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

Tianfu Wu and Song-Chun Zhu
Department of Statistics, University of California, Los Angeles
{tfwu, sczhu}@stat.ucla.edu

Abstract

Many object detectors, such as AdaBoost, SVM and deformable part-based models (DPM), compute additive scoring functions at a large number of windows scanned over image pyramid, thus computational efficiency is an important consideration beside accuracy performance. In this paper, we present a framework of learning cost-sensitive decision policy which is a sequence of two-sided thresholds to execute early rejection or early acceptance based on the accumulative scores at each step. A decision policy is said to be optimal if it minimizes an empirical global risk function that sums over the loss of false negatives (FN) and false positives (FP), and the cost of computation. While the risk function is very complex due to high-order connections among the two-sided thresholds, we find its upper bound can be optimized by dynamic programming (DP) efficiently and thus say the learned policy is near-optimal. Given the loss of FN and FP and the cost in three numbers, our method can produce a policy on-the-fly for Adaboost, SVM and DPM. In experiments, we show that our decision policy outperforms state-of-the-art cascade methods significantly in terms of speed with similar accuracy performance.

1. Introduction

Many popular object detectors, such as AdaBoost [26], SVM [25], deformable part-based models (DPM) [8, 24], are implemented with additive scoring functions. During offline training, those scoring functions are learned by minimizing some regularized convex surrogate functions of the 0/1 loss function; and during detection, they are evaluated at a vast number of sliding windows in an image pyramid. Therefore computational efficiency is a crucial problem in real time applications besides accuracy performance, and has been extensively studied in vision, machine learning, and statistics using a range of methods, including cascade [3, 20, 23, 26], branch-and-bound [13, 14], cost-sensitive learning [15, 16], and sequential probabilistic ratio test [23].

In this paper, we present a generic empirical global risk minimization framework and a DP algorithm for learning near-optimal cost-sensitive decision policy online. The policy consists of a sequence of two-sided thresholds to execute early rejection or early acceptance based on the accumulative scores at each stage. By "cost-sensitive", it means that the decision policy accounts for two factors explicitly:

- *The loss of misclassification*: the losses C_{FN} and C_{FP} for a false negative and a false positive respectively.
- *The cost of computation*: the expected run-time for computing the scoring function sequentially in a sliding window, which is controlled by a parameter λ.

The parameters $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$ are either given according to different vision tasks or searched to satisfy the constraints of some given false positive/negative rates (FNR and FPR). We will elaborate on the conversions between Θ and FNR and FPR later.

The number of features and the scoring functions are all learned by the offline training stage as usual. Our policy

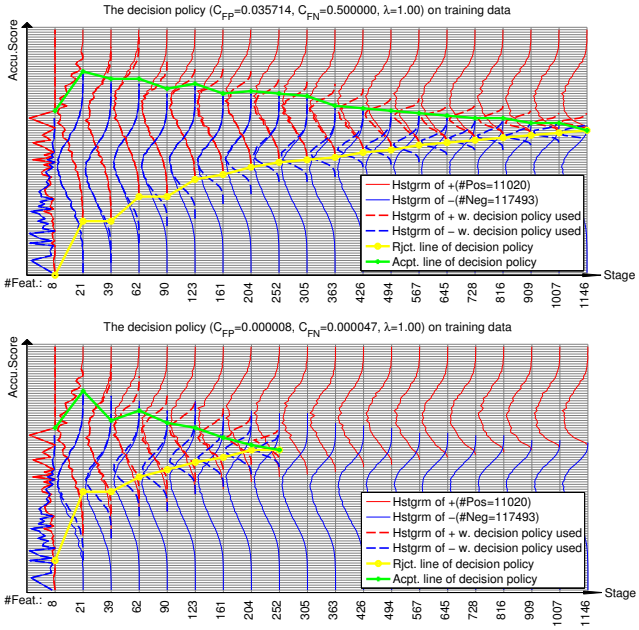

Figure 1. Illustration of two decision policies learned for a human face AdaBoost classifier. The horizontal axis is the $1,146$ boosted weak classifiers divided in 20 stages, and the vertical axis is the accumulative scores. At each stage the solid histograms show the original positive (red) and negative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

does not change these settings and is learned "online" with very flexible design of the configuration of sequential stages (i.e., the number of stages and the basic terms in each stage). The policy is adjustable before the online detection starts. In other words, given a set of parameters $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$, our algorithm can produce a family of policies on the fly. Fig. 1 shows two examples of decision policies learned for the human face AdaBoost classifier based on different parameter settings: the top one is more cautious, while the bottom one is more aggressive in early rejection/acceptance.

Overview. Learning decision policy is formulated under the risk minimization framework with two phases.

