

Building a Cascade Detector and Applications in Automatic Target Detection

Xiaoming Huo and Jihong Chen

July 10, 2003

Abstract

A hierarchical classifier (cascade) is proposed for target detections. In building an optimal cascade, three heuristics are considered: (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data with various underlying distributions are carried out. It is found that weighting heuristic is optimal in both computational complexity and error rates. The contribution of this paper is to initiate a systematic comparison of several potential heuristics that can be utilized in building a hierarchical model. A range of discussions – regarding the implications and the promises of the cascade architecture, as well as techniques that can be integrated into this framework – will be provided.

The current winner – weighting algorithms – is applied to an IR data set. It is found that it outperforms some state-of-the-art approaches that utilize the same type of simple classifiers.

1 Introduction

Recently, we have seen a boosting of hierarchical-structure-based methodologies. The following list provides a few examples.

- In statistics, there are CART and MARS [2] and many other follow-up works.
- In machine learning, there are C4.5 [13], perception trees [1], and a lot more...
- In signal processing and harmonic analysis, we have seen quad-tree based image coding [14, 16], pyramid [3], and so on.

A commonality of the above methodologies is that they all rely on a hierarchical structure.

In this paper, a special type of hierarchical structure is studied. We limit ourselves to a binary classification problem: the response is either 0 or 1. We consider a hierarchy of classifiers such that at each level (in the hierarchy), there are exactly two branches, and at least one of them is terminal: it has no subsequential classifier. For a historic reason, this structure is called a *cascade*.

Cascades are the fundamental structures of many successful techniques in various applications. Some recent works include the Maximum Rejection Classifier attributed to M. Elad et al. [5], a similar method in pattern recognition by H. Hel-Or and Y. Hel-Or [12, 11], a successful framework in computer vision by Viola and Jones [18], and an application in automatic target recognition by a group in M.I.T. [10].

There are many advantages in using a cascade. A few of them will be listed in the following.

1. *Interpretability*. Such a framework can generate a model that is easy to interpret.

2. *Computational efficiency.* In general, it is easy to design a fast algorithm in implementations. The computational complexity of a derived algorithm will be small.
3. *Generality.* It will be easy to incorporate other features, for example, multi-scale features, such as wavelets, curvelets, beamlets, and edgelets.
4. *Flexibility.* The designing of the algorithm is flexible: we can optimize the computing procedure based on prior knowledge of the data.

This paper is about numerical strategies in searching for an optimal cascade. To study the trade-off between two types of errors (false alarm and mis-detection), a formulation is provided. The basic idea is to derive algorithms that can approximate the frontier of the feasible region of pairs of two types of error rates. This is in spirit similar to the idea of using a Receiver Operating Characteristics (ROC) curve. Three heuristics are presented. They are (1) a propagating method, (2) another approximation method by controlling the error rates, and (3) a method based on minimizing weighted error rates. Simulations are design to empirically examine the properties of each of these methods. It is found that the weighting strategy is optimal in terms of both the error rates and the speed. This finding implies that in designing algorithms for cascade, weighting is more favorable.

We experiment our current cascade to an IR data set. The results do not outperform another existing method: support vector machine (SVM). However, we only allow a very simple type of classifier in our hierarchy; while SVM utilizes nonlinear classifiers. We discuss the limitation of our current framework, and point out the direction for future improvement. Our approach is systematic, generic, and flexible. It has strong promises to be developed into a powerful method. The current suboptimal performance in a particular data set indicates the necessity of such an

improvement.

The rest of the paper is organized as follows. Section 2 describes the general architecture of a cascade model. Section 3 describes three types of heuristics, and their related implementation strategies. Section 4 describes the experiments and results. Section 5 gives some discussions. Some future research topics are raised. Some concluding marks are provided in Section 6.

2 Architecture of a Cascade

We consider a binary classification problem. A cascade classifier has the following structure:

$$\begin{array}{c}
 \text{Input Data } \mathbf{x} \\
 \Downarrow \\
 \boxed{f_1(\mathbf{x}) + b_1} \quad \stackrel{\leq 0}{\implies} \mathcal{C}_1 \\
 \Downarrow \geq 0 \\
 \boxed{f_2(\mathbf{x}) + b_2} \quad \stackrel{\leq 0}{\implies} \mathcal{C}_1 \\
 \Downarrow \geq 0 \\
 \boxed{f_3(\mathbf{x}) + b_3} \quad \stackrel{\leq 0}{\implies} \mathcal{C}_1 \\
 \Downarrow \geq 0 \\
 \mathcal{C}_2
 \end{array} \tag{1}$$

Here \mathcal{C}_1 stands for the first class, and \mathcal{C}_2 stands for the second class. In our experiment, “ \mathcal{C}_1 ” is the clutter class (labelled by “0”s) and “ \mathcal{C}_2 ” is the target class (labelled by “1”s).

Each intermediate node is made by a simple classifier:

$$f_j(\mathbf{x}) + b_j \leq 0,$$

where j is the index of the intermediate node. The previous diagram gives a cascade classifier with three intermediate nodes. Each $f_j(\mathbf{x})$ may have the form:

1. single entry of \mathbf{x} :

$$f_j(\mathbf{x}) = \mathbf{x}_j,$$

where \mathbf{x}_j is the j -th coordinate of vector \mathbf{x} ;

2. linear:

$$f_j(\mathbf{x}) = \mathbf{c}^T \mathbf{x};$$

3. quadratic:

$$f_j(\mathbf{x}) = \mathbf{c}^T \mathbf{x} + \mathbf{x}^T M \mathbf{x}; \quad \text{or}$$

4. support vector machine (SVM) with kernels which are either higher than degree-2 polynomials, or radial basis functions

$$f_j(\mathbf{x}) = \sum_{k=1}^N \alpha_k K(\mathbf{x}, \mathbf{x}_k),$$

where $K(\cdot, \cdot)$ is a kernel function, \mathbf{x}_k 's are the N observations and α_k 's are the weights that are trained by applying a SVM.

In our simple classifier, an observation is predicted to be in class \mathcal{C}_2 if and only if

$$f_j(\mathbf{x}) + b_j \geq 0, \text{ for all } j;$$

otherwise the prediction is \mathcal{C}_1 . We choose b_j small enough such that the chance that an observation from \mathcal{C}_2 is misclassified as an observation from \mathcal{C}_1 is small. When a cascade is built, the thresholds b_j 's will remain unchanged afterwards.

Note that at the early stage of a cascade algorithm, we would like to choose a simple kernel. In later stages of a cascade algorithm, we tend to choose more complex kernels, to explore more difficult structures.

In building a cascade model, in the second node, we exclude all the training data that are predicted by the first classifier as in \mathcal{C}_1 . In other words, the second simple classifier is built for the “remaining” observations that are classified as in \mathcal{C}_2 by the first classifier. In general, the consequent classifier is built for the data that are classified as \mathcal{C}_2 by the previous classifier. Apparently the number of observations decreases, therefore the training of classifiers becomes easier.

