos used for training and the last 5 ones for testing. The Caltech training images are generated by sampling a frame from every 30 frames of the training videos and the Caltech 10× training images are generated by sampling a frame from every 3 frames of the training videos. The testing dataset consists of 4024 frames among which there are 1014 positives.

Fixed Cascade is proposed by Viola and Jones [20]. "Fixed" means that the detection rate d_i and the false positive rate f_i of each stage are fixed. If the target detection rate of the cascade is D_o, the target false positive rate is F, and the number of the stages is N, then $d_i = D_o^{1/N}$ and $f_i = F^{1/N}$. In Recycling Cascade, the score from the previous strong classifier stages serves as a starting point for the score of the new strong classifier stage. The detection rate d_i and false positive rate f_i in each stage are same as Fixed Cascade. The benefit of Recycling Cascade is the reduction of the number of the weak classifiers in the strong classifier stages and the reduction of the computation cost. A useful effect of Recycling Cascade is that the last stage of cascade can serve as an accurate strong classifier. Recycling and Retracting Cascade chooses a threshold after each weak classifier produced by Recycling Cascade to reject some negative subwindows. To set these thresholds, it evaluates each score on the set of the positive examples and chooses the minimum score as the threshold so that all the positive examples in the set can pass all the weak classifiers. Soft-Cascade and Recycling and Retracting Cascade are similar to iCascade in the sense of accumulatively using the responses of stages. Thus, Recycling Cascade, Recycling and Retracting Cascade, Soft-Cascade, and iCascade are all the embedded cascade.

A strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ is considered input of iCascade. By using the technology of bootstrap, the strong classifier is obtained by standard AdaBoost without designing of cascade structure. If the detected rectangle and the ground-truth rectangle are at least 50% of overlap, we call the detected rectangle a correct detection.

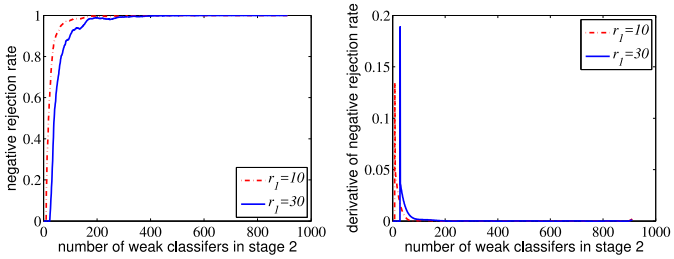
TABLE I
COMPUTATION COST f OF THE ALGORITHM IN FIG. 4 VARIES WITH THE NUMBER $(i+1)$ OF STAGES

$i+1$	2	3	4	5	6	7	8	9
f	74.42	52.87	52.16	52.15	52.14	52.14	52.14	52.14



(a) (b)

Fig. 10. (a) Negative rejection rate $p(r)$. (b) Computation cost $f(r)$.



(a) (b)

Fig. 11. (a) Some properties of $p(r_2|r_1)$. (b) Derivative of $p(r_2|r_1)$.

B. Intermediate Results of iCascade on the MIT-CMU Face Dataset

Some intermediate results are shown in this section. These results show the rationality of the assumptions and the correctness of the proposed theorems and algorithms. Haar features are employed. The computation cost means the number of average weak classifiers per window.

1) Local-Minimum-Based Cascade: In Section III, the regular strong AdaBoost classifier is divided into $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ to reject some negative subwindows earlier, and the key problem is to determine an optimal r to minimize the computation cost. To solve this problem, it is necessary to reveal the relationship between r and the negative rejection rate p.

In this part, with the MIT-CMU training dataset described in Section VI-A, we train a regular strong AdaBoost classifier and split it into two parts by r, which varies from 1 to T. In the case that detection rate is fixed at 1, Fig. 10(a) shows that the negative rejection rate p increases with r. p first grows quickly from 0 to 0.96 when r changes from 1 to a small value $r^* = 80$, and then becomes stable when r is larger than r^*. Thus, we can model $p(r)$ by combining two linear functions: $p_1(r) = 0.012r$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$. Fig. 10(a) demonstrates the rationality of (16)–(25). Fig. 10(b) shows that the computation cost f first decreases and then increases with r, and the unique minimum is nearby r^*. Fig. 10(b) experimentally proves the correctness of Theorems 1 and 2.