(i) *Off-line learning*: We assume that an additive scoring function (AdBoost, SVM or DPM) consisting of T basic additive terms is trained on a dedicated training data set, and remains fixed. For each training example, positive or negative, we record the T scores of individual basic additive terms. We divide the T basic terms into N sequential stages, and compute N histograms of accumulative scores (with K bins) on the positive (see the red histograms in Fig.1) and negative (blue) training datasets respectively. To summarize the off-line knowledge, we define a $K \times N$ matrix which records the trajectories of all training examples along stages (the trajectory is represented by the entry indexes of an example along the stages). We compute four types of statistics (Sec. 4) projected from the trajectories so that we can count the number of FPs and FNs, and calculate the computational cost for any decision policy recursively.

(ii) *On-line learning*: We derive an upper bound of the risk to address the optimization issue caused by the high-order connections between the two-sided thresholds. Then, we propose a DP algorithm to minimize the upper bound efficiently with $\Theta = (C_{\text{FN}}, C_{\text{FP}}, \lambda)$ given to produce the near-optimal decision policy. We empirically show the tightness of the upper bound. With the DP tables precomputed, the optimization is very fast (within several seconds).

The proposed method can address two situations: (i) The expected positive and negative populations of a detector will change online in some big vision tasks in which many object detectors are involved (e.g., image parsing and understanding), and (ii) The parameters Θ will change online in goal-guided searches or answering user queries. So, a fast online algorithm is entailed for automatically adapting the policy to different contexts and goals. This, together with the proposed global risk function, its tight upper bound and the DP solution, distinguishes our work from many others in the literature [3–5, 10, 12, 16, 18, 20, 23].

2. Related work and our contributions

In the literature of object detection, there are three types of methods addressing rapid object detection:

i) The cascade methods [3,4,7,20,22,26,28] utilize early rejections but no early acceptance, except for the WaldBoost [23] and active classification based on value of classifier [10]. In boosting cascade [26], the rejection thresholds of each stage are learned independently, and the training process needs time-consuming trial-and-error tuning. Recently, more principled approaches are proposed on better or optimal design of AdaBoost cascade [3, 4, 20]. [23] assumes strong independence on the output of the weak learners to guarantee optimality. This assumption is often violated in practice. [10] 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 [3] needs a sufficiently large validation set to calibrate the rejection thresholds. In [7], the cascade is learned for a off-line trained DPM. The rejection threshold of each stage is selected so that the probability of a misclassification occurring is bounded from above by a predefined small positive value.

(ii) The coarse-to-fine methods [1, 2, 9, 21] require top-down coarse-to-fine partition of the pose space of an object class, which is often done manually and then used to organize all the tests into a hierarchy (such as a decision tree). The tests are selected based on the ratio between statistical power and computing cost.

(iii) The branch-and-bound method [13, 14]. Instead of adopting the sliding window technique, algorithms based on branch-and-bound directly search sub-windows of object instances through bounding a scoring function from the upper and lower for any given sub-window, which only adapts well for detectors trained using bag-of-word features in [14] or is used to search the state space of a DPM after all filter responses are computed in [13].

Our work utilizes sequential hypothesis testing [23, 27] and is related to the work [17] on sequential image pattern matching based on Hamming distance, 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-wisely 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 considered in this paper, the histograms of accumulative scores on positive and negative training datasets are exploited to learn a decision policy.

Our contributions. This paper makes three main contributions to rapid object detection:

(i) It presents a cost-sensitive formulation to learn the decision policy accounting for the loss of misclassification and the computational cost explicitly, applicable to object detectors with additive scoring functions.

(ii) It derives an upper bound of the empirical global risk function and presents a DP algorithm to minimize it efficiently. It shows the upper bound is tight empirically.

(iii) In experiments, it shows the learned policies of AdaBoost, SVM and DPM outperform the state-of-the-art cascade methods significantly in speed with similar accuracy.

3. Problem formulation

3.1. Definition of the decision policy

Denote by $f(x) = \sum_{t=1}^{T} g_t(x)$ the off-line trained additive scoring function where $g_t(x)$ can be a weighted weak classifier in AdaBoost, a divided block (a weighted support vector) in linear (non-linear) SVM, and a root/part filter in DPM. The order of $g_t(\cdot)$'s is sorted based on the statistical power/computational cost ratio in off-line stage, similar to [2, 12]. We divide the sum of T basic additive terms $g_t(x)$'s into N stages, denoted by $G_i(x)$ $(1 \leq i \leq N)$. For an input sample x, the accumulative score at the i-th stage is defined by $f^i(x) = \sum_{j=1}^{i} G_j(x)$.

Definition 1 (The decision policy): A N-stage decision policy of $f(x)$ is defined by,

$$\Pi_N = (\tau_1, \cdots, \tau_i, \cdots, \tau_N), \tag{1}$$

where τ_i represents two-sided thresholds, $\tau_i = (\tau_i^-, \tau_i^+)$, and $\tau_i^- < \tau_i^+$ 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 learned automatically.