For simple decision rules, we can apply Linear Discriminant Analysis (LDA). When a more complex model is needed, we can use Quadratic Discriminant Analysis (QDA). A description on how to find a decision rule via LDA and QDA can be found in [9]. When a more complex decision rule is needed, support vector machines can be applied, for example, with a radial basis function as kernel. More details on how to train support vector machines can be found in [4].

2.1 A Geometric Tour on How (and Why) a Cascade Algorithm Works

We imagine the following simple procedure. A cascade starts with simple linear detectors (function f_i 's). If targets are contained in a subregion of the space that is made by all image chips, a sequence of linear detection rules will outline a convex hull of the subregion that include (hopefully) most of the targets. See Figure 2 for an illustration.

There is a point in which no linear classifier will be effective in distinguishing a significant proportion of clutters from targets. However, it can still be true that by utilizing a more complex rule, e.g., quadratic or radial basis SVMs, it is possible to distinguish a significant amount of clutters from targets. A cascade approach shall automatically switch to a more complex decision rule when it is necessary. Figure 1 shows that when the target region has holes, support vector machines with radial basis functions can render an efficient classifier, while both LDA and QDA will fail. Note that since many clutters have been “rejected” in the previous stages, the sample size at the later stage is small. Hence support vector machines can be trained efficiently.

3 The Optimal Strategies in Building a Cascade

We consider what the optimal strategies are in building a cascade classifier. We limit our analysis on binary classification problems. The general description of a cascade is given. The optimality of a cascade is defined in an empirical sense. Finally, we propose some numerical experiments to compare different strategies.

Section 3.1 formulates the problem. It provides the necessary notations to analyze the model. Section 3.2 describes three types of heuristics, and strategies of implementations under various circumstances.

3.1 Cascade Classifier

Consider the diagram (1). For a fixed natural number L , a cascade with L levels can be written as

$$y = \begin{cases} 1, & \text{if } f_1(\mathbf{x}) \geq t_1, \quad f_2(\mathbf{x}) \geq t_2, \quad f_3(\mathbf{x}) \geq t_3, \dots, \quad f_L(\mathbf{x}) \geq t_L; \\ 0, & \text{otherwise,} \end{cases}$$

where symbol \mathbf{x} denotes the input, and symbol y denotes the response, which is binary (0 or 1). The functions $f_i(\cdot), i = 1, 2, \dots, L$, are relatively simple, e.g. linear functions, univariate functions, etc. The quantities $t_i, i = 1, 2, \dots, L$, are constants. For example in CART [2], a simple classifier has the form:

$$\mathbf{x}_j \geq c, \tag{2}$$

where \mathbf{x}_j is the j th component of a random vector \mathbf{x} , and c is a constant. Let $f_1(\mathbf{x}) = \mathbf{x}_j$ and $t_1 = c$, then the classifier $f_1(\mathbf{x}) \geq t_1$ is equivalent to the classifier in (2).

To evaluate the performance of a given cascade, we consider an empirical approach.

Let N denote the total number of observations. Let $N^i, i = 0, 1$ denote the numbers of observations that belong to classes ‘0’ and ‘1’ respectively. Let N_1 denote the number of observations that are classified as in class ‘0’ by decision rule “ $f_1(x) < t_1$ ”. Let N_2 denote the number of remaining observations. In general, among the N_{2i-2} observations that are classified as ‘1’ in the $(i - 1)$ th step, let N_{2i-1} denote the number of observations that are classified as class ‘0’ by decision rule “ $f_i(x) < t_i$ ”, $1 \leq i \leq L$; and let N_{2i} denote the number of remaining observations.

Due to the hierarchical structure of a cascade, we must have

$$N = N_1 + N_2,$$

$$N_{2i-2} = N_{2i-1} + N_{2i}, \quad i = 2, 3, \dots, L, \text{ and}$$

$$N_{2i-2}^j = N_{2i-1}^j + N_{2i}^j, \quad i = 2, 3, \dots, L, j = 0, 1.$$

Among the N_j , $1 \leq j \leq 2L$, observations, let N_j^i , $i = 0, 1$, denote the number of observations that belong to class '0' and '1' respectively. The number of 1's which are misclassified as 0's is

$$\sum_{i=1}^L N_{2i-1}^1.$$

At the same time, the number of 0's which are misclassified as 1's is

$$N_{2L}^0.$$

The above quantities are embedded in a cascade according to the following diagram:

$$\begin{array}{ccc}
N = N^0 + N^1 & & \\
\downarrow \searrow & & \\
N_2^0 + N_2^1 = N_2, & N_1 = N_1^0 + N_1^1 & \\
\downarrow \searrow & & \\
N_4^0 + N_4^1 = N_4, & N_3 = N_3^0 + N_3^1 & \\
\downarrow \searrow & & \\
N_6^0 + N_6^1 = N_6, & N_5 = N_5^0 + N_5^1 & \\
\downarrow \searrow & & \\
& \vdots & \\
\downarrow \searrow & & \\
N_{2L}^0 + N_{2L}^1 = N_{2L}, & N_{2L-1} = N_{2L-1}^0 + N_{2L-1}^1 & \\
\downarrow & & \\
& \text{'1'} &
\end{array}
\left. \vphantom{\begin{array}{ccc} N = N^0 + N^1 \\ \downarrow \searrow \\ N_2^0 + N_2^1 = N_2, & N_1 = N_1^0 + N_1^1 \\ \downarrow \searrow \\ N_4^0 + N_4^1 = N_4, & N_3 = N_3^0 + N_3^1 \\ \downarrow \searrow \\ N_6^0 + N_6^1 = N_6, & N_5 = N_5^0 + N_5^1 \\ \downarrow \searrow \\ & \vdots \\ \downarrow \searrow \\ N_{2L}^0 + N_{2L}^1 = N_{2L}, & N_{2L-1} = N_{2L-1}^0 + N_{2L-1}^1 \end{array}} \right\} \rightarrow \text{'0'} \quad (3)$$

A hierarchical classifier that has the above architecture is called a *level- L cascade*. Recall that each simple classifier is a binary classifier. Let $\mathcal{H}(L)$ denote all the level- L cascade. For a given level- L classifier h (i.e., $h \in \mathcal{H}(L)$), let $E_{0 \rightarrow 1}(h)$ denote the number of 0's that are wrongly classified by h as 1's; and let $E_{1 \rightarrow 0}(h)$ denote the number of 1's that are wrongly classified as 0's.

If h produces the results in the diagram (3), we have

$$E_{1 \rightarrow 0}(h) = \sum_{i=1}^L N_{2i-1}^1, \quad E_{0 \rightarrow 1}(h) = N_{2L}^0.$$

We define the following function:

$$f(k; L) = \min_{h \in \mathcal{H}(L)} E_{0 \rightarrow 1}(h),$$

subject to $E_{1 \rightarrow 0}(h) \leq k,$

where k is a positive integer. A cascade h is optimal if and only if the following condition is satisfied:

$$E_{0 \rightarrow 1}(h) = f[E_{1 \rightarrow 0}(h); L].$$

Or in other words, the optimal cascade should reside on the *frontier* of the feasible region. An illustration is shown in Figure 3.