When we split the regular strong AdaBoost classifier into $H_L(\mathbf{x}, r_1)$ and $H_R(\mathbf{x}, r_1)$, the subwindows not rejected by stage 1 are fed to stage 2. Then, we can divide $H_R(\mathbf{x}, r_1)$ into two parts to form a 3-stage cascade. In this process, we should know some properties of the negative rejection rate of stage 2 [i.e., $p(r|r_1)$]. Fig. 11(a) shows how $p(r|r_1)$ changes with r, where the curves of $p(r|r_1)$ when $r_1 = 10$ and $r_1 = 30$ are given, respectively. $p(r|r_1)$ has the similar characteristics to $p(r)$. Fig. 11(b) shows how the derivative of $p(r|r_1)$ changes with r. Obviously, when $\widetilde{r}_1 < r_1$, $p(r_2|\widetilde{r}_1) > p(r_2|r_1)$ and $p'(r_2|\widetilde{r}_1) < p'(r_2|r_1)$. Fig. 11 directly supports the correctness of (37)–(41).

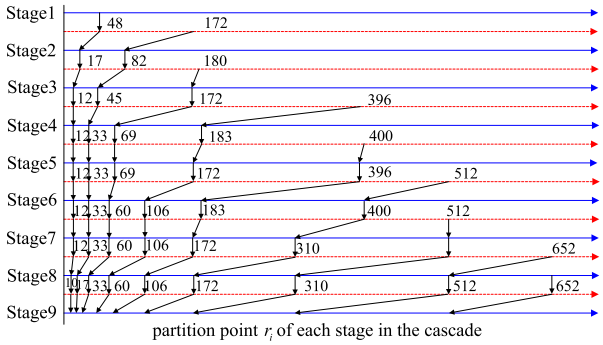


Fig. 12. Generalized decreasing phenomenon in joint-minimum-based multistage cascade.

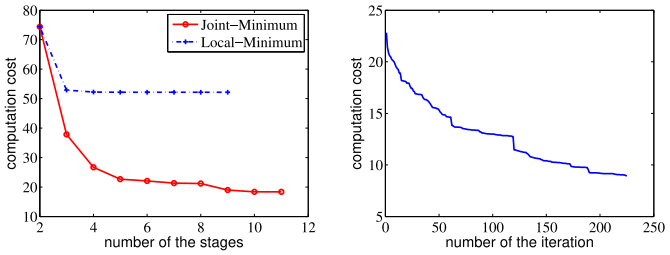
We use the local-minimum-based multistage cascade learning algorithm (see Fig. 7) to train an 9-stage cascade classifier. Table I shows how the computation cost f changes with the number of stages. The computation cost first decreases quickly and then becomes stable. This phenomenon can be explained as follows. Because the first few stages can reject the most part of the subwindows, and then only some small part of the subwindows can arrive at last few stages which takes little computation cost.

2) Joint-Minimum-Based Cascade: In the local-minimum-based multistage cascade, it seeks an optimal r_i on the condition that $(r_1, \ldots, r_{i-1})$ are known and fixed, so $(r_1, \ldots, r_{i-1}, r_i)$ cannot be jointly optimal for minimizing the computation cost $f(r_1, \ldots, r_i)$ where not only r_i but also $r_1, \ldots, r_{i-1}$ are variable. Thus, Algorithm 3 is proposed to train the joint-minimum-based multistage cascade.

Fig. 12 shows the iteration process of Algorithm 3. The number 48 on the top blue line is $r_1^{*(1)} = \arg\min f_1(r)$, which is the result of line 1 of Algorithm 3. The right number 172 on the top red line is $r_2^u = \arg\min f_2(r_2|r_1^{*(1)}) = 172$ (see line 4 of Algorithm 3). Obviously $r_2^u = 172$ is the solution of local-minimum-based optimization. The numbers 17 and 82 on the second blue line are solutions of joint-minimum-based optimization (i.e., line 13 of Algorithm 3). Generally, the right most number on each red line is the upper bound r_i^u of Algorithm 3, and the numbers on each blue line are the solutions of joint-minimum-based optimization $r_j^{*i}, j = 1, \ldots, i$.

