# Learning Near-Optimal Cost-Sensitive Decision Policy for Object Detection

Tianfu Wu and Song-Chun Zhu

**Abstract**—Many popular object detectors, such as AdaBoost, SVM and deformable part-based models (DPM), compute additive scoring functions at a large number of windows in an image pyramid, thus computational efficiency is an important consideration in real time applications besides accuracy. In this paper, a decision policy refers to a sequence of two-sided thresholds to execute early reject and early accept based on the cumulative scores at each step. We formulate an empirical risk function as the weighted sum of the cost of computation and the loss of false alarm and missing detection. Then a policy is said to be cost-sensitive and optimal if it minimizes the risk function. While the risk function is complex due to high-order correlations among the two-sided thresholds, we find that its upper bound can be optimized by dynamic programming efficiently. We show that the upper bound is very tight empirically and thus the resulting policy is said to be near-optimal. In experiments, we show that the decision policy outperforms state-of-the-art cascade methods significantly, in several popular detection tasks and benchmarks, in terms of computational efficiency with similar accuracy of detection.

**Index Terms**—Decision Policy, Cost-Sensitive Computing, Risk Minimization, Dynamic Programming, Object Detection

## 1 Introduction

### 1.1 Motivation and Objective

The most popular object detectors in recent years, such as the AdaBoost classifier [1] for faces, the SVM [2], [3] for pedestrians, the deformable part-based models (DPM) [4]–[6] for PASCAL VOC, all compute additive scoring functions, i.e., the prediction/matching score is the sum of a set of basic terms, such as, the weak classifiers in AdaBoost, the feature blocks in linear SVM, and the root/part filters in DPM. As these detectors are evaluated at a vast number of windows in an image pyramid, computational efficiency becomes important in real time applications besides accuracy. This problem has been extensively studied in vision, machine learning, and statistics using a range of methods, including cascades [1], [7], [8], marginal space learning [9], branch-and-bound [10], [11], cost-sensitive learning [12], [13] and inference [14], and sequential probabilistic ratio test [15].

In this paper, we present a new framework for computing *cost-sensitive* decision policies for rapid object detection. Our decision policy consists of a sequence of two-sided thresholds to execute three possible choices: early reject, early accept, or continue to the next stage – based on the current cumulative scores at each stage. Fig. 1 shows two examples of decision policies for the human face AdaBoost classifier which are optimized for different parameter settings: the left one is more cautious about early decisions and thus has higher computing costs on average, while the right one is more aggressive in early reject/accept.

- *T.F. Wu is with the Department of Statistics, University of California, Los Angeles. E-mail: tfwu@stat.ucla.edu*
- *S.C. Zhu is with the Department of Statistics and Computer Science, University of California, Los Angeles. E-mail: sczhu@stat.ucla.edu*

The policy is said to be "cost-sensitive" because it optimizes an empirical global risk function which is the weighted sum of two aspects:

- *The loss of misclassification*, weighted by the loss $C_{\text{FN}}$ for each false negative and loss $C_{\text{FP}}$ for each false positive respectively.
- *The cost of computation*, i.e., the expected run-time for computing the scoring function sequentially, which is weighted by a parameter $\lambda$.

Therefore, the parameters $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$ decide the trade-off between accuracy and computing cost.

We assume that the number of features and the scoring functions are all decided by the offline training stage by Adaboost, SVM, and DPM learning, and our policy does not change these settings. Then given parameters $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$, our algorithm produces a policy that optimizes the tight upper-bound of the risk function by a dynamic programming (DP) algorithm.

Under the same framework, a decision policy can also be computed by specifying the desired false positive rate (FPR denoted by $\alpha$) and false negative rate (FNR denoted by $\beta$), or by specifying a total computing budget.

Furthermore, since the DP algorithm is very fast, our method can update the policy on-the-fly when $\Theta$ changes due to contextual information. Our motivation is to consider object detectors as modules in bigger vision tasks, such as fine-grained hierarchical object recognition, image parsing and scene understanding. In such tasks, the following two aspects will change on-the-fly, which are considered fixed in existing methods for training the cascades or other rapid detectors.

i) *The expected positive and negative populations* will change during inference. For example, suppose in a bottom-up/top-down parsing task [16], the upper level algorithm has located a person with high probability and calls a face

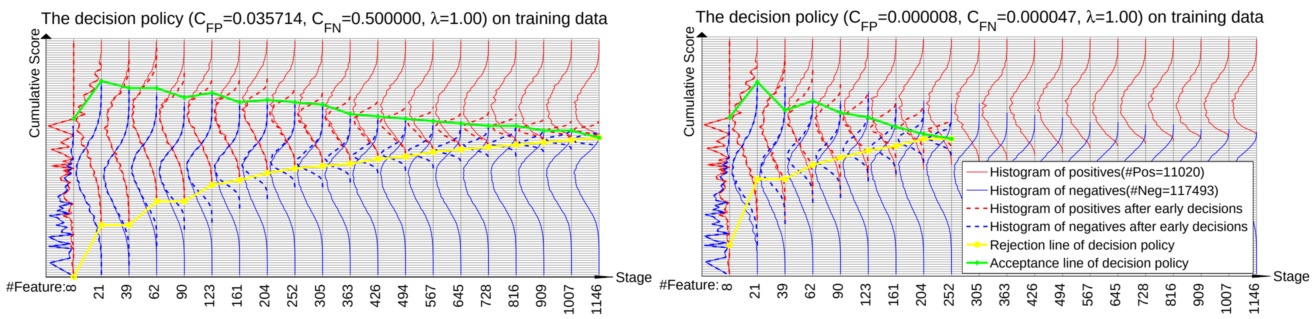

Fig. 1. Illustration of two decision policies learned for a human face AdaBoost classifier (The legend is only shown in the right plot for clarity). The horizontal axis is the $1,146$ boosted weak classifiers divided in $20$ stages, and the vertical axis is the cumulative scores. At each stage the solid histograms show the original positive (red) and negative (green) populations in training dataset, and the dotted histograms show the changed populations after early decisions by the policy.

detector to slide the windows within the bounding box of the person. The detector should expect a much higher probability of positives than in the offline training and thus adopt a more aggressive early accept.

ii) *The parameters* $\Theta$ will change in goal-guided searches. When a vision system is deployed for goal-guided search based on user's interest or queries, the loss of a false negative or false positive changes over time when attention shifts. So the decision policy should also reflect the priorities: allocating more computing steps for high loss objects, such as the policy on the left of Fig. 1, and less computing steps for low priority objects, such as the policy on the right of Fig. 1.

In both cases, a fast algorithm is entailed for automatically adapting the policy to different contexts and goals. This distinguishes our work from many existing methods in the literature [7], [8], [12], [15], [17]–[21].

## 1.2 Method Overview

Given an object detector (AdaBoost, SVM or DPM) with $T$ basic additive terms trained offline from a dataset, and the parameters $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$ or alternatively $\Theta = (\alpha, \beta, \lambda)$, our decision policy is computed in two phases.

*Phase 1: Offline statistics recording*. We divide the $T$ basic terms into $N$ stages. Unlike the traditional cascade methods that design each stage to reach the given FPR and FNR, the configuration of our stages are designed based on the statistical power/computing cost ratio [22] in a flexible way. We quantize the cumulative scores into $K$ bins, and thus obtain a $K \times N$ matrix shown in Fig. 1. For each training example, positive or negative, we record its cumulative scores of the $N$ stages, and these $N$ scores can be visualized as a trajectory. Fig. 3 plots some examples. These trajectories represent the empirical joint statistics for all samples, from which we can project to the following empirical marginal probabilities.

i) For each entry in the $K \times N$ matrix, we record the three sub-populations (i.e. the empirical probabilities) that reach this entry, above the entry (with higher cumulative scores), and below the entry (with lower cumulative scores). Based on these statistics, we can plot the histograms of positive and negative populations over the $N$ stages in Fig. 1.

ii) For every two entries $a$ and $b$ in consecutive stages, we record the various transition probabilities from $a$ to $b$, from above $a$ to above $b$, from below $a$ to below $b$ and so on. We will elaborate on these statistics in Sec.4.

We observe that the trajectories of the samples (see Fig. 3) are relatively smooth, and their cumulative scores transit only among adjacent bins. From these empirical statistics (i.e. populations of samples), we can calculate the false positive and false negative rates for different thresholds at each stage.

*Phase 2: On-line policy computation*. Given parameters in $\Theta$, we compute the two-sided thresholds for all stages jointly (i.e., a whole policy) by minimizing the empirical global risk. To evaluate each possible policy, we calculate the expected losses of early reject/accept and the computing costs based on the recorded statistics.

One complication is the double counting of risk that a training example in the data has exited early and is counted again for the loss and cost in a later stage, which means that the assignments of thresholds in all previous stages affect the current stage. From the recorded transition probabilities, we can eliminate the double counting in any two consecutive stages. But eliminating the double counting effects in higher order terms (i.e., over multiple stages) will require the recording of high order transition probabilities. We choose to ignore these high-order terms, and thus minimize a simplified risk function. It is an upper bound of the original risk function, because it may count the loss and cost of some examples more than once. Thanks to the smoothness of trajectories, we find that such high-order terms are negligible and thus the upper bound is very tight in all of our experiments. The simplified risk function is the summation of the single terms and pairwise terms, and thus can be solved efficiently by dynamic programming (DP).

With the DP tables pre-computed in phase 1, in phase 2 we can compute a family of decision policies easily for different parameter settings $\Theta$ and for different ratios of positive and negative populations, and thus different policies can be used for adaptation during fine-grained hierarchical recognition and image parsing tasks mentioned in the previous subsection.

## 1.3 Related Work and Comparisons

In the vision literature, there are three main streams of methods for rapid object detection.

*i) The cascade methods* [1], [7], [8], [20], [23]–[27] utilize early reject but no early accept, except for the WaldBoost [15] and active classification based on value of classifier [19], and [18]. In boosting cascade [1], the rejection thresholds of each stage are learned independently, and the training process needs time-consuming trial-and-error tuning to decide the number of stages, the target detection rate and false positive rates at each stage. Recently, more principled approaches are proposed on better or optimal design of AdaBoost cascade [7], [8], [20], [26]. WaldBoost [15] assumes strong independence on the output of the weak learners to guarantee optimality, and this assumption is often violated in practice. Active classification based on value of classifier [19] needs to maintain the values of all the remaining classifiers at each step, which incurs a large computing cost and is addressed by adopting some heuristics and tuning. Soft cascade [7] needs a sufficiently large validation set to calibrate the rejection thresholds. In [27], the cascade is learned after a DPM is trained. The rejection threshold of each stage (adding a part) is selected so that the probability of a misclassification is bounded from above by a predefined small positive value. In [28], the speed of DPM cascade is further boosted by exploiting some faster implementations for different components in inference. Unlike our decision policy, the cascade thresholds are fixed after the offline training stage, and not optimized for different cost-sensitive parameters $\Theta$.

*ii) The coarse-to-fine methods* [22], [29]–[31] require a coarse-to-fine partition of the pose space of an object class in modeling (i.e., learning a sequence of tests in the space of relative positions, scales and orientations). The pose space is often partitioned manually and then all the tests are organized in a hierarchy, like a decision tree. The tests are selected in decreasing order based on their ratios of the statistical power over the computing cost. In contrast, our proposed method is generic and can be directly applied to speed up any object detector with additive scoring function.

*iii) The branch-and-bound and $A^*$ methods.* The branch and bound methods [10], [11] bound the scoring functions from the upper and lower for any given sub-window and rank these sub-windows. So far, they only adapt well for detectors trained using bag-of-word features in [10] or is used to search the state space of a DPM after all filter responses are computed in [11] (thus the overall speed-up on DPM is not significant). The $A^*$ methods [32], [33] compute the lower bound for the to-go score which is usually calculated based on coarse model derived from the full model (if possible).

Besides in computer vision, there are three other streams of related work.

*iv) Cost-sensitive learning* in machine learning [12], [13], [34]. The key idea is to extend the original loss function, i.e. the exponential loss in AdaBoost and the hinge loss in SVM, to incorporate the losses of false negative and false positive $(C_{\text{FN}}, C_{\text{FP}})$ explicitly, and then derive the optimal Adaboost and SVM algorithms as the minimizer of the associated empirical loss. The algorithms are optimized in the offline setting. Since it is more natural to specify the target FNR and FPR, rather than the loss $(C_{\text{FN}}, C_{\text{FP}})$, the offline learning methods need to use cross-validation to search the $(C_{\text{FN}}, C_{\text{FP}})$ to match the target FNR and FPR. This is very time-consuming since they have to re-learn all the basic additive terms at each round. Joint learning of the scoring function and the decision policy is out of the scope of this paper. We fix the scoring function computed offline and then compute/update the near-optimal decision policy on-the-fly because we are ultimately interested in the situations where the parameters in $\Theta$ change and the positive and negative populations change. In [14], cost sensitive inference is addressed in multiscale activity recognition.