We consider strategies that can compute an optimal cascade. There are three classes of heuristics:

1. Frontier-following (in Section 3.2.1),
2. Controlled error rates (in Section 3.2.2), and
3. Weighting (in Section 3.2.3).

Before describing the above heuristics in details, we would like to make the following general comments:

1. We conjecture that finding the optimal cascade in a generic situation is a hard problem, especially when the number of levels L is large.
2. The current formulation does not address the generalization error rate. To study the generalization error, we may take advantage of the knowledge in *structured risk minimization* [17].

3. Due to the difficulty of computing the optimal cascade, we hope to utilize some greedy algorithm to find a suboptimal cascade. The heuristics, which will be described later, are ways to explore different possibilities in this problem.
4. The *frontier* gives a set of cascade classifiers. The particular choice of a cascade depends on the cost of different types of errors. It is problem dependent, and we choose not to pursue in this direction.

3.2 Three Heuristics

We describe three types of heuristics. Section 3.2.1 describes strategies which follow the frontier of the feasible region. Section 3.2.2 describe a heuristic that is based on controlled error rates. Section 3.2.3 describes a strategy that is based on the weighted misclassification rate.

3.2.1 Frontier Following

Optimal Cascade. Let $h^*(k, L)$ denote the optimal cascade classifier with L levels and k observations which are wrongly classified as 0's. We have

1. $E_{1 \rightarrow 0}([h^*(k, L)]) = k$, and
2. $E_{0 \rightarrow 1}[h^*(k, L)] = f(k; L)$.

A single cascade classifier, $h^*(k, L)$, corresponds to a point on the frontier.

Computing $h^*(k, 1)$. When the level of a cascade classifier is 1, in many cases, there are fast algorithms to compute the classifier $h^*(k, 1)$, $k = 0, 1, 2, \dots, N^1$. As an example, if the classifier are those that are used in CART:

$$\mathbf{x}_j \geq a,$$

For each variate \mathbf{x}_j , under the condition $E_{1 \rightarrow 0} \leq k$, one can find the corresponding maximal error rate for $E_{0 \rightarrow 1}$. Note that the above problem is for a univariate distribution. The $h^*(k, 1)$ is obtained by taking the maximum for all \mathbf{x}_j 's.

Propagating. We develop a propagation algorithm to *approximate* the frontier. We start with some notations.

For a cascade, let $R^1(h)$ denote the region in which the response is predicted to be 1:

$$R^1(h) = \{x : h(x) = 1\}.$$

Let H_{A, k_1} denote all the simple classifiers in region A , whose error rate $E_{1 \rightarrow 0}$ are no larger than k_1 . Apparently, if a classifier $h \in H_{A, k_1}$, we have $E_{1 \rightarrow 0}(h) \leq k_1$. Note that the domain of the classifier h is the region A . Let $r^*(A, k_1)$ denote the classifier which is in the set H_{A, k_1} and takes the minimum of $E_{0 \rightarrow 1}$:

$$r^*(A, k_1) = \arg \min_{h \in H_{A, k_1}} E_{0 \rightarrow 1}(h).$$

We may use the following heuristic to approximate the frontier. Note that we can not guarantee that the exact frontier will be found:

Algorithm Prop.

1. Let $h'(k, 1) = h^*(k, 1)$, $k = 0, 1, 2, \dots, N^1$.

2. **For** $\ell \geq 1$,

let $h'(k, \ell + 1)$, $k = 0, 1, 2, \dots$ denote the solution to the following optimization problem:

$$\min_{k_1 \leq k} E_{0 \rightarrow 1}[h'(k_1, \ell)] + E_{0 \rightarrow 1}(r^*\{R^1[h'(k_1, \ell)], k - k_1\}), \quad (4)$$

End.

3. Use $h'(k, L)$ to approximate $h^*(k, L), 1 \leq k \leq N^1$.

Note that the number N^1 in the first row of the above algorithm can be reduced to a small value, given that the error rate, $E_{1 \rightarrow 0}$, can be controlled significantly smaller than the quantity N^1 . By doing so, a large amount of computation can be saved.

An Implementational Strategy. We describe a numerically appealing alternative to implement the algorithm in Section 3.2.1. Let K denote a fixed integer. We consider how to compute $h'(k, \ell+1), 0 \leq k \leq K$, while $h'(k, \ell)$'s are given. The main idea can be illustrated by the following table:

	0	1	2	\dots	$K-1$	K
	$h'(0, \ell)$	$h'(1, \ell)$	$h'(2, \ell)$	\dots	$h'(K-1, \ell)$	$h'(K, \ell)$
	A_0	A_1	A_2	\dots	A_{K-1}	A_K
0	$r^*(A_0, 0)$	$r^*(A_1, 0)$	$r^*(A_2, 0)$	\dots	$r^*(A_{K-1}, 0)$	$r^*(A_K, 0)$
1	$r^*(A_0, 1)$	$r^*(A_1, 1)$	$r^*(A_2, 1)$	\dots	$r^*(A_{K-1}, 1)$	
\vdots	\vdots	\vdots	\vdots			
$K-2$	$r^*(A_0, K-2)$	$r^*(A_1, K-2)$	$r^*(A_2, K-2)$			
$K-1$	$r^*(A_0, K-1)$	$r^*(A_1, K-1)$				
K	$r^*(A_0, K)$					

In the above table, we have

$$A_i = R^1[h'(i, \ell)], \quad i = 0, 1, \dots, K.$$

Following a similar description as in Section 3.2.1, for entries in one column of the above table,

$$r^*(A_i, k), k = 0, 1, \dots, K-i,$$

there are efficient algorithms to compute. The solutions to the problem in (4) can be quickly extracted by comparing the values of the objective function in (4) at entries that are in a line which is parallel to the second diagonal, i.e. entries $(0, i), (1, i - 1), \dots, (i - 1, 1), (i, 0)$. In implementations, this approach should give an efficient algorithm.

One question is how to integrate the above algorithm with specific types of classifiers, e.g. LDA or QDA. We provide a few suggestions. In the LDA, the discriminant direction can be computed first. Then the threshold is varied to achieve different error rates. In the QDA, similar approach can be taken: finding the quadratic form first, then the threshold is varied to obtain different error rates.

The complexity of a propagating algorithm can be relatively high, especially when the error rate $E_{1 \rightarrow 0}$ is large. Suppose the simple classifiers are based on single variate functions, as in (2). For a fixed coordinate, the computational complexity of finding the optimal classifier should be at least the order of complexity to sort the N coordinates, which is $O(N \log(N))$. Let d denote the dimensionality of the data: there are d variates. The propagating algorithm searches in table in (5), which has $O(K^2)$ entries. Overall, the computational complexity is $O(K^2 \cdot d \cdot N \log(N))$.