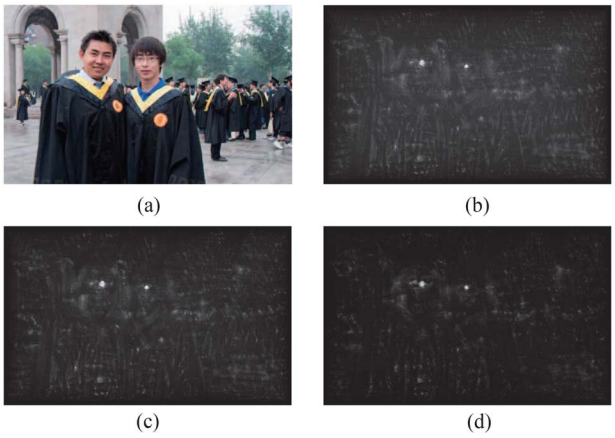
TABLE II
COMPUTATION COST OF THE CASCADE TRAINED BY ALGORITHM 3 VARIES WITH THE NUMBER $(i+1)$ OF THE STAGES

$i+1$	2	3	4	5	6	7	8	9	10
$f_{local}(i)$	74.42	52.87	31.24	26.00	22.16	21.99	21.26	21.19	18.98
$f_{joint}(i)$	74.42	37.83	26.67	22.70	22.06	21.31	21.20	18.99	18.38



(a) (b)

Fig. 13. (a) Comparison of the computation cost between local-minimum-based multistage cascade and joint-minimum-based one. (b) Computation cost decreases with the update of threshold t_i.



(a) (b) (c) (d)

Fig. 14. Computation cost shown as a function of image location. (a) Original image. (b) Recycling response image. (c) Recycling and retracting response image. (d) iCascade response image.

The generalized decreasing phenomenon can be seen from Fig. 12. For example, r_1 decreases from 48 to 17, 12; r_2 decreases from 172 to 82, 45, 33; and r_3 decreases from 180 to 172, 69, 60. Table II gives the computation cost of the cascade corresponding to Fig. 12. $i+1$ means the number of the stages in the cascade. $f_{\text{local}}(i)$ means the computation cost $f(r_i|r_1,\ldots,r_{i-1})$. In Table II, $f_{\text{local}}(1) = 74.42$ is the computation cost $f(r_1)$ with $r_1 = 48$, and $f_{\text{local}}(2) = 52.87$ is equal to $f(r_2|r_1)$ with $r_1 = 48$ and local optimization solution $r_2 = 172$. $f_{\text{joint}}(i)$ is the computation cost $f(r_1,\ldots,r_i)$ of the proposed joint-minimum algorithm, where $r_1,\ldots,r_i$ are all unknown. Note that $f_{\text{joint}}(1) = f_{\text{local}}(1)$, because the local-minimum and joint-minimum is same for two-stage cascade. However, $f_{\text{joint}}(2) = 37.83$ and $f_{\text{joint}}(3) = 26.67$ are much smaller than $f_{\text{local}}(2) = 52.87$ and $f_{\text{local}}(3) = 31.24$, respectively.

To compare the joint-minimum Algorithm 3 with the local-minimum algorithm, we visualize f_{joint} in Table II and f in Table I in Fig. 13(a). With the number of stages increasing, the computation costs both decrease. But the difference is that the computation cost of joint-minimum algorithm decreases more quickly than that of local-minimum algorithm. For example, when the numbers of stages are 4 and 9, the computation costs of the joint-minimum and local-minimum algorithms are (26.67 and 18.99) and (52.16 and 52.14), respectively. In summary, Fig. 13(a) demonstrates the advantage and importance of the proposed joint-minimum optimization algorithm.