*v) Marginal space learning* [9] combines a set of classifiers learned in different marginal spaces of object parameters and the full classifier learned in the joint space. The former aims to prune uninteresting locations of the object parameter space, and the latter is used to detect the objects in the pruned space. The efficiency depends on whether it can learn good classifiers in marginal spaces which are discriminative enough to prune the space and computationally cheap to save the computing cost.

*vi) Sequential probability ratio tests (SPRT)* in statistics. SPRT can be dated back to Wald in the 1940s [35], and is often used for point hypotheses with different variations, such as the truncated SPRT in [36]. Most recently, the SPRT procedure has been adopted in vision by Pele and Werman [37] for fast pattern matching. The task was to crop an image patch in one frame and find it in sequential frames in a video using Hamming distance. They proposed a Bayesian sequential hypothesis testing method which makes early decision (accept or reject) by exploiting a generic prior distribution of the Hamming distance between general patches in natural images. Our method is related to this method [37], but with two main differences: (i) the scoring functions studied in this paper are more general than the Hamming distance function (which is a step-wise increasing function of the number of samples tested); and (ii) instead of introducing a generic prior distribution which is not straight forward to compute for the scoring functions, we use the histograms of cumulative scores on positive and negative training datasets to learn the decision policies. In [38], a method of learning the optimal policy offline for DPMs is proposed. There are also some recent work on so-called anytime algorithms [39].

Our preliminary work has been published in [40], and is extended here by: (i) elaborating details substantially in deriving the formulation and algorithm; (ii) adding more analyses on different aspects on our method; (iii) evaluating our method on PASCAL VOC2007 20-category object detection benchmark [41] with better computational efficiency and similar accuracy compared to the DPM cascade method [27].

### 1.4 Contributions and Paper Organization

This paper makes three main contributions to fast object detection.

i) It presents a new framework for computing decision policies by minimizing a global risk function that explicitly accounts for the loss of misclassification and the computational cost. This framework is generally applicable to object detectors with additive scoring functions without disturbing or iterating with the offline training.

ii) It derives an upper bound of this risk function and shows that the upper bound is tight empirically. Then it presents a DP algorithm to minimize the upper-bound efficiently. This allows us to produce the decision policy to adapt to different parameters $\Theta$ and varying ratios of the positive and negative populations. This is a desired property for adapting the object detection modules in the context of bigger tasks, such as parsing and image understanding. None of the existing cascade methods can provide such flexibility and efficiency for adaptation on-the-fly.

iii) In extensive experiments on public datasets, such as the CMU+MIT face detection benchmark [42], the INRIA person detection dataset [3], and the PASCAL VOC 2007 20-class object dataset [41], it shows that the learned decision policies for AdaBoost, SVM and DPM outperform the state-of-the-art cascade methods significantly in speed with similar accuracy.

**Paper Organization**. The remainder of this paper is organized as follows. Sec. 2 introduces the cost-sensitive object detection. Sec. 3 defines the decision policy and presents the formulation of learning a cost-sensitive decision policy. Sec. 4 presents details to compute the empirical risk. Sec. 5 derives an upper bound for the original risk function, and then proposes a DP algorithm. Sec. 6 evaluates our method in a series of experiments and compares with the state-of-the-art cascade methods. Finally, Sec. 7 concludes this paper with discussions of future directions.

## 2 Background

In this section, we introduce cost-sensitive object detection to set up the background.

### 2.1 Object Detection

We assume a sliding-window method for object detection. Thus a detector is treated as a binary classifier which is a function $h : \mathcal{X} \to \{-1, 1\}$, mapping a $d$-dimensional feature vector extracted from each window, $x \in \mathcal{X} \subset R^d$, into a class label, $y \in \{-1, 1\}$. This mapping is often implemented as

$$h(x) = \text{sign}[f(x)], \tag{1}$$

where $f : \mathcal{X} \to R$ is a real-valued scoring function. The pairs $(x, y)$'s are samples from a underlying fixed, but unknown, probability distribution $P_{X,Y}(x, y)$. In practice, to learn $f(x)$, a set of training examples is given, $D = D^+ \cup D^-$, which consists of a set of positive examples $D^+$ and a set of negative examples $D^-$. Let $S = |D|$ be the size of training dataset and $S^+ = |D^+|$ and $S^- = |D^-|$. To define the optimality of a scoring function $f(x)$, we need to introduce the concepts of loss, cost and risk as follows.

### 2.2 The Loss, Cost and Risk

**The Loss of Misclassification.** The scoring function with optimal training accuracy is learned by minimizing the empirical loss,

$$\mathcal{L}(f) = E_{X,Y}[L(y, f(x))] \approx \frac{1}{S} \sum_{(x,y)\in D} L(y, f(x)), \tag{2}$$

where $L(y, f(x))$ is a loss function which measures both the loss of a false positive $C_{\text{FP}}$ and the loss of a false negative $C_{\text{FN}}$,

$$L(y, f(x)) = \begin{cases} 0, & \text{if } h(x) = y \\ C_{\text{FP}}, & \text{if } y = -1 \text{ and } h(x) = 1 \\ C_{\text{FN}}, & \text{if } y = 1 \text{ and } h(x) = -1. \end{cases} \tag{3}$$

When $C_{\text{FP}} = C_{\text{FN}}$ (i.e., the widely used 0/1 loss function), the scoring function obtained by minimizing $\mathcal{L}(f)$ is $(C_{\text{FP}}, C_{\text{FN}})$-insensitive; otherwise, it is $(C_{\text{FP}}, C_{\text{FN}})$-sensitive [43].

*Two equivalent specifications in $(C_{FP}, C_{FN})$-sensitive learning*. $(C_{\text{FP}}, C_{\text{FN}})$ are usually specified from domain knowledge. In some cases, such as the fraud detection problem, they are given in terms of prior experience. In object detection, it is often more natural to specify FPR, $\alpha$ and FNR, $\beta$, rather than the two penalties directly. But, the two ways of specifications are equivalent in the sense that:

i) Given $(C_{\text{FP}}, C_{\text{FN}})$, the optimal scoring function, in terms of accuracy, is given by the Bayesian decision rule,

$$f^*(x) = \arg\min_f \mathcal{L}(f) = \log \frac{P_{Y|X}(1|x)C_{\text{FN}}}{P_{Y|X}(-1|x)C_{\text{FP}}} \tag{4}$$

ii) For the alternative specification, the goal is to minimize FPR, $\alpha$, subject to a FNR, $\beta$. According to the Neyman-Pearson Lemma [44], for any $\beta$, the optimal accuracy scoring function is still Eqn. (4) with the rejection region being:

$$\mathcal{H} = \{x; \frac{P_{Y|X}(1|x)}{P_{Y|X}(-1|x)} \leq \frac{C_{\text{FP}}}{C_{\text{FN}}}\}. \tag{5}$$

So, the two penalties $(C_{\text{FP}}, C_{\text{FN}})$ need to be searched such that a given $\beta$ is met, i.e., $\beta = \int_{\mathcal{H}} P_{X|Y}(x|y = 1)dx$. The search is often done by cross-validation.

As mentioned before, in this paper, we consider more general situations in object detection where $(\alpha, \beta)$ can be dynamically changed. So it will not be practical to re-train the scoring function accordingly (especially with expensive cross-validation required) or to train the scoring functions for all possible specifications offline.

**The Cost of Computation.** The expected and empirical computing cost of a scoring function is defined as,

$$\mathcal{C}(f) = E_{X,Y}[C(y, f(x))] \approx \frac{1}{S} \sum_{(x,y)\in D} C(y, f(x)), \tag{6}$$

where $C(y, f(x))$ is the computing cost for evaluating a random sample $(x, y)$.

**The Global Risk.** The global risk is the sum of the expected loss plus the expected cost,

$$\mathcal{R}(f) = \mathcal{L}(f) + \lambda \cdot \mathcal{C}(f), \tag{7}$$

where $\lambda$ is the regularization parameter. The default value is $\lambda = 1$. Eqn. (7) can be also interpreted as a constrained optimization problem, i.e., either minimizing the empirical loss subject to computational constraints or minimizing the computational cost subject to reachable bounds of accuracy. We will adopt this interpretation when formulating the learning of policy.

### 2.3 The Additive Scoring Functions

Popular object detectors, such as AdaBoost, SVM and DPM, all use an offline trained additive scoring function with $T$ basic terms,

$$f(x) = \sum_{t=1}^{T} g_t(x), \tag{8}$$

where $g_t(x)$ can be a weighted weak classifier in AdaBoost [1], a divided block (a weighted support vector) in linear (non-linear) SVM [3], [31], or a root/part filter in DPM [4].

**The order of basic additive terms.** We assume that the computing order of $g_t(x)$'s is sorted in offline training, which is based on the ratio between statistical power and computing cost, similar to [17], [22]. The computing cost is fixed for $g_t(\cdot)$ after offline training. The statistical power of $g_t(\cdot)$ is defined by 1-FPR subject to FNR = 0 which can be calculated on the training dataset. For notational simplicity, we will denote by $g_t(\cdot)$ the sorted basic additive terms.

## 3 Formulation

### 3.1 The Definition of a Decision Policy

For a given scoring function in Eqn.(8), we divide the $T$ basic additive terms into $N$ stages, and denote the summed scoring function at each stage by $G_i(x)$, $1 \leq i \leq N$. For an input sample $x$, the cumulative score at the $i$-th stage is defined by,

$$f^i(x) = \sum_{j=1}^{i} G_j(x). \tag{9}$$

The number of stages $N$ and the number of basic additive terms in each stage are quite flexible in our experiments. For example, for the AdaBoost face detection task, we set $N = 20$ stages for comparing with the cascade methods using the same number of stages, and put more basic additive terms in later stages. In DPM, each root or part filter corresponds to a stage, similar to [27].

**Definition 1 (The decision policy):** An $N$-stage decision policy is denoted by,

$$\Pi_N = (\tau_1, \cdots, \tau_i, \cdots, \tau_N), \tag{10}$$

where each $\tau_i$ is a pair of thresholds (two-sided),

$$\tau_i = (\tau_i^-, \tau_i^+),\ i = 1, \cdots, N. \tag{11}$$

$\tau_i^- < \tau_i^+$ for $i = 1, \cdots, n-1$ and $\tau_i^- = \tau_i^+$ for $i = n, \cdots, N$, and $n$ is the number of stages actually used by a decision policy (see the two examples in Fig.1, $n = 20$ and $n = 9$ respectively). $n$ is computed automatically.

$\Pi_N$ makes three possible actions at stage $i$ in terms of cumulative score $f^i(x)$,

- *Reject* $x$, if $f^i(x) < \tau_i^-$,
- *Accept* $x$, if $f^i(x) \geq \tau_i^+$, and
- *Continue to compute* $G_{i+1}(x)$, otherwise.

We call $(\tau_1^-, \cdots, \tau_n^-)$ the **rejection line** (green) and $(\tau_1^+, \cdots, \tau_n^+)$ the **acceptance line** (yellow) which are shown in Fig. 1. At each stage, $\tau_i^-$ can take the special value $\tau_i^- = -\infty$ meaning that no rejections should be decided, and similarly $\tau_i^+ = +\infty$ indicating that no acceptances could be made. We will write a decision policy $\Pi_N$ in the recursive form, $\Pi_i = (\Pi_{i-1}, \tau_i)$.

### 3.2 The risk function of a decision policy

Different decision policies incur different risks. Following Enq. (7), we denote by $\mathcal{R}(\Pi_N; \Theta)$ the risk function of $\Pi_N$,

$$\mathcal{R}(\Pi_N; \Theta) = \mathcal{L}(\Pi_N; C_{\text{FP}}, C_{\text{FN}}) + \lambda \cdot \mathcal{C}(\Pi_N), \tag{12}$$

where $\mathcal{L}(\cdot)$ is the expected loss and $\mathcal{C}(\cdot)$ the expected computational cost. We will compute the empirical risk and still use $\mathcal{R}(\cdot)$, $\mathcal{L}(\cdot)$ and $\mathcal{C}(\cdot)$ to denote the empirical counterparts for simplicity. Note that Enq. (7) can be viewed as a special case of Enq. (12) as it corresponds to a policy which has no early accept and early reject.

Recall that $D = D^+ \cup D^-$ denotes the training dataset with positives in $D^+$ and negatives in $D^-$, and $S^+ = |D^+|, S^- = |D^-|$ and $S = |D| = S^+ + S^-$ their sizes.

**The expected loss** $\mathcal{L}(\Pi_N; C_{\text{FP}}, C_{\text{FN}})$ is defined by,

$$\begin{aligned} \mathcal{L}(\Pi_N; C_{\text{FP}}, C_{\text{FN}}) &= E_{X,Y}[\mathbf{1}_{(y=1)} \cdot p(\text{FN}; \Pi_N) \cdot C_{\text{FN}} \\ &\qquad + \mathbf{1}_{(y=-1)} \cdot p(\text{FP}; \Pi_N) \cdot C_{\text{FP}}] \\ &\approx \frac{S^+}{S} \cdot \hat{p}(\text{FN}; \Pi_N) \cdot C_{\text{FN}} + \frac{S^-}{S} \cdot \hat{p}(\text{FP}; \Pi_N) \cdot C_{\text{FP}}, \end{aligned} \tag{13}$$

where $\mathbf{1}_{(\cdot)}$ is the indicator function, $p(\text{FN}; \Pi_N)$ and $p(\text{FP}; \Pi_N)$ are the probabilities of a FN and a FP occurring according to $\Pi_N$ respectively, and the second expression in Eqn. (13) is the empirical loss on the training dataset $D$.