3.2.2 Controlled Error Rate

Sometimes we may want to quickly compute a sub-optimal cascade. The following approach can be adapted.

Algorithm **CER**.

1. Let \tilde{h}_0 be a classifier, which classifies everything to class ‘1’:

$$\tilde{h}_0(x) = 1, \forall x.$$

2. Set $k = 0$, $\ell = 0$.

3. **While** the last improvement is greater than 0, (in the first time, this condition is always true,)

Find $r^*[R^1(\tilde{h}_\ell), 0]$;

Let $\tilde{h}_{\ell+1}$ be the combination of \tilde{h}_ℓ and $r^*[R^1(\tilde{h}_\ell), 0]$;

Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$.

Record the last improvement as $E_{0 \rightarrow 1}(\tilde{h}_{\ell+1}) - E_{0 \rightarrow 1}(\tilde{h}_\ell)$;

End.

4. Set $h'(k) = \tilde{h}_\ell$.

5. Set $k \leftarrow k + 1$.

Find $r^*[R^1(\tilde{h}_\ell), 1]$;

Let $\tilde{h}_{\ell+1}$ be the combination of \tilde{h}_ℓ and $r^*[R^1(\tilde{h}_\ell), 1]$;

Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$.

6. If $k = K$, where K is a predetermined maximal allowable error rate, $E_{1 \rightarrow 0}$, then stop;

Otherwise, go back to the step 4.

7. The function $h'(k), k = 0, 1, 2, \dots, K$, give us an approximation to the frontier.

Let $f(k)$ denote the lower bound of all cascades (the number of levels can be arbitrarily large):

$$f(k) = \min_L f(k; L).$$

Let $h^*(k)$ denote the classifier which corresponds to $f(k)$, or in other words,

$$h^*(k) = \operatorname{argmin}_{h^*(k,L), L=1,2,3,\dots} E_{0 \rightarrow 1}[h^*(k, L)].$$

The intuition of designing the above procedure is that hopefully, the final \tilde{h}_ℓ is close to the optimal classifier, $h^*(k)$.

We expect this approach to be very computationally efficient. This algorithm is analogous to the ideas in the MRC.

3.2.3 Weighting

We propose an alternative to the method *controlled error rate*. This method is also motivated by using Lagrangian multipliers to locate the frontier of a feasible region.

Let λ be a potentially large positive constant. By following a similar argument in Section 3.2.1, the following optimization problem can be solved efficiently.

$$s^*(A, \lambda) = \operatorname{argmin}_{h \in \mathcal{H}(A)} E_{0 \rightarrow 1}(h) + \lambda E_{1 \rightarrow 0}(h), \quad (6)$$

where A denotes a region, the set $\mathcal{H}(A)$ includes all simple classifiers residing in the region A , and λ is a positive constant.

The starting value of λ is typically large. One can gradually reduce the value of λ to produce a classifier that minimizes both $E_{0 \rightarrow 1}$ and $E_{1 \rightarrow 0}$ simultaneously.

We propose the following algorithm.

Algorithm **Weighting**.

1. Let \tilde{h}_0 be a classifier, which classifies everything to class ‘1’:

$$\tilde{h}_0(x) = 1, \forall x.$$

2. Set $\ell = 0$.

3. Choose a large initial value (denoted by λ_0) for λ : $\lambda = \lambda_0$.
4. **While** the last improvement in the objective function in (6) is ‘significantly nonzero’,

find $s^*[R^1(\tilde{h}_\ell), \lambda]$;

Let $\tilde{h}_{\ell+1}$ be the combination of \tilde{h}_ℓ and $s^*[R^1(\tilde{h}_\ell), \lambda]$;

Update: $\ell \leftarrow \ell + 1$ and $\tilde{h}_\ell \leftarrow \tilde{h}_{\ell+1}$;

Record the decreasing of the objective function that is in (6).

The above is the “last improvement”.

End.

5. Let $k = E_{1 \rightarrow 0}(\tilde{h}_\ell)$, $f(k) = E_{0 \rightarrow 1}(\tilde{h}_\ell)$.
6. Let λ take a smaller value: $\lambda \leftarrow \lambda/\theta$, e.g. $\theta = 2$.
7. If $f(k) < 1$, then go back to the step 4; Otherwise, continue.
8. Output $[k, f(k)]$ to approximate the frontier.

Note that by controlling the final value of parameter λ , we can realize different trade-off between the two error rates: $E_{0 \rightarrow 1}$ and $E_{1 \rightarrow 0}$.

4 Experiments

In this section, our objective is to assess the performance of the above three heuristic algorithms. Experiments were carried out on two synthetic data sets and an infra-red (IR) image data. For simplicity, we stayed with univariate functions employed in CART. In the results, the approximate frontier f -curves reflected the empirical performance on training data. And the generalization

performance on testing data was evaluated by the complementary ROC (c-ROC) curves, with x -axis representing the missed detection rate and y -axis representing the false alarm rate. Each point on an f curve corresponded to a cascade, which was used to generate a point on the c-ROC curve.

All experiments were conducted in MATLAB codes.

4.1 Synthetic Data

In the following two examples, we simulated two data sets with different underlying distributions and compared the performance of three heuristic algorithms. For the method *propagating*, the levels of cascades were assigned as $L = 2n$, where n is the number of dimensions. For *weighting*, We set $\lambda = 40$ and $\theta = \sqrt{2}$.

Example 1: $X \in \mathbb{R}^2$. The probability of $y = 1$ is 0.3, and $y = 0$ is 0.7. If $y = 1$, X are drawn from $N(\mathbf{0}, I)$, otherwise X are drawn from $N(\mathbf{0}, 10I)$. The theoretical boundary is

$$x_1^2 + x_2^2 = \frac{20}{9} \log\left(\frac{30}{7}\right).$$

An explanation on the above decision rule is postponed to Appendix A.

Example 2: $X \in [0, 1]^{10}$. The two classes have equal prior probability. Define the region $A = \{X : 0.3 \leq X_1 \leq 0.7, 0.4 \leq X_2 \leq 0.6, 0.2 \leq X_3 \leq 0.4\}$, and denote $|A|$ the area of region A . The conditional density of X in class i ($i = 0, 1$) is as follows

$$f_i(X) = \begin{cases} \alpha_i, & X \notin A, \\ \frac{1}{|A|}(1 - \alpha_i) + \alpha_i, & X \in A. \end{cases}$$

In the first example, we generated 1000 observations for training and 1000 for testing. In the second example, we assign $\alpha_1 = 0.2$ and $\alpha_2 = 0.8$, and both training and testing data sets contain 1000 observations for each class. Simulated data sets for these two examples are shown in Figure 4. The running time comparisons of three algorithms are given in Table 1. Figure 5(a) and 6(a) contain the f -curves based on training data, and Figure 5(b) and 6(b) are the c-ROC curves for testing data. For both examples, *weighting* and *propagating* approximate the theoretical boundaries very well, which implies their good generalization property. A small zigzag phenomenon was observed on the c-ROC curves generated by *propagating*. In Figure 7, we study the phenomenon by connecting all pairs of corresponding points, and it is clearly seen that the whole trend is homogeneous despite some small random movements.