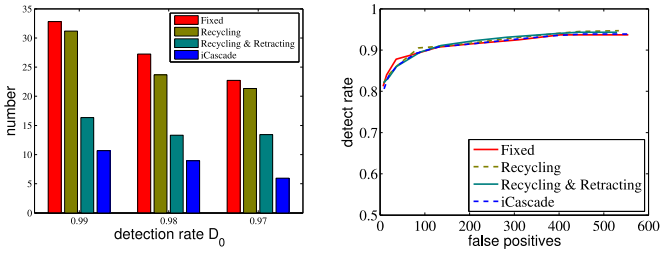
3) Threshold Learning: The thresholds $t_i, i = 1,\ldots,S$ affect the computation cost of iCascade. Algorithm 4 gives the iteration process to choose the threshold of each stage for iCascade. Note that the variation ΔD_i of detection rate is obtained by changing t_i to $t_i + \Delta t_i$. As Δt_i gradually decreases, the detection accuracy increases whereas the training time drastically grows. A set of t_i is evaluated. We find that the performance is stably good if $\Delta t_i \leq 0.02$. As a tradeoff, $\Delta t_i = 0.01$ is empirically employed. Fig. 13(b) shows how the computation cost updates in the iteration process of the first 20 stages' thresholds. It can be seen that the computation cost significantly decreases with the iteration. Fig. 13(b) shows the convergence of the proposed threshold learning algorithm. Fig. 13(b) also supports the correctness of Theorem 13.

C. Comparison on the MIT-CMU Face Dataset

In this section, we compare iCascade with Fixed Cascade [20], Recycling Cascade [9], and Recycling and Retracting Cascade [9]. Haar features are adopted.

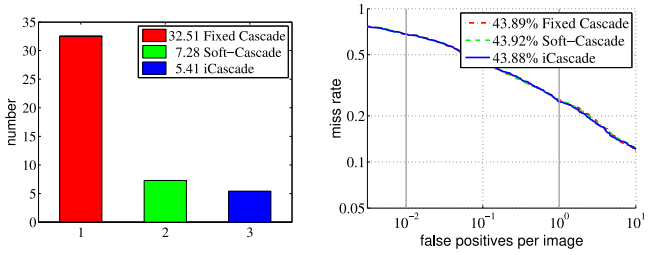
The number of average features per window is used to represent the computation cost. Fig. 14 reflects the computation cost of different algorithms (i.e., iCascade, Recycling Cascade, and Retracting and Recycling Cascade) as a function of image locations. The number of the average features used in a sliding window is accumulated to the center pixel of the sliding window. After detection, the value of each pixel is normalized to 0–255. The larger the value is, the greater the computation cost is, and the greater the probability that a face exists. It can be observed that Fig. 14(d) (i.e., iCascade) is much darker and sparser than Fig. 14(b) and (c). The darkness and sparsity imply that iCascade consumes less computation cost than the other two algorithms.

Fig. 15(a) shows the number of average features per window of different methods at different target detection rates D_0. For example, when the target detection rate is 0.97, iCascade averagely uses 5.95 features, whereas Fixed Cascade, Recycling Cascade, and Recycling and Retracting Cascade use 22.84, 20.78, and 13.32 features, respectively. Fig. 15(b) shows the receiver operating characteristic (ROC) of the different algorithms. The difference between detection performances of different methods is insignificant. We can conclude from Fig. 15 that iCascade has less computation cost without loss of detection performance.



(a) (b)

Fig. 15. (a) Comparison of the computation cost between different algorithms on the MIT-CMU face dataset. (b) ROC of different algorithms on the MIT-CMU face dataset.



(a) (b)

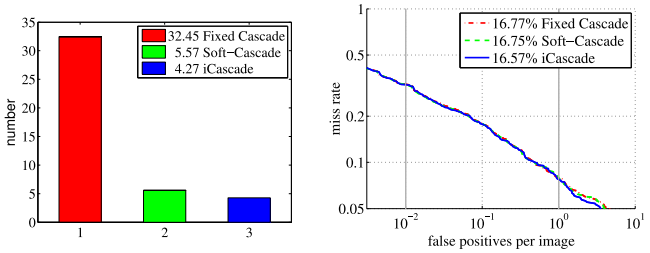
Fig. 16. Results on the Caltech dataset when ACF features [34] are employed. (a) Computation cost. (b) ROC curves.