$\hat{p}(\cdot; \Pi_N)$ are the corresponding empirical probabilities,

$$\hat{p}(\text{FN}; \Pi_N) = \hat{p}(\Pi_N \text{ rejects } x | x \in D^+) = \frac{\#\text{FN by } \Pi_N}{S^+},$$

$$\hat{p}(\text{FP}; \Pi_N) = \hat{p}(\Pi_N \text{ accepts } x | x \in D^-) = \frac{\#\text{FP by } \Pi_N}{S^-}.$$

These two empirical probabilities are calculated by exploiting the statistics of cumulative scores using the training or validation dataset in Sec. 4.

**The expected computational cost** $\mathcal{C}(\Pi_N)$. Following Eqn. (9), we denote by $c(G_i)$ the computational cost of an individual stage $i$ (which is fixed after $f(x)$ is trained offline). We denote the normalized cumulative computational cost of the first $i$ stages by,

$$\mathbf{C}_i = \frac{1}{c(f)} \sum_{j=1}^{i} c(G_j). \tag{14}$$

where $c(f)$ is the total cost of the whole scoring function, and we have:

$$0 < \mathbf{C}_i < 1, \text{ for } i < N, \text{ and } \mathbf{C}_N = 1. \tag{15}$$

For a given sample $x$, let $n(x)$ be the number of stages tested before a decision is made by $\Pi_N$ (i.e., its label is assigned), and its computational cost is $\mathbf{C}_{n(x)}$. So,

$$\mathcal{C}(\Pi_N) = E_X[\mathbf{C}_{n(x)}] \approx \frac{1}{S}\sum_{x\in D}\mathbf{C}_{n(x)} = \sum_{i=1}^{N}\hat{p}(i;\Pi_N)\cdot\mathbf{C}_i, \tag{16}$$

where $\hat{p}(i;\Pi_N)$ is the empirical probability of making an early decision at stage $i$ (to be calculated in Sec. 4),

$$\hat{p}(i;\Pi_N) = \hat{p}(\Pi_N \text{ classifies } x \text{ at stage } i | x\in D) = \frac{\#S_i}{S},$$

where $\#S_i$ is the number of examples in $D$ which are classified at stage $i$ by $\Pi_N$.

By substituting the three empirical probabilities into the risk function in Eqn. (12), we obtain the empirical risk,

$$\mathcal{R}(\Pi_N;\Theta) = \frac{1}{S}[\#\text{FN}\cdot C_{\text{FN}} + \#\text{FP}\cdot C_{\text{FP}} + \lambda\sum_{i=1}^{N}\mathbf{C}_i\cdot\#S_i] = \frac{1}{S}\sum_{i=1}^{N}[\#\text{FN}_i\cdot C_{\text{FN}} + \#\text{FP}_i\cdot C_{\text{FP}} + \lambda\mathbf{C}_i\cdot\#S_i], \tag{17}$$

where $\#\text{FN}_i$ and $\#\text{FP}_i$ are the number of FNs and FPs at stage $i$ made by $\tau_i^-$ and $\tau_i^+$ in $\Pi_N$ respectively.

**The optimal decision policy.** Given parameters $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$, the optimal decision policy $\Pi_N^*(\Theta)$ is found by minimizing the empirical risk in Eqn. (17),

$$\Pi_N^*(\Theta) = \arg\min_{\Pi_N}\mathcal{R}(\Pi_N;\Theta). \tag{18}$$

As mentioned above, in object detection, it is often more convenient or intuitive to specify reachable bounds on the false positive error rate (FPR) $\alpha$ and false negative error rate (FNR) $\beta$ based on the ROC curve of the detector $f(x)$, rather than defining $(C_{\text{FP}}, C_{\text{FN}})$. In fact, solving Eqn. (18) is equivalent to minimizing $\mathcal{C}(\Pi_N)$ subject to some given reachable accuracy bounds $(\alpha,\beta)$. Then, the problem can be reformulated as a constrained optimization problem,

$$\Pi_N^* = \arg\min_{\Pi_N}\mathcal{C}(\Pi_N), \quad \text{subject to } \hat{p}(\text{FP};\Pi_N)\le\alpha \text{ and } \hat{p}(\text{FN};\Pi_N)\le\beta. \tag{19}$$

The two types of formulations are equivalent as the following lemma states.

**Lemma 1.** The solution $\Pi_N^*$ of Eqn. (18) is also the solution to the constrained optimization problem in Eqn. (19) with $\alpha = \hat{p}(\text{FP};\Pi_N^*)$ and $\beta = \hat{p}(\text{FN};\Pi_N^*)$.

*Proof:* See the proof in the supplementary material. □

This equivalence enables the transfer from $(\alpha,\beta)$ to $(C_{\text{FP}}, C_{\text{FN}})$, as is stated by the following proposition. Algorithm. 1 will list the steps of the search in Sec.5.3.

**Proposition 1.** Given $(\alpha,\beta)$, the corresponding $(C_{\text{FP}}, C_{\text{FN}})$ are sought by binary search in the range $[0, C_{\text{FP}}^{max}]$ and $[0, C_{\text{FN}}^{max}]$ where,

$$C_{\text{FP}}^{max} = \frac{\lambda\cdot S}{\alpha\cdot S^-} \quad\text{and}\quad C_{\text{FN}}^{max} = \frac{\lambda\cdot S}{\beta\cdot S^+}. \tag{20}$$

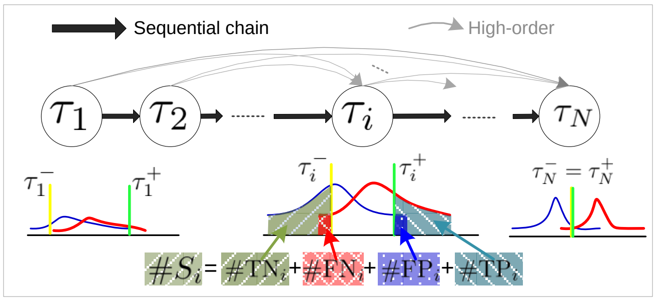

Fig. 2. Graph interpretation of minimizing the empirical risk. The two-sided thresholds at a stage $i$ generate the three numbers ($\#\text{FN}_i$, $\#\text{FP}_i$, $\#S_i$). All the thresholds are fully dependent in the sense that the positive and negative sub-populations observed at a stage $i$ are affected by the thresholds at all previous stages.

*Proof:* See the proof in the supplementary material. □

Based on proposition 1 and Eqn. (17), we can give some intuitive interpretations for what roles the three numbers in $\Theta$ play and what the optimization algorithm tries to solve. At a stage $i$, if the algorithm attempts to increase the lower threshold $\tau_i^-$, it needs to consider two factors:

i) the remaining computational costs are saved for the increased number of both true negatives (TN) and FNs (i.e., the "good" side whose actual effect is dependent on $\lambda$).
ii) the number of FN might also increase (i.e., the "bad" side which will be penalized according to $C_{\text{FN}}$)

In addition, the positive and negative sub-populations observed by stage $i$ are affected by all previous stages, which makes the decision "harder". Similar factors exist when the algorithm tries to decrease $\tau_i^+$.

The optimization procedure resolves all the factors jointly to make the loss of misclassification and the cost of computation "balanced" (i.e., minimizing the empirical global risk). For example, at an early stage $i$, especially the first few ones, if a relatively large number of TNs can be rejected (i.e., their remaining computing costs are saved) with only very few FNs incurred when $\tau_i^-$ is increased, the algorithm might take advantage of this depending on the parameters $(C_{\text{FP}}, C_{\text{FN}}, \lambda)$ (i.e., when the increased FN penalty is less than the decreased computing cost).

### 3.3 High-order connections between $\tau_i$'s in $\Pi_N$

Before presenting the method of minimizing the risk in Eqn. (18), we show that all two-sided thresholds $\tau_i$'s in $\Pi_N$ are fully dependent.

In Fig.2, we consider $\tau_i$ at stage $i$, based on Eqn. (17), we need to calculate the three quantities $(\#\text{FN}_i, \#\text{FP}_i, \#S_i)$. To do that, we first need to know the positive and negative sub-populations (illustrated by the two histograms in the center-bottom of Fig.2), which are affected by all the previous stages $j$ ($j < i$) as early thresholds $\tau_j$'s change the populations through their decisions. So, $\tau_i$ is affected by all previous $\tau_j$'s. By transition, $\tau_i$ affects the assignments of all the subsequent stages $\tau_j$ ($j > i$), and then we can see that all $\{\tau_i\}$'s are fully dependent.

This underlying fully-dependent graph structure makes the problem of minimizing the risk in Eqn. (18) very hard in general. To address the optimization issue, we proceed in two steps.

i) *Offline statistics recording*. We derive a recursive formula for counting (#FN$_i$, #FP$_i$, #$S_i$) in each stage.
ii) *Online policy computation*. By removing all the high-order connections in Fig. 2 we derive an upper bound for the empirical risk which is then minimized by a DP algorithm efficiently.

## 4 Offline Recording of Statistics

To compute the empirical risk function in Eqn.(17), we record a number of statistics in the training data $D$.

We first compute two histograms of the cumulative scores $f^i(x)$ at each stage $i$ on $D^+$ and $D^-$ respectively. At stage $i$, the set of cumulative scores is denoted by

$$A(i) = \{f^i(x); \forall x \in D\}. \tag{21}$$

We discretize $A(i)$ into $K$ bins (e.g., $K = 102$ used in all of our experiments) and let $bin(x, i)$ be the bin index of example $x$ at stage $i$. Then, we create the $K \times N$ matrix to record the trajectories of all training examples. The trajectory of a sample $x$ in the matrix is denoted by,

$$\text{Trajectory}(x): \{bin(x, 1), \cdots, bin(x, N)\}.$$

Fig.3 shows the $K \times N$ matrix and 10 trajectories of positive examples and 10 trajectories of negative examples. From the $K \times N$ matrix, we have the following statistical observation.

**Statistical observation.** *The trajectories of examples over the sequence of cumulative scores remain relatively steady*. Similar to Fig. 3, we can plot the histograms of bin index changes between two successive stages bins, and observe that most of the changes are within 2-4 bins for both positive and negative examples. This is true for different additive scoring functions studied in this paper. This observation has the following indications.

i) If $\tau_{i-1}$ classifies (accepts or rejects) $x$ based on $f^{i-1}(x)$, it is unlikely that its scores $f^i(x)$ will move inside the thresholds $[\tau_j^-, \tau_j^+)$ of later stages $j \geq i$, thus the policy will unlikely regret its decisions. Intuitively, the steadiness of the trajectories is the essential reason for the decision policy to work well.
ii) Similarly, an example $x$ decided by stage $i-1$ will have a much lower probability at stages $j > i$, and thus the double counting of loss estimation will be very rare. This makes our upper bound very tight later.
iii) As a guideline for selecting the number of stages $N$ in experiments, the number $N$ fundamentally depends on how smooth these trajectories are. This is much like the sampling theorem in Signal Processing. Higher fluctuation (or frequency) needs denser sampling.