4.2 An IR Data

The method *weighting* seems most promising when taking both the performance and running time into consideration. We applied *weighting* to an infra-red (IR) image data set. The data set consists of 20000 training examples and 6898 testing examples of 300 dimensions. The results are shown in Figure 8. In the c-ROC curve plot, we compared its performance with Maximal Rejection Classifier (MRC). It is clear that our method outperforms MRC.

5 Discussion

We will discuss the possibilities of techniques that can be integrated into the above framework. We also discuss the relation of our approach to other existing methodologies.

5.1 Related works

Connection with MART and Boosting. MART [8, 7] is essentially a boosting method for tree models. Boosting has been proven to be an effective method in training classifiers, see [6, 15]. A disadvantage of both MART and a direct output of boosting is that the trained classifier can be very complex: it is an additive model having a large number of components. Comparing to MART, a hierarchical model is much easier to implement, and will be guaranteed to be fast. The training of a hierarchical model can be faster than boosting. However, we expect boosting to provide a better performance.

It is interesting to note that in a successful application in computer vision, cf. [18], the researchers first use a boosting approach to find an ‘ideal’ model. They then use a cascade (hierarchical model) to ‘approximate’ the ‘ideal’ one. They reported superior performance in simulations.

Connection with MRC. MRC [5] and an enhanced approach [12, 11] use the structure of a cascade detector. The difference from our proposed method is that in training an intermediate detector f_i , we will allow different types of classifiers, e.g. SVM. The function f_i can be highly nonlinear. This has not been systematically studied in existing framework. Moreover, we will research on how to choose the function type for f_i *adaptively*. Conceptually, MRC is similar to *CER*.

5.2 Theoretical questions

The following are some theoretical problems for constructing a cascade.

1. What is the optimal strategy in deciding the single classifier at each step of a cascade?

2. Given a cascade, how to evaluate its rate of convergence? And in which sense?
3. How to characterize the generalization error of a cascade?
4. How to characterize the *rate of approximation* of a searching method to the frontier of (a class of) cascades.
5. How to guarantee the consistency of a computed cascade?

The detailed discussion on the above problems will be postponed for future publications.

5.3 Nonparametric approaches

Zhu and Hastie [19] recently proposed a nonparametric method in classification. They consider the likelihood ratio between two classes. The likelihood ratio is estimated by some kernel based nonparametric density estimation methods. In their framework, LDA and QDA become special cases. They provide evidences that their method can successfully classify data, in the situations that are hard for LDA and QDA. Their work can be considered as a generalization to the existing paradigm of LDA and QDA. It will be interesting to examine how to integrate their method into a cascade algorithm.

5.4 More on decision rules that are based on marginal distributions

Many questions that we asked above eventually transfer to how to design a classifier in a univariate case. In fact, many of the classifiers are combinations of a projection followed by a univariate classifier. If we can understand better about what an optimal univariate classifier should be, we have a good guidance to design a cascade. Note that so far, the decision rule

$$f_i(\mathbf{x}) \leq 0$$

implies a simple cut-off decision rule. In the sense of Neyman-Pearson, such a classifier is only optimal when the likelihood ratio is a monotonic function of the variable. We will study the case with more general assumptions and more data-driven methods.

5.5 Regularization

Another interesting problem is to study whether the regularization (or the penalized likelihood methods) can be a framework to build a cascade model. If yes, what is(are) the advantage(s)?

5.6 How to integrate multi-scale features

An advantage of a cascade approach is its flexibility to incorporate various types of features, e.g. multiscale features.

As a matter of fact, if we choose LDA or QDA to train the detector f_i as in Section 2, the resulting linear or quadratic decision rule gives us a ‘feature’. Since multiscale analysis has provided a set of feature, we should take advantage of them. Let $\mathbf{z} = \{z_1, z_2, \dots, z_T\}$ be the set of features. (Note that each z_i can be a linear transform of the observation \mathbf{x} .) An intuitive idea is to restrict a function f_i to be a univariate function of those features z_i ’s. By doing so, we avoid re-training from the observation \mathbf{x} . To efficiently determine function f_i , one can adopt the algorithm that has been used in CART [2]. In fact, for our purpose, a simplified version will suffice.

Note that in many multiscale transforms, e.g. wavelets, beamlets, wedgelets, and ridgelets, there exist fast discrete transform. Since the searching of an optimal univariate function f_i is fast too. There will be an fast implementation for the entire procedure.

It is possible to consider f_i as a function of a small number of features z_j ’s. By doing so, we

can incorporate the interaction between features. Note that function f_i can be highly nonlinear, and nonseparable.

5.7 The necessity of using more flexible classifiers

We compare the results between a cascade (using simple classifiers as “ $\mathbf{x}_j > c$ ”) and support vector machine using radial basis functions. Both methods are applied to an IR data set. The results are plotted in Figure 9. It is found that SVM outperforms a cascade. It is mentioned earlier that a set of linear classifier can only detect the convex hull of a target region. The experimental results seem to indicate that the target region is *not* convex. By using more complex simple classifiers in a cascade, this under-performance is likely to be overcome. We leave this as a future work.

6 Conclusion

We studied three heuristics in building a cascade. The three heuristics are (1) frontier following approximation, (2) controlling error rates, and (3) weighting. Simulations on synthetic data are carried out. It is found that weighting heuristic is optimal in both computational complexity and error rates. The current winner – weighting algorithms – is applied to an IR data set. It is found that it outperforms some state-of-the-art approaches that utilize the same type of simple classifiers. However, it fails to some classifiers that utilize more complex decision rules.

The contribution of this paper is to initiate a systematic comparison on several potential heuristics that can be utilized in building a hierarchical model. We point out the implications and promises of the cascade architecture. We describe the feasibilities of integrating a range of

techniques into this framework. Many future research directions are discussed.