In testing, Π_N makes three possible actions at stage i in terms of accumulative 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) as in Fig. 1. At each stage, τ_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.

3.2. The risk function of a decision policy

Denote by $\mathcal{R}(\Pi_N; \Theta)$ the global risk function of Π_N,

$$\mathcal{R}(\Pi_N; \Theta) = \mathcal{L}(\Pi_N; C_{\text{FP}}, C_{\text{FN}}) + \lambda \cdot \mathcal{C}(\Pi_N), \tag{2}$$

where $\mathcal{L}(\cdot)$ is the expected loss and $\mathcal{C}(\cdot)$ the expected computational cost. We will compute the empirical global risk in training dataset and still use $\mathcal{R}(\cdot)$, $\mathcal{L}(\cdot)$ and $\mathcal{C}(\cdot)$ to denote the empirical counterparts for simplicity.

Let $D = D^+ \cup D^-$ be the training dataset (consisting of a positive dataset D^+ and a negative dataset D^-) used for training $f(x)$ with the cardinalities $S^+ = |D^+|, S^- = |D^-|$ and $S = |D| = S^+ + S^-$.

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

$$\begin{aligned}\mathcal{L}(\Pi_N; C_{\text{FP}}, C_{\text{FN}}) &= E_{X,Y}[\mathbf{1}_{(y=1)} \cdot p(\text{FN}; \Pi_N) \cdot C_{\text{FN}} \\ &\quad + \mathbf{1}_{(y=-1)} \cdot p(\text{FP}; \Pi_N) \cdot C_{\text{FP}}] \\ &\approx \frac{S^+}{S} \cdot \hat{p}(\text{FN}; \Pi_N) \cdot C_{\text{FN}} + \frac{S^-}{S} \cdot \hat{p}(\text{FP}; \Pi_N) \cdot C_{\text{FP}}, \end{aligned} \tag{3}$$

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

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

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

These two empirical probabilities are calculated by exploiting the trajectories of training examples in Sec. 4.

The expected cost of computation $\mathcal{C}(\Pi_N)$. Denote by $c(G_i)$ the computational cost of an individual stage i (which is fixed after $f(x)$ is trained offline). Then, we have the normalized accumulative computational cost of the first i stages as $\mathbf{C}_i = \frac{1}{c(f)} \sum_{j=1}^{i} c(G_j)$ where $c(f)$ is the total cost of the whole scoring function, and thus $0 < \mathbf{C}_i < 1$ $(i < N)$ and $\mathbf{C}_N = 1$.

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

$$\mathcal{C}(\Pi_N) \approx \frac{1}{S} \sum_{x \in D} \mathbf{C}_{n(x)} = \sum_{i=1}^{N} \hat{p}(i; \Pi_N) \cdot \mathbf{C}_i \tag{4}$$

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

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

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

By substituting the three empirical probabilities into the global risk function in Eqn. (2), we obtain its intuitive form,

$$\begin{aligned}\mathcal{R}(\Pi_N; \Theta) &= \frac{1}{S}[\#\text{FN} \cdot C_{\text{FN}} + \#\text{FP} \cdot C_{\text{FP}} + \lambda \cdot \sum_{i=1}^{N} \mathbf{C}_i \cdot \#S_i] \\ &= \frac{1}{S} \sum_{i=1}^{N} [\#\text{FN}_i \cdot C_{\text{FN}} + \#\text{FP}_i \cdot C_{\text{FP}} + \lambda \cdot \mathbf{C}_i \cdot \#S_i] \end{aligned} \tag{5}$$

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

The optimal decision policy. Given parameters $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$, the optimal decision policy $\Pi_N^*(\Theta)$ is sought by minimizing the empirical global risk in Eqn. (5),

$$\Pi_N^*(\Theta) = \arg\min_{\Pi_N} \mathcal{R}(\Pi_N; \Theta) \tag{6}$$

In object detection, it is more convenient to specify reachable bounds on FPR (denoted by α) and FNR (denoted by β) based on the ROC curve of $f(x)$, rather than $(C_{\text{FP}}, C_{\text{FN}})$. We show that solving Eqn. (6) is equivalent to minimizing $\mathcal{C}(\Pi_N)$ subject to some given reachable accuracy bounds (α, β). Then, we have the constrained optimization problem,

$$\begin{aligned}\Pi_N^* = \arg\min_{\Pi_N} &\mathcal{C}(\Pi_N), \\ &\text{subject to } \hat{p}(\text{FP}; \Pi_N) \leq \alpha \text{ and } \hat{p}(\text{FN}; \Pi_N) \leq \beta. \end{aligned} \tag{7}$$