**Notational Usages**. We compute four types of statistics from the matrix, and illustrate them in Fig.4. These statistics

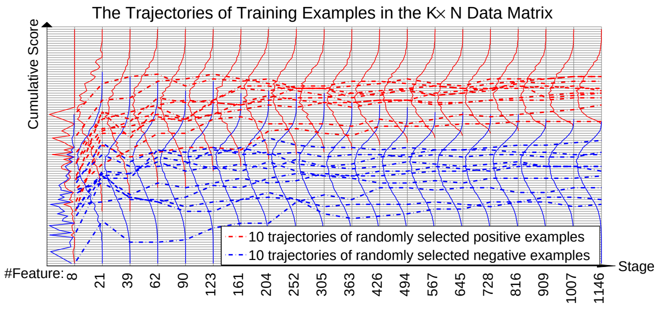

Fig. 3. 20 trajectories of positive (red dashed lines) and negative (blue) training examples in the $K \times N$ matrix. The examples are from the human face AdaBoost classifier. between successive steps in the trajectories.

are used for computing #FN$_i$, #FP$_i$, #$S_i$ in Eqn.(17). In all the following definitions, if the only difference is whether $x \in D^+, D^-$ or $D$, we then only write definitions for $D^+$ explicitly for clarity. And, we use "$S$" in notations denoting the cardinality of a corresponding set. Furthermore, we use the notation of a threshold itself (such as $\tau_i^+$ or $\tau_i^-$) in the decision policy to denote its bin index in the $K \times N$ matrix without confusion.

i) *Single-entry based statistics*. Denote by $k_i$ an entry (row $k$, column $i$) in the $K \times N$ matrix. At each stage/column $i$, the training dataset ($D = D^+ \cup D^-$) is distributed into different rows. We denote the sub-population of positive examples falling into bin $k$ at stage $i$ by,

$$D^+(k_i) = \{x : bin(x, i) = k, x \in D^+\} \tag{22}$$

and its cardinality by $S^+(k_i) = |D^+(k_i)|$ (e.g., $D^+(5_1)$ in Fig.4 (a)).

ii) *Column-based statistics*. Denote by $\underline{k_i}$ all the entries below row $k$ (exclusive) at column $i$, and correspondingly by $\overline{k_i}$ all the entries above row $k$ (inclusive),

$$D^+(\underline{k_i}) = \cup_{r=1}^{k-1} D^+(r_i) \tag{23}$$

and

$$D^+(\overline{k_i}) = \cup_{r=k}^{K} D^+(r_i) \tag{24}$$

For examples, see $D^+(\overline{7_5})$ and $D^+(\underline{6_5})$ in Fig.4 (b) and (c) (note that (c) is not $D^+(\underline{5_5})$ due to the exclusion used in definitions).

iii) *Rejection line and acceptance line based statistics.* We rewrite $\Pi_N = (\Pi_i, \tau_{i+1}, \cdots, \tau_N)$, and denote the entries below the rejection line of $\Pi_i$ (front part in $\Pi_N$) by,

$$\underline{\Pi_i} = (\underline{\tau_1^-}, \cdots, \underline{\tau_i^-}), \tag{25}$$

and the entries above the acceptance line by,

$$\overline{\Pi_i} = (\overline{\tau_1^+}, \cdots, \overline{\tau_i^+}). \tag{26}$$

For examples, $D^+(\overline{\Pi_3})$ and $D^+(\underline{\Pi_3})$ in Fig.4 (d) and (e) are the sub-populations of positives accepted and rejected by $\Pi_3$ respectively.

In the next step, we count the numbers of false positives and false negatives generated by the policy up to stage $i$, $\Pi_i$.

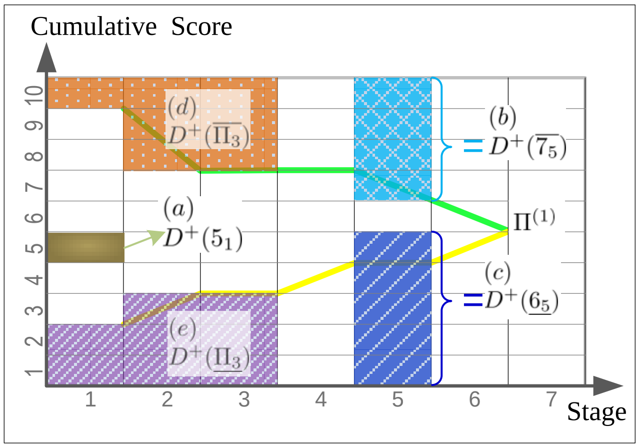

Fig. 4. Illustrating the notations of the four types of statistics projected from trajectories of all training examples (only $D^+$ is used for clarity). Note that notations like $\underline{6_5}$ and $\overline{7_5}$ are used for this specific illustration.

**Definition 2: False negatives generated by** $\Pi_i$ is defined by,

$$D^+(\underline{\Pi_i}) = \{x; \Pi_i \text{ rejects x}, x \in D^+\}. \quad (27)$$

Similarly, true positives (TP) by $\Pi_i$ is defined by $D^+(\overline{\Pi_i})$, and **the sub-population of positives classified by** $\Pi_i$ is then defined by,

$$D^+(\Pi_i) = D^+(\underline{\Pi_i}) \cup D^+(\overline{\Pi_i}). \quad (28)$$

**Definition 3: False positives generated by** $\Pi_i$ is defined by,

$$D^-(\overline{\Pi_i}) = \{x; \Pi_i \text{ accepts x}, x \in D^-\}. \quad (29)$$

When computing the number of FNs (i.e., $S^+(\underline{\Pi_i})$) and FPs (i.e., $S^-(\overline{\Pi_i})$) by different $\Pi_i$'s, instead of counting them in a brute-force way, we want to compute them efficiently in a recursive manner. So we need to compute the sample transitions between different sub-populations, for example, $D^+(\underline{\Pi_{i-1}})$ and $D^+(\underline{\tau_i})$, to count the number of double-countings.

iv) *Transition based statistics*. By considering sample transition from one sub-population (e.g., those defined based on an entry $k_i$, the bottom part $\underline{k_i}$ of a column $i$, or a partial rejection line $\underline{\Pi_i}$, as stated above) to the other, we can count the double-counting of examples between them. Specifically, we define the two as follows.

**Definition 4: False negatives double-counted at** $\tau_i$ **w.r.t.** $\Pi_{i-1}$ is the sub-population of examples in $D^+$ that has been classified by $\Pi_{i-1}$ in the first $i-1$ stages, and is rejected again by $\tau_i^-$. This set of examples will be counted more than once if we compute the empirical risk as the sum of risks caused by individual stages. We have,

$$D^+(\underline{\tau_i^-}; \Pi_{i-1}) = D^+(\underline{\tau_i^-}) \cap D^+(\Pi_{i-1}).$$

**Definition 5: False positives double-counted at** $\tau_i$ **w.r.t.** $\Pi_{i-1}$ is the sub-population of examples in $D^-$ that has been classified by $\Pi_{i-1}$ in the first $i-1$ stages, and is accepted again by $\tau_i^+$. We have,

$$D^-(\overline{\tau_i^+}; \Pi_{i-1}) = D^-(\overline{\tau_i^+}) \cap D^-(\Pi_{i-1}).$$

**Recursions**. By $\Pi_i = (\Pi_{i-1}, \tau_i)$, we can re-write $D^+(\underline{\Pi_i})$ and $D^-(\overline{\Pi_i})$ in a recursive manner as,

$$D^+(\underline{\Pi_i}) = D^+(\underline{\Pi_{i-1}}) \cup (D^+(\underline{\tau_i^-}) \setminus D^+(\underline{\tau_i^-}; \Pi_{i-1})),$$
$$D^-(\overline{\Pi_i}) = D^-(\overline{\Pi_{i-1}}) \cup (D^-(\overline{\tau_i^+}) \setminus D^-(\overline{\tau_i^+}; \Pi_{i-1})),$$

where '$\cdot \setminus \cdot$' represents the set minus operator.

So, **the number of FNs by** $\Pi_i$ is recursively defined by,

$$\begin{aligned} S^+(\underline{\Pi_i}) &= S^+(\underline{\Pi_{i-1}}) + [S^+(\underline{\tau_i^-}) - S^+(\underline{\tau_i^-}; \Pi_{i-1})] \\ &= \sum_{j=1}^{i} [S^+(\underline{\tau_j^-}) - S^+(\underline{\tau_j^-}; \Pi_{j-1})] \quad (30) \\ &\triangleq \sum_{j=1}^{i} \#\text{FN}_j, \end{aligned}$$

and **the number of FPs by** $\Pi_i$ is calculated by,

$$\begin{aligned} S^-(\overline{\Pi_i}) &= S^-(\overline{\Pi_{i-1}}) + [S^-(\overline{\tau_i^+}) - S^-(\overline{\tau_i^+}; \Pi_{i-1})] \\ &= \sum_{j=1}^{i} [S^-(\overline{\tau_j^+}) - S^-(\overline{\tau_j^+}; \Pi_{j-1})] \quad (31) \\ &\triangleq \sum_{j=1}^{i} \#\text{FP}_j. \end{aligned}$$

**The number of examples in** $D$ **which are classified at stage** $i$ **by** $\Pi_i$ is defined by,

$$\begin{aligned} S(\tau_i) &= |D(\underline{\tau_i^-}) \cup D(\overline{\tau_i^+})| \quad (32) \\ &= [S(\underline{\tau_i^-}) - S(\underline{\tau_i^-}; \Pi_{i-1})] + [S(\overline{\tau_i^+}) - S(\overline{\tau_i^+}; \Pi_{i-1})] \\ &\triangleq \#S_i. \end{aligned}$$

Now, we are ready to compute the three probabilities in the empirical global risk function in Eqn.(13).

$$\hat{p}(\text{FN}; \Pi_N) = S^+(\underline{\Pi_N})/S^+, \quad (33)$$
$$\hat{p}(\text{FP}; \Pi_N) = S^-(\overline{\Pi_N})/S^-, \quad (34)$$
$$\hat{p}(i; \Pi_N) = S(\tau_i)/S. \quad (35)$$

In summary, all these statistics are computed only once at the offline stage, and can be used to compute the decision policy on-the-fly for any given $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$.

## 5 Online Computing of Policy

In this section, we derive an upper bound for the empirical risk function and propose a DP algorithm to minimize the upper bound.

By substituting Eqn. (30), Eqn. (31) and Eqn. (32) into Eqn. (17), the empirical risk is divided into two parts: the sum of risks caused by the assignment of an individual $\tau_i$,

$$\begin{aligned} R(\tau_i; \Theta) = \frac{1}{S} \cdot [\lambda \cdot \mathbf{C}_i \cdot (S(\underline{\tau_i^-}) + S(\overline{\tau_i^+})) \\ + S^+(\underline{\tau_i^-}) \cdot C_{\text{FN}} + S^-(\overline{\tau_i^+}) \cdot C_{\text{FP}}], \quad (36) \end{aligned}$$

and the sum of risks caused by examples which are double-counted at stage $i$ w.r.t the first $i-1$ stages in $\Pi_{i-1}$,

$$\begin{aligned} \mathbb{R}(\tau_i, \Pi_{i-1}; \Theta) = \frac{1}{S} [\lambda \cdot \mathbf{C}_i \cdot (S(\underline{\tau_i^-}; \Pi_{i-1}) + S(\overline{\tau_i^+}; \Pi_{i-1})) \\ + S^+(\underline{\tau_i^-}; \Pi_{i-1}) \cdot C_{\text{FN}} + S^-(\overline{\tau_i^+}; \Pi_{i-1}) \cdot C_{\text{FP}}]. \quad (37) \end{aligned}$$

So, the empirical risk in Eqn. (17) is re-produced as,

$$\mathcal{R}(\Pi_N;\Theta)=\sum_{i=1}^{N}[R(\tau_i;\Theta)-\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)]. \tag{38}$$

## 5.1 The upper bound of the risk

To obtain an upper bound of the risk in Eqn. (38), we need a lower bound of $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$. We remove those high-order connections (gray arrows in Fig. 2), and only consider the sequential chain connections (black arrows) to obtain the lower bound of $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$, denoted by $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$.

**Proposition 2 (The upper bound of risk)**: By substituting $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$ with its lower bound $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$, we obtain an upper bound of $\mathcal{R}(\Pi_N;\Theta)$,

$$\widehat{\mathcal{R}}(\Pi_N;\Theta)=\sum_{i=1}^{N}R(\tau_i;\Theta)-\sum_{i=2}^{N}\mathbb{R}(\tau_i,\tau_{i-1};\Theta), \tag{39}$$

where $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$ is the lower bound of $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$,

$$\begin{aligned}\mathbb{R}(\tau_i,\tau_{i-1};\Theta)=\frac{1}{S}\cdot\{\lambda\cdot\mathbf{C}_i\cdot[(S(\underline{\tau_i^-};\underline{\tau_{i-1}^-})+S(\underline{\tau_i^-};\overline{\tau_{i-1}^+})\\ +S(\overline{\tau_i^+};\underline{\tau_{i-1}^-})+S(\overline{\tau_i^+};\overline{\tau_{i-1}^+}))]\\ +[S^-(\overline{\tau_i^+};\underline{\tau_{i-1}^-})+S^-(\overline{\tau_i^+};\overline{\tau_{i-1}^+})]\cdot C_{\text{FP}}\\ +[S^+(\underline{\tau_i^-};\underline{\tau_{i-1}^-})+S^+(\underline{\tau_i^-};\overline{\tau_{i-1}^+})]\cdot C_{\text{FN}}\,\}.\end{aligned} \tag{40}$$

*Proof:* The intuitive idea is that $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$ considers the risk caused by examples double-counted at $\tau_i$ w.r.t. the previous stage $i-1$ only, while $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$ is calculated by taking into account all the first $i-1$ previous stages. See the proof in the supplementary material. □

## 5.2 The DP algorithm

The upper bound in Eqn. (39) is in the same form as the energy functions widely used in the literature of discrete optimization, which includes a singleton energy term (i.e., the sum of $R(\cdot)$'s) and a pairwise energy term (i.e., the sum of $\mathbb{R}(\cdot)$'s), and can be optimized by DP efficiently (within several seconds to minutes) as follows.