A Theoretical boundary for Example 1

The class conditional probability has Gaussian distribution

$$f_i(x) = \frac{1}{(2\pi|\Sigma_i|)^{1/2}} e^{-\frac{1}{2}x^T \Sigma_i^{-1} x}, \quad i = 0, 1.$$

The posterior distribution

$$Pr(Y = i|X = x) = \frac{f_i(x)Pr(Y = i)}{\sum_{i=0,1} f_i(x)Pr(Y = i)}, \quad i = 0, 1.$$

In our example, we have $\Sigma_0 = 10I$, $\Sigma_1 = I$, and $Pr(Y = 0) = 1 - Pr(Y = 1) = 0.7$. The log-ratio is

$$\begin{aligned} \log \frac{Pr(Y = 1|X = x)}{Pr(Y = 0|X = x)} &= \log \frac{f_1(x)}{f_0(x)} + \log \frac{P(Y = 1)}{P(Y = 0)} \\ &= -\frac{1}{2}x^T \Sigma_1^{-1} x + \frac{1}{2}x^T \Sigma_0^{-1} x + \log \frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} + \log \frac{3}{7} \\ &= -\frac{9}{20}(x_1^2 + x_2^2) + \log \frac{30}{7} \leq 0. \end{aligned}$$

So a decision boundary can be described as

$$(x_1^2 + x_2^2) \leq \frac{20}{9} \log \frac{30}{7}. \tag{7}$$

References

- [1] K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu (2000). Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3), 295-313, December.
- [2] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone (1984). *Classification and regression trees*. Wadsworth Advanced Books and Software, Belmont, CA.

- [3] P. J. Burt, and E. H. Adelson (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 9(4): 532–540.
- [4] N. Cristianini and J. Shawe-Taylor (2000). *Support vector machines and other kernel-based learning methods*. Cambridge University Press, New York.
- [5] M. Elad, Y. Hel-Or, and R. Keshet (2002). Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459-1471, October.
- [6] J. Friedman, T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting. *Ann. Statist.*28 (2): 337-407.
- [7] J. H. Friedman (2001). Greedy function approximation: a gradient boosting machine. *Ann. Statist.* 29 (5): 1189–1232.
- [8] J. H. Friedman (2002). Tutorial: Getting Started with MART in R. <http://www-stat.stanford.edu/~jhf/>, April.
- [9] T. Hastie, J. Friedman, and R. Tibshirani (2001). *Elements of Statistical Learning: data mining, inference and prediction*. Springer-Verlag.
- [10] B. Heisele, T. Serre, S. Mukherjee and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. *Proceedings of 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, IEEE Computer Society Press, Kauai, Hawaii, Vol. 2, 18-24, December 2001.

- [11] Y. Hel-Or and H. Hel-Or (2002). Real time pattern matching using projection kernels. Interdisciplinary Technical Report #CS-2002-1. <http://www.faculty.idc.ac.il/toky/Publications/publications.htm>.
- [12] Y. Hel-Or and H. Hel-Or (2003). Generalized pattern matching using orbit decomposition. Submitted to the *International Conference on Image Processing*, January. <http://www.faculty.idc.ac.il/toky/Publications/publications.htm>.
- [13] J. R. Quinlan (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.
- [14] A. Said and W.A. Pearlman (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Techn.*, 6: 243–250, June.
- [15] R.E. Schapire, Y. Freund, P. Bartlett, W. S. Lee (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.* 26 (5): 1651–1686.
- [16] J.M. Shapiro (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12): 3445–3462, December.
- [17] V. Vapnik (1995), *The Nature of Statistical Learning Theory*, Springer Verlag.
- [18] P. Viola and M. Jones (2001). Robust real-time object detection. In *ICCV Workshop on Statistical and Computation Theories of Vision*, Vancouver Canada, July.
- [19] M. Zhu and T. Hastie (2002). Feature Extraction for Non-Parametric Discriminant Analysis. Accepted by and to appear in *Journal of Computational and Graphical Statistics*. Also avail-

able as Working Paper 2002-06, Department of Statistics and Actuarial Science, University of Waterloo.

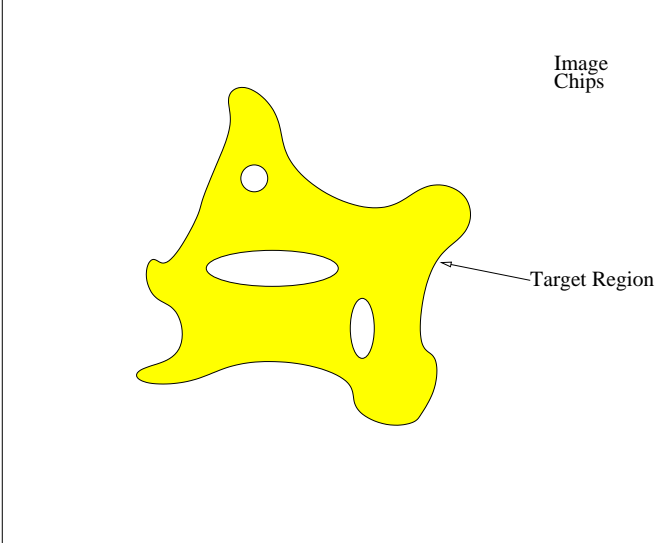


Figure 1: A target region with 'holes'. A radial basis SVM will find a good classifier in such a situation, while a set of linear classifiers will not.

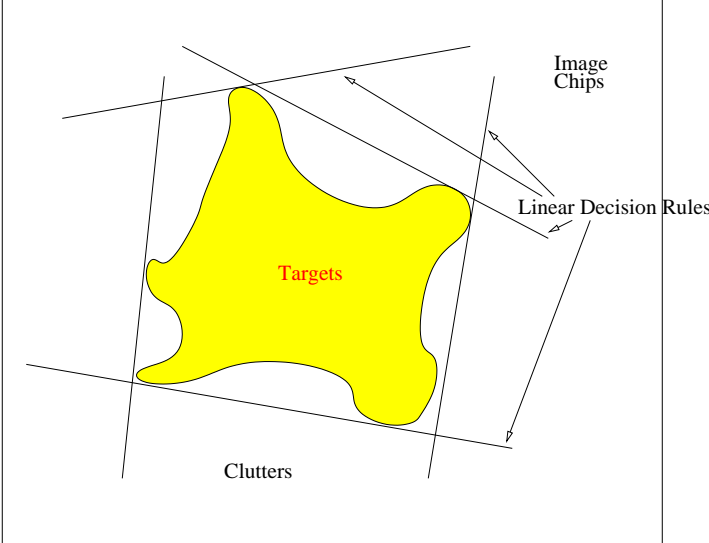


Figure 2: A set of linear classifiers can quickly find the convex hull of a target region.