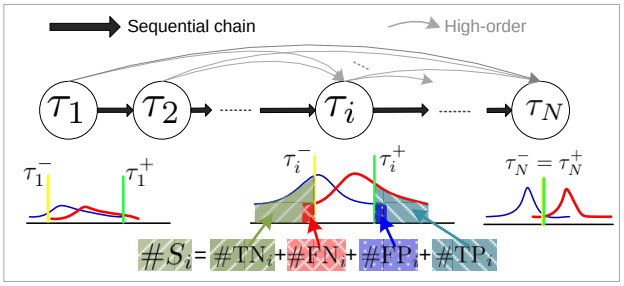

Figure 2. Graph interpretation of minimizing the empirical risk. The two-sided thresholds at a stage i generate the three numbers ($\#\text{FN}_i$, $\#\text{FP}_i$, $\#S_i$). All the thresholds are fully dependent in the sense that the positive and negative sub-populations observed at a stage i are affected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

Lemma 1. The solution Π_N^* of Eqn. (6) is also the solution to the constrained optimization problem in Eqn. (7) with $\alpha = \hat{p}(\text{FP}; \Pi_N^*)$ and $\beta = \hat{p}(\text{FN}; \Pi_N^*)$.

Proof. See the proof in the supplementary material. □

The equivalence can map one specification to another.

Proposition 1. Given (α, β), 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,

$$C_{\text{FP}}^{max} = \frac{\lambda \cdot S}{\alpha \cdot S^-} \quad \text{and} \quad C_{\text{FN}}^{max} = \frac{\lambda \cdot S}{\beta \cdot S^+}. \tag{8}$$

Proof. See the proof in the supplementary material. □

In learning, when $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$ is given, a fast DP algorithm is utilized to find $\Pi^*(\Theta)$ (in Sec. 5).

3.3. High-order connections between $\{\tau_i\}$'s in Π_N

Before presenting the method of minimizing the risk in Eqn. (6), we show that all two-sided thresholds τ_i's in Π_N are fully dependent.

In Fig.2, we consider τ_i at stage i. Based on Eqn. (5), 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 Fig.2), which are affected by all the previous stages j ($j < i$) as early thresholds τ_j's change the populations through their decisions. Thus, τ_i is affected by all previous τ_j's. By transition, τ_i affects the assignments of all the subsequent stages τ_j ($j > i$), so all $\{\tau_i\}$'s are fully dependent.

This underlying fully-dependent graph structure makes the problem of minimizing the risk in Eqn. (6) very hard in general. To address this issue, we proceed in two steps.

i) *Offline learning*. We first derive recursive formula for counting ($\#\text{FN}_i$, $\#\text{FP}_i$, $\#S_i$) along the stages. By using the derived recursions, the empirical risk in Eqn. (5) is divided into two parts: one is the sum of risks caused by individual stages (i.e., measured independently), and the other accounts for all the double-counting occurred by removing the sub-populations that have been decided in previous stages. The details are presented in Sec. 4.

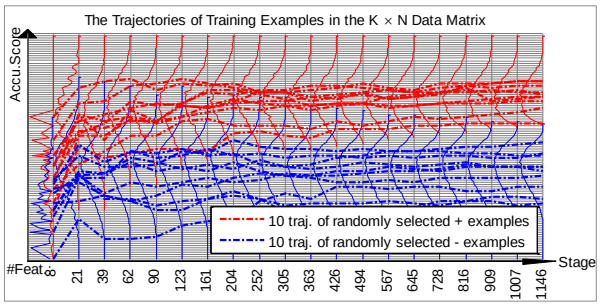

Figure 3. The $K \times N$ matrix created for recording the trajectories of positive (red dashed lines) and negative (blue ones) training examples (created for the human face AdaBoost classifier in Fig.1).

ii) *Online learning*. By relaxing the fully-dependent graph structure, i.e., removing all the high-order connections in Fig. 2 and only keeping the sequential chain connections, we derive an upper bound for the empirical risk which is then minimized by a DP algorithm efficiently. It is an upper bound because certain populations could be counted more than once in the empirical risk. We also show the tightness of the upper bound empirically (see Fig. 5).

We discuss the offline learning and the online learning in Sec. 4 and Sec. 5 respectively.

4. Offline learning: computing the risk

We first compute two histograms of the accumulative scores $f^i(x)$ at each stage i on D^+ and D^- respectively. At stage i, the set of accumulative scores is denoted by $A(i) = \{f^i(x); \forall x \in D\}$. 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 along stages (columns) in the matrix is characterized by, $\{bin(x, 1), \cdots, bin(x, N)\}$. Fig.3 shows 10 examples of +/- sample trajectories.

Statistical observation. *The trajectories of examples over the sequence of accumulative scores remain relatively steady*, as illustrated in Fig.3. This is true for different additive scoring functions studied in this paper. In other words, if τ_{i-1} classified x based on $f^{i-1}(x)$, its scores $f^i(x)$ in the next steps unlikely jump inside $[\tau_i^-, \tau_i^+)$, thus the policy will not regret its decisions.