Let $B_i[\tau_i]$ be the risk of the best assignment to the $i$ first stages with the constraint that the $i$-th one is $\tau_i$. We have,

$$\begin{aligned}B_1[\tau_1]&=R(\tau_1;\Theta),\\ B_i[\tau_i]&=R(\tau_i;\Theta)\\ &\quad+\min_{\tau_{i-1}}(B_{i-1}[\tau_{i-1}]-\mathbb{R}(\tau_i,\tau_{i-1};\Theta)).\end{aligned} \tag{41}$$

Then, the DP algorithm consists of two steps:

i) *The forward step* for computing all $B_i[\tau_i]$'s, and caching the optimal solution for $\tau_{i-1}$ as a function of $\tau_i$ for later back-tracing starting at $i=2$,

$$\mathbf{T}_i[\tau_i]=\arg\min_{\tau_{i-1}}(B_{i-1}[\tau_{i-1}]-\mathbb{R}(\tau_{i-1},\tau_i;\Theta)).$$

ii) *The backward step* for finding the near-optimal decision policy $\Pi_N^*=(\tau_1^*,\cdots,\tau_N^*)$, where we first take,

$$\tau_N^*=\arg\min_{\tau_N}B_N[\tau_N], \tag{42}$$

```
Input: The given FPR and FNR, i.e., $(\alpha,\beta)$, and the
       precomputed DP tables.
Output: The learned decision policy.
Initialization: Let $C_{\text{FP}}^{min}=C_{\text{FN}}^{min}=0$. $C_{\text{FP}}^{max}$ and
$C_{\text{FN}}^{max}$ are set using Eqn. (20);
· Repeated.
    i)   $C_{\text{FP}}=\frac{C_{\text{FP}}^{min}+C_{\text{FP}}^{max}}{2}$ and $C_{\text{FN}}=\frac{C_{\text{FN}}^{min}+C_{\text{FN}}^{max}}{2}$
    ii)  Solve $\Pi_N^*(C_{\text{FP}},C_{\text{FN}})=\arg\min_{\Pi}\mathcal{R}(\Pi_N;\Theta)$ by
         the DP algorithm.
    iii) Compute $\hat{p}(\text{FP};\Pi_N^*)$ and $p(\text{FN};\Pi_N^*)$. Set
         $C_{\text{FP}}^{min}=C_{\text{FP}}$ if $p(\text{FP};\Pi_N^*)\geq\alpha$, $C_{\text{FP}}^{max}=C_{\text{FP}}$
         otherwise. Set $C_{\text{FN}}^{min}=C_{\text{FN}}$ if $p(\text{FN};\Pi_N^*)\geq\beta$,
         $C_{\text{FN}}^{max}=C_{\text{FN}}$ otherwise.
· Stop and return $\Pi_N^*$ if
         $|p(\text{FP};\Pi_N^*)-\alpha|+|p(\text{FN};\Pi_N^*)-\beta|<\delta$,
  where $\delta$ is some predefined error tolerance.
```

**Algorithm 1:** Learning $\Pi$ for given FNR and FPR.

and then in the order of $i=N-1,\cdots,1$ trace back,

$$\tau_i^*=\mathbf{T}_{i+1}[\tau_{i+1}^*]. \tag{43}$$

To run the DP algorithm for different $\Theta=(C_{\text{FP}},C_{\text{FN}},\lambda)$ which are either specified or needed to be searched for a given pair of accuracy measure $(\alpha,\beta)$, we create three DP tables for fast computation of $R(\tau_i;\Theta)$ and $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$ respectively (see details in the supplementary material).

*Refinement of the DP Solution.* Given the near-optimal decision policy from the DP algorithm, we can compute its groundtruth risk value based on the original risk function since the number of FNs and FPs, and the number of samples exited at each stage can be counted exactly. Then, we adopt Gibbs-sampling like adjustments to the thresholds $\tau_i^-$'s and $\tau_i^+$'s one by one, and we accept an adjustment if it could further reduce the original risk.

## 5.3 Decision Policy for Given FNR and FPR

Using the same DP algorithm for $\Theta=(C_{\text{FP}},C_{\text{FN}},\lambda)$, we can compute the near-optimal decision policy for the second setting where the reachable FPR and FNR, i.e., $\Theta=(\alpha,\beta,\lambda)$ are given. We use the Algorithm. 1 to search for the corresponding $C_{\text{FP}},C_{\text{FN}}$ through binary search. Without loss of generality, we set $\lambda=1$ here.

## 5.4 Decision Policy for Given Computing Budget

We can also use the DP algorithm to compute the decision policy for a given total computing budget – the third way of specifying a decision policy.

Suppose that the total computing budget is $\mathbb{C}$, and then our objective is,

$$\begin{aligned}&\min_{\Pi_N}\mathcal{L}(\Pi_N;C_{\text{FP}},C_{\text{FN}})\\ &\qquad\text{subject to }\mathcal{C}(\Pi_N)\leq\mathbb{C},\end{aligned} \tag{44}$$

where the loss $\mathcal{L}(\cdot)$ is defined in Eqn. (13) and the cost $\mathcal{C}(\cdot)$ in Eqn. (16).

| | | | | | Classification | | | Detection | |
|---|---|---|---|---|---|---|---|---|---|
| Method | #Feat | #Stages | $C_{\text{FP}}$ | $C_{\text{FN}}$ | FPR | FNR | CostPerEx | AP | CostPerWindow |
| AdaBoost | 1146 | 1 | / | / | 0.0017972 | 0.3030 | 1146 | 0.815 | 1146 |
| AdaBoost Cascade [1] | 2497 | 20 | / | / | 0.0046214 | 0.3284 | 297.84 | 0.807 | 37.279 |
| AdaBoost SoftCascade [7] | 2135 | 20 | / | / | 0.0029214 | 0.3369 | 227.84 | 0.805 | 36.109 |
| AdaBoost Policy $\Pi$ | 1146 | 20 | 0.0357 | 0.5000 | 0.0018094 | 0.3078 | **162.72** | 0.809 | **27.459** |

TABLE 1
Comparison between the cascade and our decision policy ($\Pi$) on human face AdaBoost classifier. Our decision policy outperforms the cascade methods in speed sigificantly on both the classification testing dataset and the CMU+MIT detection benchmark with similar accuracy. See text for details.

For all feasible solutions $\Pi_N$'s to Eqn. (44), we can adopt the same upper bound relaxation presented in Sec. 5.1 with $\Theta = (C_{\text{FP}}, C_{\text{FN}}, 1)$. Then, we modify $B_i[\tau_i]$ in Eqn. (41) of the DP algorithm to set

$$B_i[\tau_i] = +\infty, \text{ if } \hat{\mathcal{C}}(\Pi_N) > \mathbb{C}, \tag{45}$$

where $\hat{\mathcal{C}}(\Pi_N)$ is the upper bound of the computing cost which can be calculated based on the first parts in Eqn. (36) and Eqn. (40) respectively.

## 6 Experiments

In the experiments, we learn decision policies for the three popular detectors: AdaBoost, SVM and DPM, and compare with their corresponding cascade methods [1], [7], [27].

**Settings:** In all the comparison experiments, we learn the decision policies based on the training FPR and FNR $(\alpha, \beta)$, and the corresponding $(C_{\text{FP}}, C_{\text{FN}})$ are searched using the algorithm stated in Sec.5.3. According to Eqn. (20), we observe in our experiments that for the final comparison results of decision policies only the ratios $\frac{C_{\text{FP}}}{\lambda}$ and $\frac{C_{\text{FN}}}{\lambda}$ matter, so we set $\lambda = 1.0$ for simplicity. Furthermore, we show the normalized values of searched $C_{\text{FP}}$ and $C_{\text{FN}}$ by $S$ in all the figures and tables. Note that unlike our method, the popular cascade methods [1], [7], [27] are not adjusted (or optimized) to given parameters $\Theta$.

### 6.1 Decision Policy for AdaBoost Classifier

In this experiment, we learn the decision policy for the AdaBoost classifier trained for human face detection. We compare with the original "hard" cascade method [1] and the soft cascade method [7] which are the two of the most popular cascade methods for face detection.

**The training and testing data.** *The training dataset* consists of 11020 positive human face images (with the size being $20 \times 20$ pixels) and 15366 background images which do not contain human faces. The background images are not cropped image patches but the whole images having different sizes (such as $100 \times 80$ pixels and $1024 \times 768$ pixels). *The testing set* for classification includes 7092 human face images and 81794 negative image patches randomly collected from the background images consisting of animal faces, building, wall, grass and tree clutter. We use the CMU+MIT human face detection benchmark [42] for evaluating the detection performance, which consists of 130 testing images where 511 human faces appear.

**Features, weak classifiers and the computational costs.** We use the same 4 types of Haar features used in [1], and adopt the decision stump as the weak classifier. In the experiments, we use the integral image to compute the Haar features as done in [1]. So, the computational cost for each weak classifier is the same and the computational cost for each stage is proportional to the number of weak classifiers included in that stage.

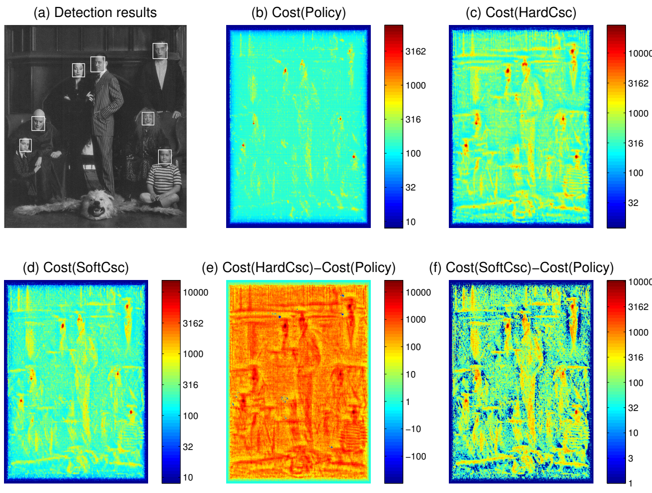

Fig. 5. Illustration of the pixel-based computational costs compared between our policy (b), the hard cascade (c), and the soft cascade (d) on a testing image in MIT+CMU face detection dataset. All the three methods can detect all faces in the testing image (a). (e) and (f) show the computing cost difference between our policy and the hard cascade and the soft cascade respectively. To record the work load at each image pixel location, starting from a zero image, every sliding window tested by the detector contributes an amount of intensity proportional the number of Haar features (as indicated by the color bar on the right of sub-figures from (b) to (f)) applied to that window. For the smallest detection windows (i.e., the windows in the bottom of the testing image pyramid, i.e., the original image), this intensity is concentrated in a single pixel at the center of the window; for larger windows (i.e., windows in the higher levels of the testing pyramid), it is spread out over a proportionally larger area.

**Training the cascade of AdaBoost classifiers.** We train the cascade consisting of 20 stages for both the "hard" and soft cascade using the publicly available OpenCV package (http://opencv.org). At each stage, we use the same number of negatives as that of positives, set the minimal hit rate to 0.999 and the maximal FPR to 0.5, and use the equal data

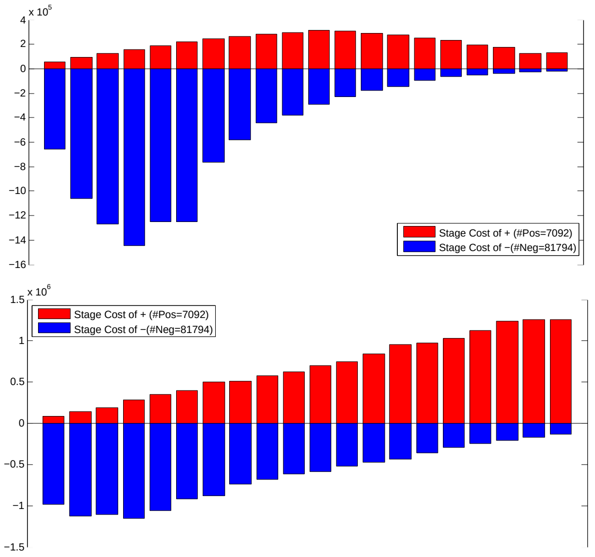

Fig. 6. Comparisons of the stage computing costs between our policy (top) and the hard cascade (bottom) on human face classification testing dataset (red bars for positives and blue ones for negatives where for illustration we use the minus computing costs of negatives in the plots). The stage computing cost of the $i$-th stage is computed as the product of the number of examples exited (i.e., $\#S_i$ in Eqn.(17)) and the cumulative computational cost of the current stage (i.e., $\sum_{j=1}^{i} c(G_j)$ in Eqn.(14)).

weight at the beginning. For the first stage, the negatives are randomly cropped from the background images. At step $n = 2, \cdots, 20$, the negative dataset of step $n-1$ is first shrunk by removing those which are correctly classified by the already trained cascade, and then the false positives (i.e., hard negatives) are mined from the background images to fill the gap. The trained "hard" cascade consists of 2497 boosted weak classifiers in total and the soft cascade consists of 2135 boosted weak classifiers. Note that the soft cascade needs an additional validation dataset to tune the thresholds, and we use 5000 positives and 40,000 negative examples, and the stages are independent of each other in the "hard" cascade, instead of using the cumulative scores.