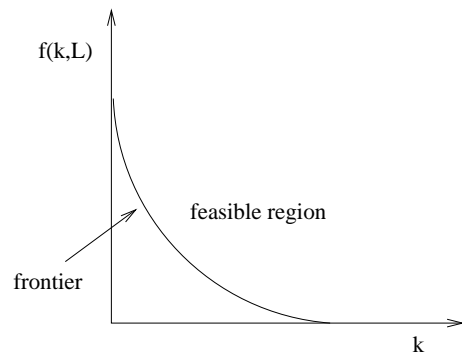


Figure 3: An illustration of the frontier

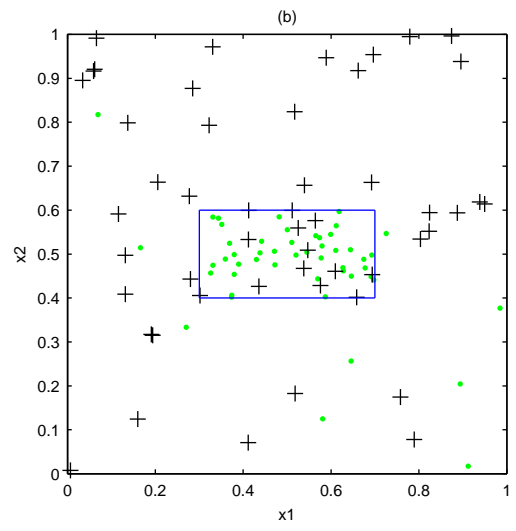
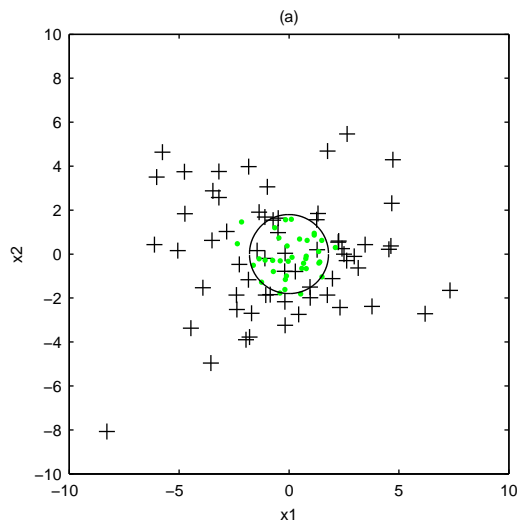


Figure 4: Illustrations of two artificial data sets with theoretical boundaries. (a) Example 1: two Gaussian distributions, with different variances, (b) Example 2: two mixed Uniform distributions, viewed from the first two dimensions, with different mixing parameters.

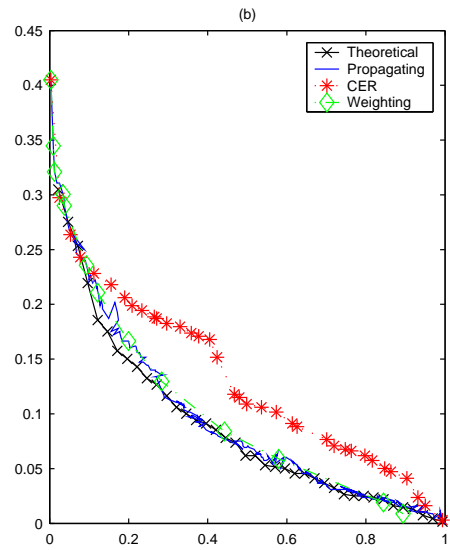
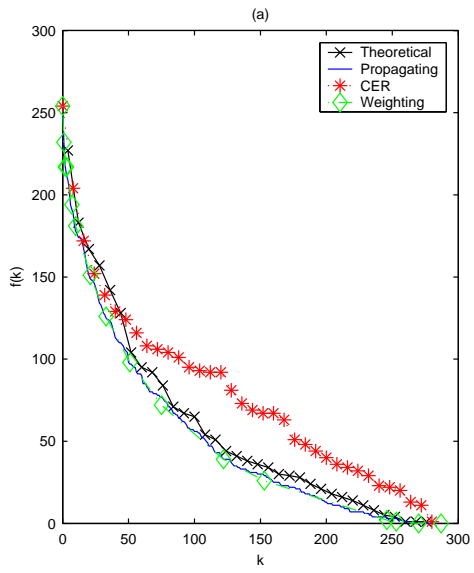


Figure 5: A comparison of three algorithms for Example 1. (a) f -curves for training data, (b) c-ROC curves for testing data. It shows that *CER* always is the worst, and *Weighting* is nearly as good as the *Propagating*. For the testing data, the closeness of *Weighting* and *Propagating* implies a good generalization properties of them in this setting.

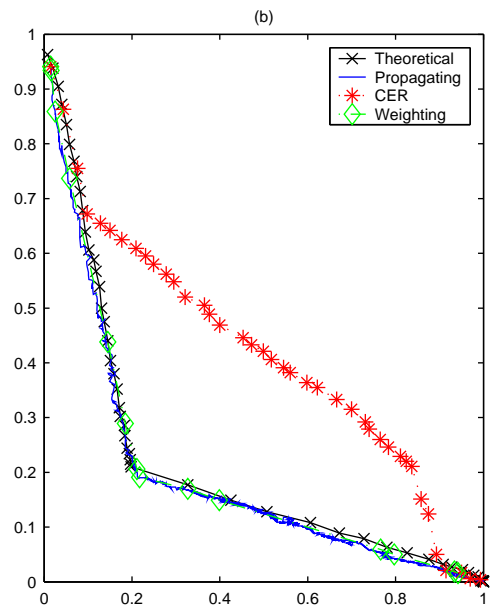
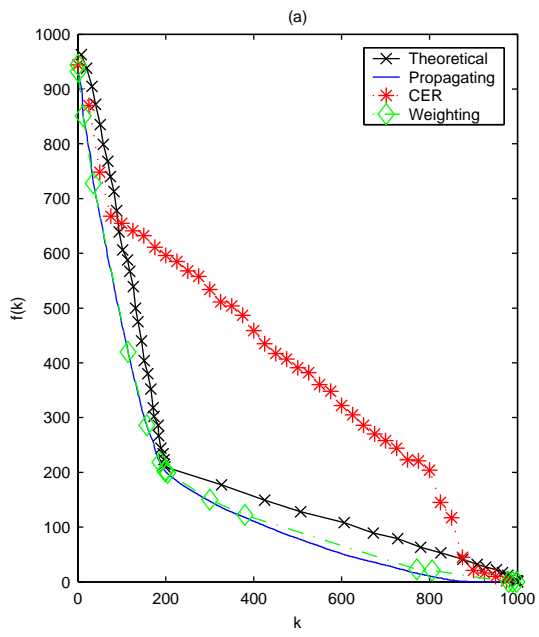


Figure 6: A comparison of three algorithms for Example 2. (a) f -curves for training data, (b) c-ROC curves for testing data. Similar conclusions as in Example 1 can be drawn here.