Based on all the trajectories summarized in the $K \times N$ matrix, we further compute four types of statistics (see Fig.4) which lead to the recursions of counting ($\#\text{FN}_i$, $\#\text{FP}_i$, $\#S_i$) in the empirical global risk in Eqn. 5.

Notational Usages. 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 size of a corresponding set. Furthermore, we use the notation of a threshold itself

(such as τ_i^+ or τ_i^-) in the decision policy to denote its bin index in the $K \times N$ matrix without confusion. We use Fig.4 to illustrate our notations.

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,

$$D^+(k_i) = \{x : bin(x, i) = k, x \in D^+\}, \tag{9}$$

and its cardinality by $S^+(k_i) = |D^+(k_i)|$ (e.g., $D^+(5_1)$ in Fig.4 (a)). Note that here two entries in different columns may have the same positive example x based on the definition above. This leads to the double-counting which we have to take into account in the following definitions.

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

$$D^+(\underline{k_i}) = \cup_{r=1}^{k-1} D^+(r_i) \text{ and } D^+(\overline{k_i}) = \cup_{r=k}^{K} D^+(r_i). \tag{10}$$

For examples, see $D^+(\overline{7_5})$ and $D^+(\underline{6_5})$ in Fig.4 (b) and (c).

iii) *Rejection line and acceptance line based statistics.* By definition, we can write $\Pi_N = (\Pi_i, \tau_{i+1}, \cdots, \tau_N)$. Denote the entries below the rejection line of Π_i (front part in Π_N) by,

$$\underline{\Pi_i} = (\underline{\tau_1^-}, \cdots, \underline{\tau_i^-}), \tag{11}$$

and the entries above the acceptance line by,

$$\overline{\Pi_i} = (\overline{\tau_1^+}, \cdots, \overline{\tau_i^+}). \tag{12}$$

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 Π_3 respectively.

Then, we can count FNs and FPs generated by Π_i.

Definition 2: False negatives created by Π_i is defined by,

$$D^+(\underline{\Pi_i}) = \{x; \Pi_i \text{ rejects x}, x \in D^+\}. \tag{13}$$

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

$$D^+(\Pi_i) = D^+(\underline{\Pi_i}) \cup D^+(\overline{\Pi_i}). \tag{14}$$

Definition 3: False positives created by Π_i is defined by,

$$D^-(\overline{\Pi_i}) = \{x; \Pi_i \text{ accepts x}, x \in D^-\}. \tag{15}$$

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

iv) *Transition based statistics*. By considering sample transition from one sub-population as is defined above to another, we can count the double-counting of examples between them. Specifically, we define the two as follows.

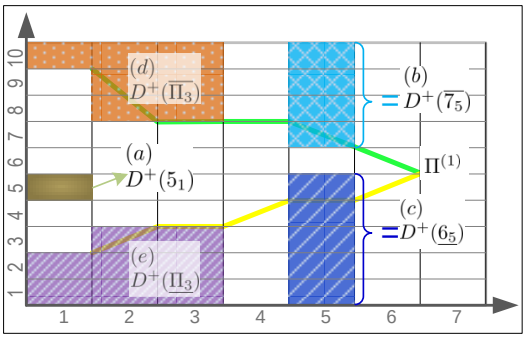

Figure 4. Illustrating the notations of the four types of statistics projected from trajectories of all training examples (only D^+ is used for clarity). Note that notations like $\underline{6_5}$ and $\overline{7_5}$ are used for this specific illustration.

Definition 4: False negatives double-counted at τ_i **w.r.t.** Π_{i-1} is the sub-population in D^+ that has been classified by Π_{i-1} already, and is rejected again by τ_i^-,

$$D^+(\underline{\tau_i^-}; \Pi_{i-1}) = D^+(\underline{\tau_i^-}) \cap D^+(\Pi_{i-1}). \tag{16}$$

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.

Definition 5: False positives double-counted at τ_i **w.r.t.** Π_{i-1} is the sub-population in D^- that has been classified by Π_{i-1} already, and is accepted again by τ_i^+,

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

Recursions. By using $\Pi_i = (\Pi_{i-1}, \tau_i)$, 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** Π_i is recursively defined by,

$$S^+(\underline{\Pi_i}) = S^+(\underline{\Pi_{i-1}}) + [S^+(\underline{\tau_i^-}) - S^+(\underline{\tau_i^-}; \Pi_{i-1})]$$
$$= \sum_{j=1}^{i} [S^+(\underline{\tau_j^-}) - S^+(\underline{\tau_j^-}; \Pi_{j-1})] \triangleq \sum_{j=1}^{i} \#\text{FN}_j \tag{17}$$

and **the number of FPs by** Π_i is calculated by,

$$S^-(\overline{\Pi_i}) = S^-(\overline{\Pi_{i-1}}) + [S^-(\overline{\tau_i^+}) - S^-(\overline{\tau_i^+}; \Pi_{i-1})]$$
$$= \sum_{j=1}^{i} [S^-(\overline{\tau_j^+}) - S^-(\overline{\tau_j^+}; \Pi_{j-1})] \triangleq \sum_{j=1}^{i} \#\text{FP}_j \tag{18}$$