D. Comparison on the Caltech Pedestrian Dataset

In this section, the Caltech pedestrian dataset [36] is used to compare iCascade with Fixed Cascade [20] and Soft-Cascade [22]. The state-of-the-art NNNF-L4 features [18] and the classical ACF features [34] are adopted, respectively. Caltech and Caltech 10× training datasets are used for the ACF and NNNF-L4 features, respectively. In ACF [34], 4096 level-2 decision trees are used for ACF. The decision trees are obtained after four rounds. In each round, 5000 hard negatives are added and the cumulative negatives are limited to 10 000. In [18], 4096 level-4 decision trees are used for NNNF-L4. The decision trees are obtained after five rounds. In each round, 20 000 hard negatives are added and the cumulative negatives are limited to 50 000. The number of average detection trees per window is used to represent the computation cost.

Fig. 16 shows the results when the ACF features are employed. Fig. 16(a) shows that the number of decision trees per window is used. Specifically, iCascade averagely uses 5.41 level-2 decision trees per window, Soft-Cascade uses 7.28 level-2 decision trees, and Fixed Cascade uses 32.51 decision trees. Thus, the number of level-2 decision trees per window used by iCascade is 1.87 smaller than that used by Soft-Cascade. As two features are calculated per level-2 decision tree, the number of features per window used by iCascade is 3.74 smaller than that used by Soft-Cascade. Fig. 16(b) shows that the log-average miss rates of iCascade and Soft-Cascade are 43.88% and 43.92%, respectively. It means that iCascade has less computation cost than Soft-Cascade when they have almost the same detection performance.

Fig. 17 shows the results when the NNNF-L4 features are employed. It is observed from Fig. 17(a) shows that iCascade requires the smaller number of features than Soft-Cascade.



(a) (b)

Fig. 17. Results on the Caltech dataset when NNNF-L4 features [18] are employed. (a) Computation cost. (b) ROC curves.

Specifically, iCascade and Soft-Cascade require 4.27 and 5.57 level-4 decision trees per window, respectively. Thus, the number of level-4 decision trees per window used by iCascade is 1.30 smaller than that used by Soft-Cascade. In other word, the number of features per window used by iCascade is 5.20 smaller than that used by iCascade. Fig. 17(b) shows that iCascade and Soft-Cascade have the similar performance. Thus, iCascade has less computation cost than Soft-Cascade without performance loss.

VII. Conclusion

In this paper, we have proposed to design a two-stage cascade structure by partitioning a strong classifier into left and right parts. Moreover, we have proposed to design a multistage cascade structure by iteratively partitioning the right parts. Solid theories have been provided to guarantee the existence and uniqueness of the optimal partition points with the goal of minimizing computation cost of the designed cascade classifier. Decreasing phenomenon has been discovered and theoretically justified for efficiently searching the optimal solutions. In addition, we have presented an effective algorithm for learning the optimal threshold of each stage classifier. In the future, we plan to develop more efficient cascade learning algorithm by flexibly and optimally changing the ordering of the weak classifier obtained by the AdaBoost algorithm.

References

[1] Y. Pang, H. Zhu, X. Li, and X. Li, "Classifying discriminative features for blur detection," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2015.2472478.

[2] J. Marín, D. Vázquez, A. M. López, J. Amores, and L. I. Kuncheva "Occlusion handling via random subspace classifiers for human detection," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 342–354, 2014.

[3] Y. Pang, K. Zhang, Y. Yuan, and K. Wang, "Distributed object detection with linear SVMs," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2122–2133, Nov. 2014.

[4] M. A. A. Aziz, J. Niu, X. Zhao, and X. Li, "Efficient and robust learning for sustainable and reacquisition-enabled hand tracking," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 945–958, Apr. 2016.

[5] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "A scalable stagewise approach to large-margin multiclass loss-based boosting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 1002–1013, May 2014.