**Training the AdaBoost classifier and learning the decision policy.** To train the single Adaboost classifier, we adopt the iterative training procedure and set the minimal hit rate to 0.999 and the maximal FPR to 0.0001 at each iteration. In the first step, it is actually the same as training the first stage of the cascade. After that, we keep all the negatives and use the current classifier to mine the same number of false positives as that of positives from the background images. Then, we use all the negatives to start over to train a new AdaBoost classifier. The iteration stops when no more false positives can be found in the background images, resulting in $117,493$ negatives in total. The final AdaBoost classifier consists of $1,146$ boosted weak classifiers and FNR $\beta_{\text{train}} = 0$ and FPR $\alpha_{\text{train}} = 1.1916e{-}004$ on the training dataset. By doing this, we are first maximizing the detection accuracy and then let the learned decision policy to advance its computational efficiency to account for different parameter settings.

**The learned decision policy.** To learn the decision policy, we first divide the $1,146$ boosted weak classifiers into 20 subsets which consist of 8, 13, 18, 23, 28, 33, 38, 43, 48, 53, 58, 63, 68, 73, 78, 83, 88, 93, 98, 139 ones respectively. Here, the number of weak classifiers in each step increases linearly (e.g., 5 for the first 19 steps). Note that the configuration of the decision policy, i.e. the cardinality of each subset, is not particularly chosen, but can be very flexible for different situations (which can not be easily specified in training the cascade).

We summarize the results in Table. 1. Overall, the decision policy outperforms the two cascade methods in speed with similar detection accuracy. *The computational efficiency* is measured by average #Feat tested per example for classification, and average #Feat tested per sliding window for detection. *The accuracy* is measured by FPR and FNR for classification and Average Precision (AP) for detection. For example, in detection, the two cascade methods and our decision policy obtain similar APs, but our decision policy saves about 10 Haar feature evaluations per sliding window on average. Fig. 5 shows the comparisons of the pixel-based computational costs between our policy and the two cascade methods on a testing image in MIT+CMU face dataset. Fig. 6 shows the stage-wise computing costs for our decision policy and the hard cascade. Since the cascade does not accept examples in early stages, its stage computing costs of positives are consistently larger than the decision policy. So, our policy will obtain more speed-up when the positive and negative population ratio increased as we show in Sec. 6.4 later.

### 6.2 Decision Policies for SVM classifier

In this experiment, we learn the decision policy for linear SVM classifier trained on INRIA person dataset [3]. We compare with the probably approximate admissible (PAA) threshold [27], which selects the rejection thresholds using,

$$\tau_i^- = \min_{\{x; f(x) \geq t_1, x \in D^+\}} f^i(x), \qquad (46)$$

where $t_1$ is predefined to focus on high-scoring positive examples.

**The training and testing data.** In INRIA person dataset, there are 1208 persons in the training dataset and 566 persons in the testing dataset. We use 1119 out of the 1208 person images for training, and 552 out of 566 person images for classification testing by excluding those labeled as difficult ones or being smaller than the trained model size ($120 \times 40$ pixels). For evaluating classification, we randomly crop 49869 negative image patches from a set of collected street scene and indoor scene images in which no persons appear. For evaluating detection, we use all the images containing persons together with the 453 background images in which no persons appear.

**SVM classifier.** We train the linear SVM classifier using the modified 32-dimensional HOG features [4] with the template size being $15 \times 5$ cells (each cell is of $8 \times 8$

| | | | | | Classification | | | Detection | |
|---|---|---|---|---|---|---|---|---|---|
| Method | #Cell | #Stages | $C_{\text{FP}}$ | $C_{\text{FN}}$ | FPR | FNR | CostPerEx | AP | Avg.Cost |
| SVM | 150 | 1 | / | / | 0.0185 | 0.0052 | 150 | 0.80 | 150 |
| SVM PAA | 150 | 20 | / | / | 0.0134 | 0.019 | 100.7 | 0.79 | 75.8 |
| SVM policy $\Pi$ | 150 | 20 | 0.00031 | 0.01852 | 0.0180 | 0.0069 | **29.9** | 0.788 | **31.3** |

TABLE 2
Comparison between PAA [27] and our policy $\Pi$ of the SVM classifier on the INRIA person dataset [3].

| | aero | bike | boat | bttle | bus | car | mbik | train | bird | cat | cow | dog | hrse | sheep | pers | plant | chair | tble | sofa | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PAA | 32.3 | 57.58 | 15.51 | 25.1 | 51.2 | 54.15 | 47.4 | 44 | 10.5 | 17.9 | 24 | 11.5 | 55.35 | 22.5 | 43.35 | 14.3 | 21 | 25.3 | 34 | 41.2 |
| Policy $\Pi$ | 32.1 | 57.4 | 15.3 | 25.1 | 51.1 | 54 | 47.22 | 43.8 | 10 | 17.6 | 23.7 | 11.2 | 55.1 | 22.5 | 43.1 | 14.3 | 20.8 | 25.1 | 34 | 41.1 |
| Speedup | 1.62 | 1.78 | 1.68 | 1.32 | 1.51 | 1.31 | 1.63 | 1.57 | 1.31 | 1.51 | 1.69 | 1.56 | 1.73 | 1.44 | 1.82 | 1.36 | 1.69 | 1.53 | 1.97 | 1.4 |

TABLE 3
Comparison between PAA [27], and our policy $\Pi$ of the DPMs trained and tested on PASCAL VOC2007 dataset [41]. The first two rows show the average precision (APs) in detection. In the third row, we use the speed-up ratio based on total running time on the testing dataset as efficiency evaluation for comparing with [45].

pixels) using the publicly available code [45], and we have $15 \times 5 \times 32$ weight parameters and 1 bias term in the SVM classifier. During training, we use the hard negative mining strategy as done in [3], [4] and obtain 114635 negatives in total. Similar to [27], we also specify a simplified classifier by projecting the trained SVM classifier into the top 5 principal components pooled from the whole training dataset, resulting in $15 \times 5 \times 5$ weight parameters. So, we have 150 cells in total which are reordered based on the criteria stated in Sec. 2.3, and then organized into 20 subsets with the size of the first 19 stages being 7 cells and the last stage being 17 cells. *Computational cost setting*. For simplicity, we treat the computing cost of a cell in the simplified classifier and that of the trained one as being equal (due to the projection overhead needed in the simplified classifier). So, the computational cost of a stage is proportional to the number of cells.

The results are summarized in Table. 2. Our decision policy outperforms the PAA method in speed in both classification and detection with similar detection accuracy. Our policy can outperform the PAA method in speed mainly due to (i) we reorder the cells in terms of the ratio between the statistical power and the computational cost (as stated in Sec. 2.3), while the PAA method only takes into account the variance of the positive scores (as done in the latest release code [45]); By the reordering, we obtain more stable trajectories of the training examples in learning the policy, which is better for solving the policy by minimizing the upper bound; and (ii) our DP algorithm chooses all the thresholds jointly to minimize the empirical global risk with computational cost taken into account explicitly, while the PAA selects the rejection thresholds based on the high-scored positive sub-population (see Eqn. 46).

## 6.3 Decision Policies for DPM

In this experiment, we learn decision policies for the pre-trained DPMs in [45] on the 20 object classes in PASCAL VOC2007 [41], and compare them with the PAA cascade method [27] as stated in Sec.6.2.

**The models, the order of parts, and the computational cost settings.** The pretrained model for each object class is a mixture of DPMs consisting of 6 components. Each component has one root filter and 8 part filters (which are the same size, i.e., $6 \times 6$ HOG cells). In PAA [27], for each component, the order of a part $i$ (out of the 8 parts) is determined based on the variance of scores on positives without considering part $i$ itself. In learning decision policy, we reorder the parts based on the method stated in Sec.2.3. As done in [27], we also use the PCA projected model (using top 5 PCs). The computational costs for a part in the projected model and in the original model are $6 \times 6 \times 5$ and $6 \times 6 \times 32$ (32 is the length of HOG feature vector). The computational costs for the roots are computed in similar way with respect to their own filter sizes.

**Training and testing dataset.** To collect the positive training data, we reuse the same code in [45] that collect scores of each part and the root. For the negative training data, we run the detection of the pretrained model on the negative images and randomly choose 100 samples (at random scales and locations) from each negative images (typically, we have about 200,000 negative examples). Then, we use them to learn the decision policy. We test the learned decision policy on detection only using the PASCAL VOC protocol, and use the speed-up of the total running time of detection (which does not take into account the HOG pyramid computation and the PCA projection of the pyramid) as efficiency evaluation for the comparison with the PAA method [27]. We reused the DPM cascade code in [45] and implemented the detection module based on our policy using similar coding scheme for fair comparisons (e.g., we also did not parallelize the detection among different mixture components).

The results are summarized in Table. 3. Our policy speeds up the PAA cascade on all the 20 categories consistently with similar performance.

## 6.4 Efficiency of Decision Policies over the Population Ratios

In the three experiments above, the positive and negative population ratio is based on a training or testing dataset which is fixed and largely asymmetric (i.e., the number of negatives is more than the number of positives). As is stated in the introduction, the learning framework proposed in this paper is able to address the issues of adaptation of object

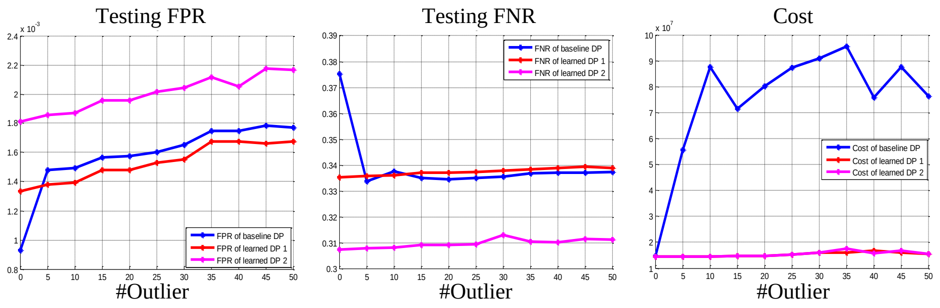

Fig. 8. Comparison of robustness to outliers between the naive decision policy and the learned decision policy of human face AdaBoost classifier on the classification testing data. Blue curves are for the baseline decision policy and red and magenta curves are for two of our learned decision policies (which prefer relatively smaller FPR and FNR respectively). The naive policy is more sensitive to the outliers in terms of the computational efficiency.

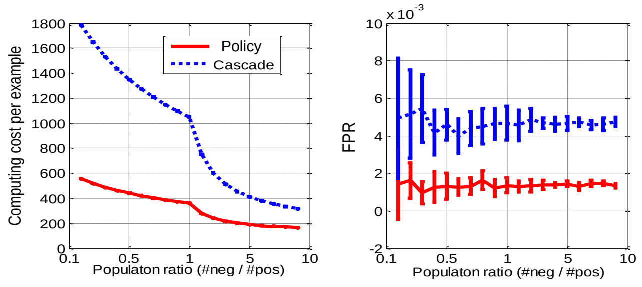

Fig. 7. Efficiency of the policy (red curves) over the population ratios compared with cascade (blue curves). *Left:* The computing costs per example over different population ratios. *Right:* The comparisons on FPR by mean(horizontal curve)-std (vertical bar) plots.

detectors when the population ratio changes dynamically. Our policy can obtain larger speed-up when the positive and negative population ratio increases. Here, we show this by manipulating the ratios in human face classification dataset.

We test 19 different positive and negative population ratios: $\{0.1, 0.2, \cdots, 1, 2, 3, \cdots, 10\}$. The positive set does not change (i.e., the same as in Sec. 6.1). For each ratio, we randomly sample the negatives from the whole negative testing dataset. Then, we learn the decision policy and evaluate it. For each ratio, we run 10 times. Fig. 7, we compare the computational efficiency (the left sub-figure) and the detection accuracy (the right sub-figure) of our decision policy and the cascade. Our policy obtains more speed-ups when the positive and negative population ratio is large, and better and more stable accuracy.

### 6.5 Robustness of Decision Policies to Outliers

To show the robustness of our policy, we compare with the naive decision policy which select $\tau_i^-$ using the PAA method [27] (see Sec. 6.2) and similarly choose $\tau_i^+ = \max_{\{x; f(x)<t_0, x\in D^-\}} f^i(x) + \epsilon$ (where $\epsilon$ is a very small positive value such as $0.00001$ in our experiments). we demonstrate this by adding outliers in human face classification dataset. To add outliers to the positive dataset, we randomly select patches from the negative images. To add outliers to the negative dataset, we randomly select face images. Fig. 8 compares the robustness to outliers in both positive and negative testing datasets between the naive decision policy and the learned decision policy, which shows that the learned decision policy is consistently more robust with different number of outliers being used.