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

$$S(\tau_i) \triangleq \#S_i = |D(\underline{\tau_i^-}) \cup D(\overline{\tau_i^+})| \tag{19}$$
$$= [S(\underline{\tau_i^-}) - S(\underline{\tau_i^-}; \Pi_{i-1})] + [S(\overline{\tau_i^+}) - S(\overline{\tau_i^+}; \Pi_{i-1})].$$

All the statistics are computed once at the offline stage, and can be used to compute the decision policy online for any given $\Theta = (C_{\text{FP}}, C_{\text{FN}}, \lambda)$.

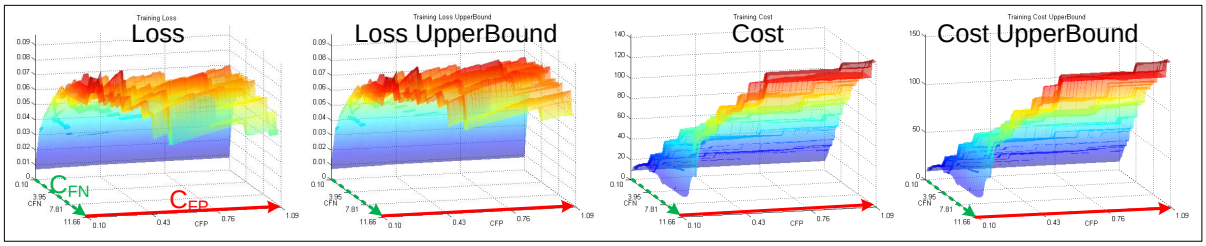

Figure 5. This figure shows the tightness of the upper bound (Eqn. (23)) for loss and computing cost empirically.

5. Online learning: minimizing the risk

By substituting Eqn. (17), Eqn. (18) and Eqn. (19) into Eqn. (5), it is divided into two parts: (i) the sum of risks caused by the assignment of an individual τ_i,

$$R(\tau_i;\Theta) = \frac{1}{S}\cdot[\lambda\cdot\mathbf{C}_i\cdot(S(\underline{\tau_i^-}) + S(\overline{\tau_i^+})) + S^+(\underline{\tau_i^-})\cdot C_{\text{FN}} + S^-(\overline{\tau_i^+})\cdot C_{\text{FP}}], \quad (20)$$

and (ii) the sum of risks caused by examples which are double-counted at stage i w.r.t Π_{i-1},

$$\mathbb{R}(\tau_i,\Pi_{i-1};\Theta) = \frac{1}{S}[\lambda\cdot\mathbf{C}_i\cdot(S(\underline{\tau_i^-};\Pi_{i-1}) + S(\overline{\tau_i^+};\Pi_{i-1})) + S^+(\underline{\tau_i^-};\Pi_{i-1})\cdot C_{\text{FN}} + S^-(\overline{\tau_i^+};\Pi_{i-1})\cdot C_{\text{FP}}]. \quad (21)$$

So, the empirical risk in Eqn. (5) is re-produced as,

$$\mathcal{R}(\Pi_N;\Theta) = \sum_{i=1}^{N}[R(\tau_i;\Theta) - \mathbb{R}(\tau_i,\Pi_{i-1};\Theta)]. \quad (22)$$

The high-order connections between τ_i's in Eqn. (21) make the problem of minimizing the risk very hard in general. Next, we derive an upper bound.

5.1. The upper bound of the risk

From Eqn. (22), we know that we can obtain an upper bound of the risk by deriving a lower bound for $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$ (due to the minus operator). 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 $\mathbb{R}(\tau_i,\Pi_{i-1};\Theta)$.

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 $\mathcal{R}(\Pi_N;\Theta)$,

$$\widehat{\mathcal{R}}(\Pi_N;\Theta) = \sum_{i=1}^{N} R(\tau_i;\Theta) - \sum_{i=2}^{N}\mathbb{R}(\tau_i,\tau_{i-1};\Theta), \quad (23)$$