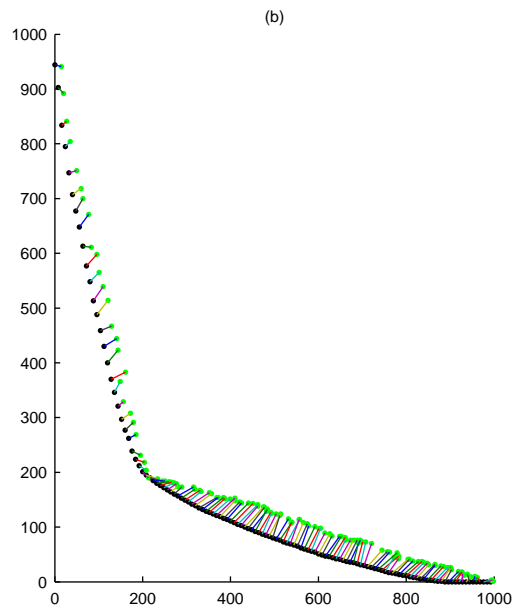
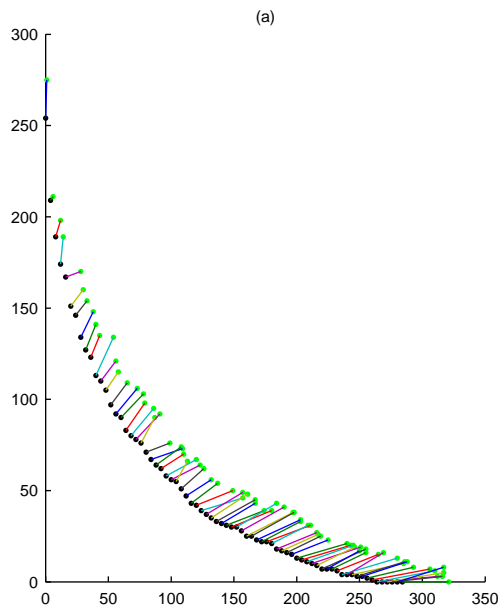


Figure 7: Dark dots denote the points on f -curve via *propagating* on the training data and light dots denote the points on the c-ROC curve for the same classifiers using the testing data. A line is drawn to connect a corresponding pair. Figure (a) is for Example 1, and (b) for Example 2. These figures show how the error rates change while the trained classifiers, which are trained by the *training* data, are applied to the *testing* data.

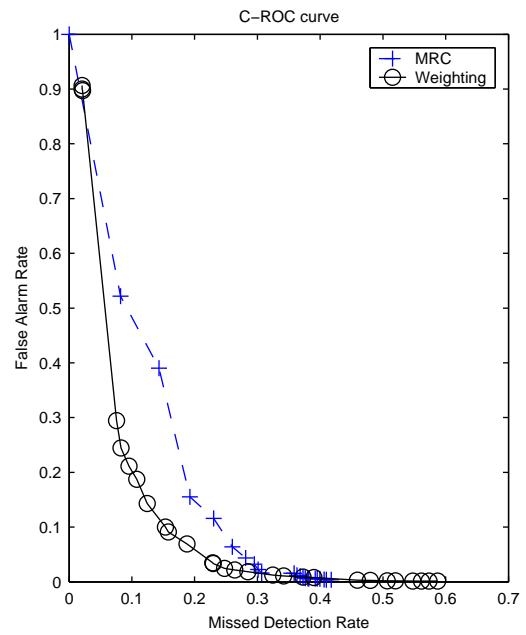
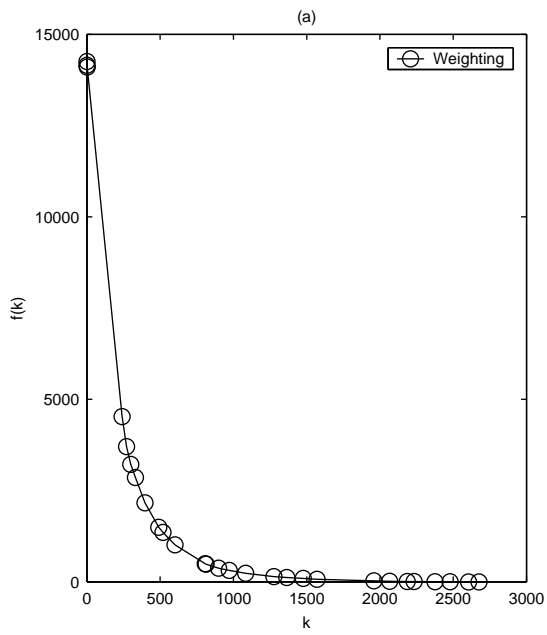


Figure 8: IR image data. (a) f -curve, (b) c-ROC curve: a comparison with MRC.

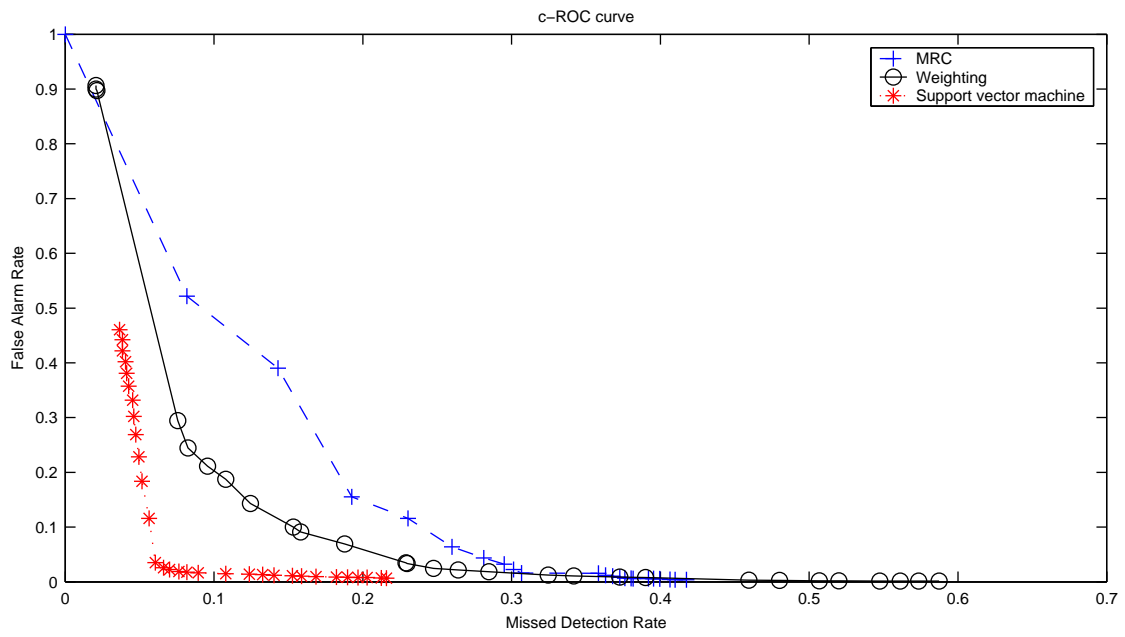


Figure 9: The comparison among MRC, Weighting and SVM on an IR data. In this case, SVM renders the best performance. The Weighting use a single-coordinate-based classifier: $\mathbf{x}_j > a$. This demonstrates the importance of using a more complex classifier at each stage.

	Propagating	CER	Weighting
Example 1	40.8	0.3	1.2
Example 2	3861.8	6.7	21.2

Table 1: Training times (in seconds) comparison for three algorithms.