## 7 Discussion

This paper presents a framework for computing a decision policy (a sequence of two-sided thresholds) in terms of optimizing an upper bound of the risk function. The latter is a weighted sum of the loss (missing detection and false alarm) and computational cost, with the weights being specified ahead or updated on-the-fly. The decision policy is part of a bigger computational problem in computer vision — *goal-guided cost-sensitive computing*, and thus it should be integrated with other computing and learning modules. In the following, we discuss two possible directions to extend the work.

Firstly, the ordering of features in object detection, e.g. Adaboost, is mostly done in greedy ways and thus is not optimal, and the decision policy is computed after this order is decided. One may consider combining the two computing processes in an iterative way to achieve more effective detectors. One possible algorithm is Markov Chain Monet Carlo (MCMC) which consists of various reversible jumps for switching orders of features and replacing features in term of minimizing the loss function. The decision policy can be one step of this iterative process.

Secondly, the study of decision policies is closely related to a long-standing computing vision problem [46], [47], i.e., schedule bottom-up/top-down computing processes in a big hierarchical model for scene understanding and image parsing. In our previous work [16], we presented a numerical study on ordering the bottom-up/top-down processes (i.e. the $\alpha, \beta, \gamma$-processes ) for object parsing. These processes were scheduled in a greedy way like best-first-search in And-Or graphs [48]. In the current work, the decision policy is optimized in a linear chain structure, we plan to extend the policy to the And-or graph so that we can optimally schedule the $\alpha, \beta, \gamma$-processes together with the early reject or accept of each node in the And-Or graph. In this bigger framework, contextual information will become available to update the population ratios of the positives and the negatives, and will be used for updating the decision policies as we argued in the beginning of the paper.

## ACKNOWLEDGMENTS

This work is supported by DARPA MSEE grant FA 8650-11-1-7149, NSF IIS1018751 and MURI grant ONR N00014-10-1-0933. We thank Dr. Ying Nian Wu for helpful discussions.

## REFERENCES

[1] P. Viola and M. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.
[2] V. Vapnik, *Statistical learning theory*. Wiley, 1998.
[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[4] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, vol. 32, no. 9, pp. 1627 – 1645, 2010.
[5] H. Pirsiavash and D. Ramanan, "Steerable part models," in *CVPR*, 2012, pp. 3226–3233.
[6] X. Song, T. Wu, Y. Jia, and S.-C. Zhu, "Discriminatively trained and-or tree models for object detection," in *CVPR*, 2013.
[7] L. D. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *CVPR*, 2005.
[8] M. J. Saberian and N. Vasconcelos, "Learning optimal embedded cascades," *PAMI*, vol. 34, no. 10, pp. 2005–2018, 2012.
[9] A. Barbu, "Multi-path marginal space learning for object detection," *Theodoridis, S and Chelappa, R (Eds.), E-Reference Signal Processing. Elsevier*, 2013.
[10] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *PAMI*, vol. 31, no. 12, pp. 2129–2142, 2009.
[11] I. Kokkinos, "Rapid deformable object detection using dual-tree branch-and-bound," in *NIPS*, 2011.
[12] H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *PAMI*, vol. 33, no. 2, pp. 294–309, 2011.
[13] ——, "Risk minimization, probability elicitation, and cost-sensitive svms," in *ICML*, 2010.
[14] M. R. Amer, D. Xie, M. Zhao, S. Todorovic, and S. C. Zhu, "Cost-sensitive top-down/bottom-up inference for multiscale activity recognition," in *ECCV*, 2012.
[15] J. Sochman and J. Matas, "Waldboost - learning for time constrained sequential detection," in *CVPR*, 2005.
[16] T. F. Wu and S.-C. Zhu, "A numerical study of the bottom-up and top-down inference processes in and-or graphs," *IJCV*, vol. 93, no. 2, pp. 226–252, 2011.
[17] A. Grubb and J. A. D. Bagnell, "Speedboost: Anytime prediction with uniform near-optimality," in *AISTATS*, 2012.
[18] B. Póczos, Y. Abbasi-Yadkori, C. Szepesvári, R. Greiner, and N. R. Sturtevant, "Learning when to stop thinking and do something!" in *ICML*, 2009.
[19] T. Gao and D. Koller, "Active classification based on value of classifier," in *NIPS*, 2011.
[20] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *IJCV*, vol. 77, no. 1-3, pp. 65–86, 2008.
[21] M. Chen, Z. E. Xu, K. Q. Weinberger, O. Chapelle, and D. Kedem, "Classifier cascade for minimizing feature evaluation cost," *JMLR - Proceedings Track*, vol. 22, pp. 218–226, 2012.
[22] G. Blanchard and D. Geman, "Hierarchical testing designs for pattern recognition," *Ann. Statist.*, vol. 33, no. 3, pp. 1155–1202, 2005.
[23] H. Schneiderman, "Feature-centric evaluation for efficient cascaded object detection," in *CVPR*, 2004.
[24] R. Xiao, H. Zhu, H. Sun, and X. Tang, "Dynamic cascades for face detection," in *ICCV*, 2007.
[25] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Fast asymmetric learning for cascade face detection," *PAMI*, vol. 30, no. 3, pp. 369–382, 2008.
[26] L. Lefakis and F. Fleuret, "Joint cascade optimization using a product of boosted classifiers," in *NIPS*, 2010.
[27] P. Felzenszwalb, R. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, 2010.
[28] J. Yan, Z. Lei, L. Wen, and S. Z. Li, "The fastest deformable part model for object detection," in *CVPR*, 2014.
[29] Y. Amit, D. Geman, and X. D. Fan, "A coarse-to-fine strategy for multiclass shape detection," *PAMI*, vol. 26, no. 12, pp. 1606–1621, 2004.
[30] S. Gangaputra and D. Geman, "A design principle for coarse-to-fine classification," in *CVPR*, 2006.
[31] H. Sahbi and D. Geman, "A hierarchy of support vector machines for pattern detection," *JMLR*, vol. 7, pp. 2087–2123, 2006.
[32] P. F. Felzenszwalb and D. A. McAllester, "The generalized a* architecture," *J. Artif. Intell. Res. (JAIR)*, vol. 29, pp. 153–190, 2007.
[33] I. Kokkinos and A. Yuille, "Inference and learning with hierarchical shape models," *IJCV*, vol. 93, no. 2, pp. 201–225, 2011.
[34] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *ICML*, 2000.
[35] A. Wald, *Sequential Analysis*. Wiley, New York, 1947.
[36] D. Seigmund, *Sequential Analysis, Test and Confidence Intervals*. Springer-Verlag, 1985.
[37] O. Pele and M. Werman, "Robust real-time pattern matching using bayesian sequential hypothesis testing," *PAMI*, vol. 30, no. 8, pp. 1427–1443, 2008.
[38] M. Zhu, N. Atanasov, G. Pappas, and K. Daniilidis, "Active Deformable Part Models Inference," in *ECCV*, 2014.
[39] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," in *CVPR*, 2014.
[40] T. Wu and S.-C. Zhu, "Learning near-optimal cost-sensitive decision policy for object detection," in *ICCV*, 2013.
[41] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results."
[42] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *PAMI*, vol. 20, no. 1, pp. 23–38, 1998.
[43] C. Elkan, "The foundations of cost-sensitive learning," in *IJCAI*, 2001, pp. 973–978.
[44] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, pp. 289–337, 1933.
[45] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models, release 5," 2012.
[46] S. Ullman, "Visual routines," *Cognition*, vol. 18, pp. 97–159, 1984.
[47] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, pp. 520–522, 1996.
[48] S.-C. Zhu and D. Mumford, "A stochastic grammar of images," *Found. Trends. CGV*, vol. 2, no. 4, pp. 259–362, 2006.

**Tianfu Wu** received a Ph.D. degree in Statistics from University of California, Los Angeles (UCLA) in 2011. He is currently a researcher assistant professor in the center for vision, cognition, learning and art (VCLA) at UCLA. His research interests focus on: (i) Learning hierarchical and compositional models from weakly-labeled big data and (ii) Scheduling bottom-up/top-down computing processes in big hierarchical models.

**Song-Chun Zhu** received a Ph.D. degree from Harvard University in 1996. He is currently a professor of Statistics and Computer Science at UCLA, and the director of the Center for Vision, Cognition, Learning and Art. He has published over 160 papers in computer vision, statistical modeling and learning, cognition, and visual arts. He received a number of honors, including the J.K. Aggarwal prize from the Int'l Association of Pattern Recognition in 2008, the David Marr Prize in 2003 with Z. Tu et al.,twice Marr Prize honorary nominations in 1999 and 2007, He received the Sloan Fellowship in 2001, a US NSF Career Award in 2001, and an US ONR Young Investigator Award in 2001. He received the Helmholtz Test-of-time award in ICCV 2013, and he is a Fellow of IEEE since 2011.

# Supplementary Material

Tianfu Wu and Song-Chun Zhu

## 1 The Additive Scoring Functions

In this section, we overview three popular object detectors, AdaBoost, SVM and DPM to set up the background complementary to Sec.2.3 in the manuscript. All the three detectors use an offline trained additive scoring function with $T$ basic terms.

**AdaBoost.** It learns a scoring function (strong classifier),

$$f(x) = \sum_{t=1}^{T} g_t(x),$$

which is the sum of $T$ boosted weak classifiers $g_t(x)$'s. In discrete AdaBoost (DAB), each boosted weak classifier represents a weighted binary rule, $g_t(x) = w_t h_t(x)$, and $h_t(x)$ is often implemented with a decision stump $h_t(x) = \text{sign}[\phi_t(x) - \tau_t]$ where $\phi_t(x)$ represents a scalar feature response (e.g., the Haar feature response [1]) and $\tau_t$ is the corresponding threshold.

**SVM.** Linear SVM learns a max-margin scoring function $f(x) =< \omega, \phi(x) > +b$ where $\omega$ is the weight parameter vector, $\phi(\cdot)$ is a feature vector representation, $< \cdot, \cdot >$ is the inner product, and $b$ is a bias term. By dividing the feature vector into a set of $T$ blocks, we have,

$$f(x) = \sum_{t=1}^{T} < \omega_t, \phi_t(x) > +b.$$

The blocks can be decided based on the types of features in a heterogeneous long feature vector $\phi(\cdot)$ or the partition of the spatial domain of a sliding window if $\phi(\cdot)$ is homogeneous, such as the blocks used in the HOG feature representation [2], [3]. We have $g_t(x) =< \omega_t, \phi_t(x) >$. For nonlinear kernel SVM, we can use the support vectors as basic additive terms [4], and then similar steps follow.

**Deformable Part-based Model (DPM)** [3]. The scoring function of a DPM with one root and $T-1$ parts is,

$$\begin{aligned} f(x) = &< \omega_1, \phi_1(x) > \\ &+ \sum_{t=2}^{T} \max_{\delta_t}(< \omega_t, \phi_t(x; \delta_t) > - d_t(\delta_t)) + b \end{aligned}$$

where $\omega_t$'s ($t = 1, \cdots, T$) are the weight parameter vectors, $\phi_t(\cdot)$'s are appearance feature vectors (such as the HOG feature) and the feature of parts $\phi_t(\cdot)$ ($t = 2, \cdots, T$) are often extracted at twice the spatial resolution relative to that of root node ($\phi_1(\cdot)$). $\delta_t$ is the displacement of the $t$-th part relative its anchor location, $d_t(\delta_t)$ is the deformation cost, and $b$ is the bias term.

- *T.F. Wu is with the Department of Statistics, University of California, Los Angeles. E-mail: tfwu@stat.ucla.edu*
- *S.C. Zhu is with the Department of Statistics and Computer Science, University of California, Los Angeles. E-mail: sczhu@stat.ucla.edu*

## 2 The DP Tables Used in Online Computing of Policy

To run the DP algorithm (Sec.5.2 in the manuscript) for different $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$ which are either specified or needed to be searched for a given pair of accuracy measure $(\alpha, \beta)$, we further create three DP tables for fast computation of $R(\tau_i; \Theta)$.

Given the $K \times N$ matrix defined in Sec.4 in the manuscript, the number of states which all $\tau_i$'s can take is $K \cdot (K+1)/2$ because we have $\tau_i^- = 1, \cdots, K$ and $\tau_i^+ = \tau_i^-, \cdots, K$ (i.e. $\tau_i^+ \geq \tau_i^-$). Then, we can pre-compute three 2-dimensional DP tables,

$$\begin{aligned} \mathbb{T}_1[i, \tau_i] &= \mathbf{C}_i \cdot [S(\underline{\tau_i^-}) + S(\overline{\tau_i^+})]/S \\ \mathbb{T}_2[i, \tau_i] &= S^-(\overline{\tau_i^+})/S \\ \mathbb{T}_3[i, \tau_i] &= S^+(\underline{\tau_i^-})/S \end{aligned}$$

Where we use $\tau_i$ as the index of the states. Then, the computational complexity for evaluating the risk $R(\tau_i; \Theta)$'s given any $(C_{\text{FP}}, C_{\text{FN}})$ is the same and very fast (three matrix multiplications and one matrix addition).