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

$$\mathbb{R}(\tau_i,\tau_{i-1};\Theta) = \frac{1}{S}\cdot\{\lambda\cdot\mathbf{C}_i\cdot[(S(\underline{\tau_i^-};\underline{\tau_{i-1}^-}) + S(\underline{\tau_i^-};\overline{\tau_{i-1}^+}) + S(\overline{\tau_i^+};\underline{\tau_{i-1}^-}) + S(\overline{\tau_i^+};\overline{\tau_{i-1}^+}))] + [S^-(\overline{\tau_i^+};\underline{\tau_{i-1}^-}) + S^-(\overline{\tau_i^+};\overline{\tau_{i-1}^+})]\cdot C_{\text{FP}} + [S^+(\underline{\tau_i^-};\underline{\tau_{i-1}^-}) + S^+(\underline{\tau_i^-};\overline{\tau_{i-1}^+})]\cdot C_{\text{FN}}\}. \quad (24)$$

Proof. Details of proof are given in the supplementary material. The intuitive idea is that $\mathbb{R}(\tau_i,\tau_{i-1};\Theta)$ consider the risk caused by examples double-counted at τ_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. □

In Fig.5, 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 tightness is due to the statistical observation in Fig.3.

5.2. The DP algorithm

The upper bound $\widehat{\mathcal{R}}(\Pi_N;\Theta)$ can be minimized by a DP algorithm efficiently. Let $B_i[\tau_i]$ be the risk of the best assignment to the i first stages with the i-th one is τ_i. Starting from $B_1[\tau_1] = R(\tau_1;\Theta)$, we have for $i = 2,\cdots,N$,

$$B_i[\tau_i] = R(\tau_i;\Theta) + \min_{\tau_{i-1}}(B_{i-1}[\tau_{i-1}] - \mathbb{R}(\tau_i,\tau_{i-1};\Theta)).$$

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 τ_{i-1} as a function of τ_i for later back-tracing starting at $i = 2$,

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

(ii) *The backward step* for finding the near-optimal decision policy $\Pi_N^* = (\tau_1^*,\cdots,\tau_N^*)$, where we first take, $\tau_N^* = \arg\min_{\tau_N} B_N[\tau_N]$ and then trace back $\tau_i^* = \mathbb{T}_{i+1}[\tau_{i+1}^*]$ in the order of decreasing $i = N-1,\cdots,1$.

To run the DP algorithm more efficiently for different Θ, we create six DP tables for computing $B_i[\tau_i]$ quickly (details will be given in the supplementary material).

6. Experiments

In the experiments, we learn decision policies for AdaBoost, SVM and DPM, and compare with their corresponding popular cascade methods [3, 7, 26].

Settings: In all the comparison experiments, we learned the decision policies by specifying (α,β), and the corresponding $(C_{\text{FP}}, C_{\text{FN}})$ are searched. From Eqn.8, we know that only the ratios $\frac{C_{\text{FP}}}{\lambda}$ and $\frac{C_{\text{FN}}}{\lambda}$ matter, so we set $\lambda = 1.0$. Furthermore, we show the normalized values of searched C_{FP} and C_{FN} by S in all the figures and tables.

					Classification			Detection	
Method	#Feat	#Stages	C_{FP}	C_{FN}	FPR	FNR	CostPerEx	AP	CostPerWin
AdaBoost	1146	1	/	/	0.0017972	0.3030	1146	0.815	1146
AdaBoost Csc [26]	2497	20	/	/	0.0046214	0.3284	297.84	0.807	37.279
AdaBoost SoftCsc [3]	2135	20	/	/	0.0029214	0.3369	227.84	0.805	36.109
AdaBoost Policy	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 (Π) 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 performance. *The computational efficiency* is measured by average #Feat tested per example for classification, and average #Feat tested per sliding window for detection. *The accuracy performance* 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, 0.807, 0.805 and 0.809 respectively, but our decision policy saves about 10 haar feature evaluations per sliding window on average.

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 [26] and the soft cascade method [3] which are the two of the most popular cascade methods for face detection.

The training and testing data. *The training dataset* consist of 11020 positive human face images (with the size being 20×20 pixels) and 15366 background images which do not contain human faces. The background images are not cropped image patch but the whole images having different sizes. *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 clutters. In addition, we use the CMU+MIT human face detection benchmark [19] 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 [26], and adopt the decision stump as the weak classifier. In the experiments, we use the integral image to compute the Haar features as don in [26]. 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.

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. 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).

The learned decision policy. We first train a single strong AdaBoost classifier with 1146 boosted weak classifiers. To learn the decision policy, we first divide the 1146 boosted weak classifier into 20 subsets (as is shown in the bottom of Fig. 1). Note that the configuration of the decision policy 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 accuracy performance.

6.2. Decision policies for SVM classifiers and DPMs

We learn decision policies for linear SVM classifier and DPM trained on INRIA person dataset [6]. We compare with the probably approximate admissible (PAA) threshold [7], which is used in the cascade of DPM and selects the rejection thresholds based on $\tau_i^- = \min_{\{x; f(x) \geq t_1, x \in D^+\}} f^i(x)$, where t_1 is predefined to focus on high-scoring positive examples.