[6] S. Paisitkriangkrai, C. Shen, Q. Shi, and A. van den Hengel, "RandomBoost: Simplified multiclass boosting through randomization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 4, pp. 764–779, Apr. 2014.

[7] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1359–1371, Jul. 2014.

[8] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 613–627.
[9] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Regh, "On the design of cascades of boosted ensembles for face detection," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 65–86, 2008.
[10] G. Gualdi, A. Prati, and R. Cucchiara, "Multistage particle windows for fast and accurate object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1589–1604, Aug. 2012.
[11] B. Jun, I. Choi, and D. Kim, "Local transform features and hybridization for accurate face and human detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1423–1436, Jun. 2013.
[12] C. Zhang and P. Viola, "Multi-instance pruning for learning efficient cascade detectors," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 1681–1688.
[13] Y. Pang, J. Cao, and X. Li, "Learning sampling distributions for efficient object detection," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2015.2508603.
[14] R. Benenson, M. Mathias, R. Timofte, and L. V. Gool, "Pedestrian detection at 100 frames per second," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, 2012, pp. 2903–2910.
[15] Y. Pang, H. Yan, Y. Yuan, and K. Wang, "Robust CoHOG feature extraction in human centered image/video management system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 458–468, Apr. 2012.
[16] X. Wang, T. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handing," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 32–39.
[17] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, 2006, pp. 1491–1498.
[18] J. Cao, Y. Pang, and X. Li, "Pedestrian detection inspired by appearance constancy and shape symmetry," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2016.
[19] M. Saberian and N. Vasconcelos, "Boosting classifier cascades," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 2047–2055.
[20] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[21] R. Xiao, L. Zhu, and H. Zhang, "Boosting chain learning for object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 709–715.
[22] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005, pp. 236–243.
[23] M. Pham, V. Hoang, and T. Cham, "Detection with multi-exit asymmetric boosting," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, 2008.
[24] S. Z. Li and Z. Zhang, "Floatboost learning and statistical face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1112–1123, Sep. 2004.
[25] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 369–382, Mar. 2008.
[26] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Fast and robust object detection using asymmetric totally corrective boosting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 33–46, Jan. 2012.
[27] C. Shen, P. Wang, S. Paisitkriangkrai, and A. van den Hengel, "Training effective node classifiers for cascade classification," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 326–347, 2013.
[28] H. Luo, "Optimization design of cascaded classifiers," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005, pp. 480–485.
[29] J. Sochman and J. Matas, "WaldBoost—learning for time constrained sequential detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005, pp. 150–156.
[30] S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Towards optimal training of cascaded detectors," in *Proc. Eur. Conf. Comput. Vis.*, Graz, Austria, 2006, pp. 325–337.
[31] X. Chen and A. Yuille, "A time-efficient cascade for real-time object detection: With applications for the visually impaired," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005, p. 28.
[32] M. Chen, Z. Xu, K. Weinberger, O. Chapelle, and D. Kedem, "Classifier cascade for minimizing feature evaluation cost minmin," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 218–226.
[33] Z. Xu, M. Kusner, K. Weinberger, and M. Chen, "Cost-sensitive tree of classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 133–141.
[34] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fastest feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[35] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 22–38, Jan. 1998.
[36] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.
[37] P. Dollár, Z. Tu, P. Perona, and S. Belongie. "Integral channel features," in *Proc. Brit. Mach. Vis. Conf.*, 2009, pp. 1–11.

Yanwei Pang (M'07–SM'09) received the Ph.D. degree in electronic engineering from the University of Science and Technology of China, Hefei, China, in 2004.

He is currently a Professor with Tianjin University, Tianjin, China. His current research interests include object detection and image processing. He has published over 100 scientific papers including 24 IEEE TRANSACTION papers in the above areas.

Jiale Cao received the B.S. degree in electronic engineering from Tianjin University, Tianjin, China, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include object detection and image analysis.

Xuelong Li (M'02–SM'07–F'12) is a Full Professor with the Center for Optical Imagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.