In the same way, we utilize three DP tables in computing the risk $\mathbb{R}(\tau_i, \tau_{i-1}; \Theta)$,

$$\begin{aligned} \mathbb{T}_4[i, \tau_{i-1}, \tau_i] = &\frac{\mathbf{C}_i}{S} \cdot [S(\underline{\tau_i^-}; \underline{\tau_{i-1}^-}) + S(\overline{\tau_i^+}; \underline{\tau_{i-1}^-}) \\ &+ S(\underline{\tau_i^-}; \overline{\tau_{i-1}^+}) + S(\overline{\tau_i^+}; \overline{\tau_{i-1}^+})] \\ \mathbb{T}_5[i, \tau_{i-1}, \tau_i] = &\frac{1}{S}[S^-(\overline{\tau_i^+}; \underline{\tau_{i-1}^-}) + S^-(\overline{\tau_i^+}; \overline{\tau_{i-1}^+})] \\ \mathbb{T}_6[i, \tau_{i-1}, \tau_i] = &\frac{1}{S}[S^+(\underline{\tau_i^-}; \underline{\tau_{i-1}^-}) + S^+(\underline{\tau_i^-}; \overline{\tau_{i-1}^+})] \end{aligned}$$

The six DP tables are often very sparse which can be implemented by the sparse matrix data structure to save the memory.

In Fig.1, we show tightness of the upper bound empirically on the human face AdaBoost classifier which is consistent across different settings of the loss of misclassification (we set $C_{\text{FP}} \in [0.1, S/S^-]$ and $C_{\text{FN}} \in [0.1, S/S^+]$ and equally sample 100 points for both). We observe the similar tightness for other types of scoring functions. The

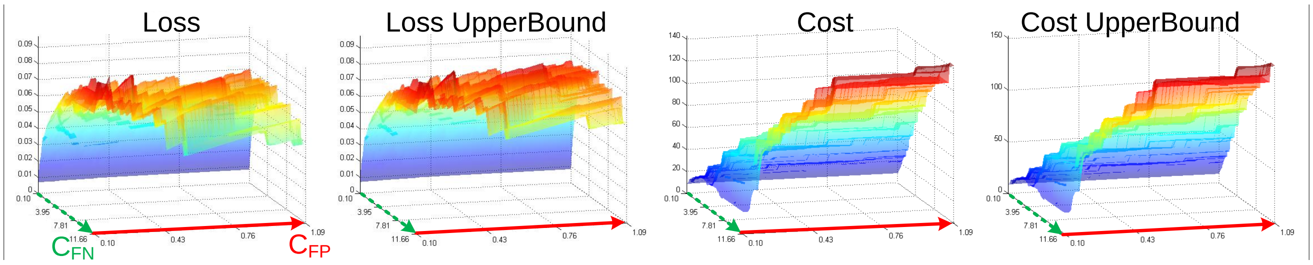

Fig. 1. This figure shows the tightness of the upper bound (Eqn. (39) in the manuscript) for loss and computing cost empirically.

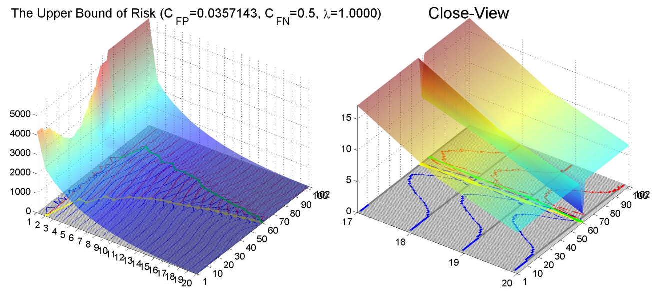

Fig. 2. Illustration of the DP algorithm minimizing the upper bound of our empirical global risk for the decision policy shown in Fig.1 in the manuscript. The surface is plotted based on $B_i(\tau_i)$ in Eqn.(41) in the manuscript.

tightness is due to the statistical observation in Fig.3 in the manuscript.

Fig. 2 illustrates the minimization of the risk upper bound by the DP algorithm for the decision policy shown in Fig.1 in the manuscript.

## 3 Proofs

We present the proofs in the Appendix A, B and C for the Lemma 1 and Proposition 1 in Sec.3.2 in the manuscript, and Proposition 2 in Sec.5.1 in the manuscript respectively.

## Appendix A
## Proof of Lemma 1

*Proof:* Suppose there was a decision policy $\Pi_N^{'}$ solving Eqn. (12) with smaller computational cost,

$$\lambda \cdot \mathcal{C}(\Pi_N^{'}) < \lambda \cdot \mathcal{C}(\Pi_N^*)$$

and better or equal detection accuracy,

$$p(\text{FP}; \Pi_N^{'}) \leq p(\text{FP}; \Pi_N^*) \text{ and } p(\text{FN}; \Pi_N^{'}) \leq p(\text{FN}; \Pi_N^*)$$

So, for the loss, based on Eqn. (13), we have,

$$\begin{aligned}
&\mathcal{L}(\Pi_N'; C_{\text{FP}}, C_{\text{FN}})\\
&= \frac{S^-}{S} p(\text{FP}; \Pi_N') \cdot C_{\text{FP}} + \frac{S^+}{S} p(\text{FN}; \Pi_N') \cdot C_{\text{FN}}\\
&\leq \frac{S^-}{S} p(\text{FP}; \Pi_N^*) \cdot C_{\text{FP}} + \frac{S^+}{S} p(\text{FN}; \Pi_N^*) \cdot C_{\text{FN}}\\
&= \mathcal{L}(\Pi_N^*; C_{\text{FP}}, C_{\text{FN}})
\end{aligned}$$

And, since $\Pi_N^*$ is the optimal solution, we have,

$$\mathcal{R}(\Pi_N^*; \Theta) \leq \mathcal{R}(\Pi_N^{'}; \Theta)$$

That is,

$$\begin{aligned}
&\lambda \cdot \mathcal{C}(\Pi_N^*) + \mathcal{L}(\Pi_N^*; C_{\text{FP}}, C_{\text{FN}})\\
\leq &\lambda \cdot \mathcal{C}(\Pi_N') + \mathcal{L}(\Pi_N'; C_{\text{FP}}, C_{\text{FN}})\\
\leq &\lambda \cdot \mathcal{C}(\Pi_N') + \mathcal{L}(\Pi_N^*; C_{\text{FP}}, C_{\text{FN}})
\end{aligned}$$

Now, we arrive at $\lambda \cdot \mathcal{C}(\Pi_N^*) \leq \lambda \cdot \mathcal{C}(\Pi_N^{'})$ which is conflicted with the original assumption above. Then, by using backward induction, it proves that $\Pi_N^*$ is the solution to the original optimization problem (Eqn. (12)) with $\alpha = p(\text{FP}; \Pi_N^*)$ and $\beta = p(\text{FN}; \Pi_N^*)$. □

## Appendix B
## Proposition 1: Computing $C_{\text{FP}}^{max}$ and $C_{\text{FN}}^{max}$

Given reachable bounds on FPR and FNR $(\alpha, \beta)$, the corresponding near-optimal decision policy, denoted by $\Pi_N^*(\alpha, \beta)$, is sought by searching $\Pi_N^*(C_{\text{FP}}, C_{\text{FN}})$ (which is solved by DP) with,

$$C_{\text{FP}} \in [0, C_{\text{FP}}^{max}] \text{ and } C_{\text{FN}} \in [0, C_{\text{FN}}^{max}]$$

until $p(\text{FP}; \Pi_N^*)$ and $p(\text{FN}; \Pi_N^*)$ are as close with the specified bounds $\alpha$ and $\beta$ as possible.

$C_{\text{FP}}^{max}$ and $C_{\text{FN}}^{max}$ are derived as follows.

Denote by $\Pi_N^N$ the decision policy where a decision is made only after the full score $f(x)$ is obtained and using the offline estimated threshold $\tau = \tau_N^- = \tau_N^+$ (i.e., no early decisions are made). So, the corresponding risk is,

$$\mathcal{R}(\Pi_N^N; \Theta) = \lambda + \frac{S^-(\overline{\tau})}{S} \cdot C_{\text{FP}} + \frac{S^+(\underline{\tau})}{S} \cdot C_{\text{FN}}$$

where $S^-(\overline{\tau}) = S^+(\underline{\tau}) = 0$ if $f(x)$ perfectly separates the positive and negative examples with the threshold $\tau$ being used, or both $S^-(\overline{\tau})$ and $S^+(\underline{\tau})$ take small values with respect to $S$. Then, the risk $\mathcal{R}(\Pi_N^N; \Theta) \approx \lambda$, i.e., the full computational cost (recall that we normalize the computational cost $\mathbf{C}_N = 1$, Eqn. (14) and Eqn. (15)).

By definition, $\mathcal{R}(\Pi_N^N; \Theta)$ is greater or equal to the risk $\mathcal{R}(\Pi_N^*; \Theta)$, we have,

$$\lambda \geq \mathcal{R}(\Pi_N^*; \Theta) > p(\text{FP}; \Pi_N^*) \cdot \frac{S^-}{S} \cdot C_{\text{FP}}$$

Then, if let

$$C_{\text{FP}} = \frac{\lambda \cdot S}{\alpha \cdot S^-} \triangleq C_{\text{FP}}^{max}$$

we can guarantee that $p(\text{FP}; \Pi_N^*) < \alpha$. Similarly, we obtain $C_{\text{FN}}^{max} = \frac{\lambda \cdot S}{\beta \cdot S^+}$. In practice, to avoid the problem of dividing by zero, we let $\alpha \geq \frac{0.1}{S^-}$ and $\beta \geq \frac{0.1}{S^+}$.

## Appendix C Proposition 2: Proof of the upper bound

*Proof:* The intuitive idea is that $\mathbb{R}(\tau_i, \tau_{i-1}; \Theta)$ consider the risk caused by examples double-counted at $\tau_i$ w.r.t. the previous stage $i-1$ only, while $\mathbb{R}(\tau_i, \Pi_{i-1}; \Theta)$ is calculated by taking into account all the first $i-1$ previous stages.

We need to prove $\mathbb{R}(\tau_{i-1}, \tau_i; \Theta) \leq \mathbb{R}(\tau_i, \Pi_{i-1}; \Theta)$. $\mathbb{R}(\tau_{i-1}, \tau_i; \Theta)$ is obtained by substituting the four double-counting terms in $\mathbb{R}(\tau_i, \Pi_{i-1}; \Theta)$, i.e., $S(\cdot; \Pi_{i-1}), S^+(\cdot; \Pi_{i-1})$ and $S^-(\cdot; \Pi_{i-1})$ with their corresponding lower bounds (e.g., $S^+(\underline{\tau_i^-}; \tau_{i-1}) \leq S^+(\underline{\tau_i^-}; \Pi_{i-1})$), and since $\lambda$, $\mathbf{C}_i$, $C_{\text{FP}}$ and $C_{\text{FN}}$ take non-negative values, we obtain the lower bound.

It is straightforward to show $S^+(\underline{\tau_i^-}; \tau_{i-1}) \leq S^+(\underline{\tau_i^-}; \Pi_{i-1})$. Consider FNs by $\tau_i$ individually, $S^+(\underline{\tau_i^-}; \tau_{i-1})$ is the number of FNs double-counted w.r.t. one single previous stage $\tau_{i-1}$, while $S^+(\underline{\tau_i^-}; \Pi_{i-1})$ is counted w.r.t. all the previous stages in $\Pi_{i-1}$. By definition, we have $D^+(\underline{\tau_i^-}; \tau_{i-1}) \subseteq D^+(\underline{\tau_i^-}; \Pi_{i-1})$ and then,

$$S^+(\underline{\tau_i^-}; \tau_{i-1}) \leq S^+(\underline{\tau_i^-}; \Pi_{i-1}).$$

Furthermore, we have,

$$\begin{aligned} D^+(\underline{\tau_i^-}; \tau_{i-1}) &= D^+(\underline{\tau_i^-}; \underline{\tau_{i-1}^-}) \cup D^+(\underline{\tau_i^-}; \overline{\tau_{i-1}^+}), \\ \emptyset &= D^+(\underline{\tau_i^-}; \underline{\tau_{i-1}^-}) \cap D^+(\underline{\tau_i^-}; \overline{\tau_{i-1}^+}). \end{aligned}$$

So, $S^+(\underline{\tau_i^-}; \tau_{i-1}) = S^+(\underline{\tau_i^-}; \underline{\tau_{i-1}^-}) + S^+(\underline{\tau_i^-}; \overline{\tau_{i-1}^+})$.

The same relaxations are taken for other double-counting terms in $\mathbb{R}(\tau_i, \Pi_{i-1}; \Theta)$. We then obtain the lower bound $\mathbb{R}(\tau_1, \tau_{i-1}; \Theta)$, and prove the upper bound $\widehat{\mathcal{R}}(\Pi_N; \Theta)$. □

## References

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

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, vol. 32, no. 9, pp. 1627 – 1645, 2010.

[4] H. Sahbi and D. Geman, "A hierarchy of support vector machines for pattern detection," *JMLR*, vol. 7, pp. 2087–2123, 2006.