SVM classifier. We train the linear SVM classifier using the modified 32-dimensional HOG features [8] with the template size being 15×5 cells (each cell is of 8×8 pixels) using the publicly available code [11], and we have $15 \times 5 \times 32$ weight parameters and 1 bias term in the SVM classifier. Similar to [7], we also specify a simplified classifier by projecting the trained SVM classifier into the top 5 principal components pooled from the thole training dataset, resulting in $15 \times 5 \times 5$ weight parameters. So, we have 150 cells in total which are reordered based on the statistical power/computational cost ratio [2], 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 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.

DPM. We learned DPM on INRIA person using the publicly released code in [11]. The root is of 15×5 cells and the 8 parts are of the same size 6×6 cells which are placed at the twice resolution w.r.t the root. Unlike the order used in [7], we order the root, the parts as well their deformations based on the the statistical power/computational cost ratio [2]. So we have $1 + 8 \times 2 = 17$ steps. Together with the simplified PCA filters as done in [7], we have 34 stages and 726 cells in total.

The results are summarized in Table. 2. Our decision policy outperforms the PAA method in speed in both clas-

					Classification			Detection	
Method	#Cell	#Stages	C_{FP}	C_{FN}	FPR	FNR	CostPerEx	AP	Avg.Cost
SVM	150	1	/	/	0.0185	0.0052	150	0.80	150
SVM PAA	150	20	/	/	0.0134	0.0190	100.7	0.79	75.8
SVM policy Π	150	20	0.00031	0.01852	0.0180	0.0069	**29.9**	0.788	**31.3**
DPM [8]	726	1	/	/	0.0074	0.0091	726	0.887	726
DPM PAA [7]	726	34	/	/	0.0096	0.0072	230.84	0.885	183.36
DPM Policy Π	726	34	0.000874	0.071429	0.0076	0.0071	**105.39**	0.879	**95.13**

Table 2. Comparison between PAA [7] and our policy Π of the SVM classifier and DPM [8] trained and tested on INRIA person dataset [6].

sification and detection with similar accuracy performance. Our policy can speed up due to (i) we reorder the cells/filters in terms of the ratio between the statistical power and the computational cost, while the PAA method only take into account the variance of the positive scores (as done in the latest release code [11]); 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.

7. Conclusion

This paper presents a framework of learning cost-sensitive decision policy for popular object detectors with additive scoring functions (such as AdaBoost, SVM and DPM). The decision policy explicitly accounts for the loss of misclassification and the cost of computation. The learning is formulated under the risk minimization framework and the upper bound of the risk function is solved by an efficient DP algorithm. By comparing with the state-of-art cascade methods, our decision policy outperforms them in speed significantly with similar accuracy in experiments.

Acknowledgments: This work is supported by DARPA MSEE grant FA 8650-11-1-7149, MURI grant ONR N00014-10-1-0933, and NSF IIS1018751.

References

[1] Y. Amit, D. Geman, and X. D. Fan. A coarse-to-fine strategy for multiclass shape detection. *PAMI*, 26(12):1606–1621, 2004.

[2] G. Blanchard and D. Geman. Hierarchical testing designs for pattern recognition. *Ann. Statist.*, 33(3):1155–1202, 2005.

[3] L. D. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2005.

[4] 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*, 77(1-3):65–86, 2008.

[5] M. Chen, Z. E. Xu, K. Q. Weinberger, O. Chapelle, and D. Kedem. Classifier cascade for minimizing feature evaluation cost. *JMLR - Proceedings Track*, 22:218–226, 2012.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9):1627 – 1645, 2010.

[9] S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *CVPR*, 2006.

[10] T. Gao and D. Koller. Active classification based on value of classifier. In *NIPS*, 2011.

[11] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5, 2012.

[12] A. Grubb and J. A. D. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *AISTATS*, 2012.

[13] I. Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *NIPS*, 2011.

[14] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 31(12):2129–2142, 2009.

[15] H. Masnadi-Shirazi and N. Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive svms. In *ICML*, 2010.

[16] H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *PAMI*, 33(2):294–309, 2011.

[17] O. Pele and M. Werman. Robust real-time pattern matching using bayesian sequential hypothesis testing. *PAMI*, 30(8):1427–1443, 2008.

[18] B. Póczos, Y. Abbasi-Yadkori, C. Szepesvári, R. Greiner, and N. Sturtevant. Learning when to stop thinking and do something! In *ICML*, 2009.

[19] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *PAMI*, 20(1):23–38, 1998.

[20] M. Saberian and N. Vasconcelos. Learning optimal embedded cascades. *PAMI*, 34(10):2005–2018, 2012.

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

[22] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *CVPR*, 2004.

[23] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *CVPR*, 2005.

[24] X. Song, T. Wu, Y. Jia, and S.-C. Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, 2013.

[25] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[26] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[27] A. Wald. *Sequential Analysis*. Wiley, New York, 1947.

[28] R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *ICCV*, 2007.