Building a cascade detector and its applications in automatic target detection

Xiaoming Huo and Jihong Chen

A hierarchical classifier (cascade) is proposed for target detection. In building an optimal cascade we considered three heuristics: (1) use of a frontier-following approximation, (2) controlling error rates, and (3) weighting. Simulations of synthetic data with various underlying distributions were carried out. We found that a weighting heuristic is optimal in terms of both computational complexity and error rates. We 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 cascade architecture as well as of techniques that can be integrated into this framework is provided. The optimum heuristic—weighting algorithms—was applied to an IR data set. It was found that these algorithms outperform some state-of-the-art approaches that utilize the same type of simple classifier.

OCIS codes: 100.5010, 000.4430, 150.0150.

1. Introduction

Recently we have seen an increase of hierarchical-structure-based methodologies. A few examples follow:

- In statistics there are CART (classification and regression trees), MARS[1] (multivariate adaptive regression splines) and many other follow-up methods.
- In machine learning there are the programs C4.5,[2] perception trees,[3] and many more.
- In signal processing and harmonic analysis, we have seen quad-tree-based image coding,[4,5] pyramids,[6] and others.

A commonality of these 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 recently described techniques include the maximum rejection classifier (MRC) attributed to Elad *et al.*,[7] a similar procedure for pattern recognition reported by H. Hel-Or and Y. Hel-Or,[8,9] a successful framework in computer vision described by Viola and Jones,[10] and an application in automatic target recognition described by a Heisele *et al.*[11]

There are many advantages in using a cascade. A few of them are as follows:

1. *Interpretability*. Such a property can generate a model that is easy to interpret.
2. *Computational Efficiency*. In general, it is easy to design an algorithm whose implementation is rapid. The computational complexity of a derived algorithm will be small.
3. *Generality*. It will be easy to incorporate other features, for example, multiscale features such as wavelets, curvelets, beamlets, and edgelets.
4. *Flexibility*. The design of the algorithm is flexible: We can optimize the computing procedure based on prior knowledge of the data.

In this paper we discuss numerical strategies for searching for an optimal cascade. To study the trade-off between two types of error (false alarm and misdetection), we provide a formulation. The basic idea is

The authors are with the Georgia Institute of Technology, School of ISyE, 765 Ferst Drive, Atlanta, Georgia 30332. X. Huo's e-mail address is xiaoming@isye.gatech.edu. J. Chen's e-mail address is chenjh@isye.gatech.edu.
Received 2 May 2003.
0003-6935/04/020293-11$15.00/0

to derive algorithms that can approximate the frontier of the feasible region of pairs of two types of error rate. This is similar in spirit to the idea of using a receiver operating characteristics (ROC) curve. Three heuristics are presented. They are (1) a propagating method, (2) method of approximation by control of error rates, and (3) a method based on minimizing weighted error rates. Simulations were designed for empirical examination of the properties of each of these methods. We found that the weighting strategy is optimal in terms of both error rates and speed. This finding implies that in designing algorithms for a cascade, the weighting strategy is most favorable.

We compared our current cascade to other methods, using IR data set. Our current cascade did not outperform another existing classifier, a support vector machine (SVM), in terms of misclassification errors. However, we only allow a simple type of classifier in our hierarchy, whereas the SVM utilizes nonlinear classifiers. We discuss the limitation of our current framework and point out a direction for future improvement. Our approach is systematic, generic, and flexible. It shows strong promise for development into a powerful classifier. The current suboptimal performance in a particular data set indicates the necessity for such improvement.

The rest of the paper is organized as follows: In Section 2 we describe the general architecture of a cascade model; in Section 3, three types of heuristic and their related implementation strategies; in Section 4, the experiments and results. Section 5 contains a discussion, and 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:

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

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 (labeled 0) and $\mathcal{C}_2$ is the target class (labeled 1).

Each intermediate node is made by a simple classifier:

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

where j is the index of the intermediate node. Flow chart (1) gives a cascade classifier with three intermediate nodes. Each $f_j(\mathbf{x})$ may have one of the following forms.

1. Single entry of $\mathbf{x}$:

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

where $\mathbf{x}_j$ is the jth coordinate of vector $\mathbf{x}$.

2. Linear:

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

3. Quadratic:

$$f_j(\mathbf{x}) = c^T\mathbf{x} + \mathbf{x}^T M\mathbf{x}.$$

4. A SVM with kernels that 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 $\mathbf{K}(\,)$ is a kernel function, $\mathbf{x}_k$ are N observations, and α_k are the weights that are trained by application of a SVM.

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

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

otherwise the predicted class is $\mathcal{C}_1$. We chose b_j small enough 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, thresholds b_j will remain unchanged afterward.

Note that at the early stage of a cascade algorithm we prefer to choose a simple kernel. In later stages of a cascade algorithm we tend to choose more-complex kernels with which to explore more-difficult structures.

In building a cascade model, from the second node we exclude all the training data that are predicted by the first classifier to be in class $\mathcal{C}_1$. In other words, the second simple classifier is built for the remaining observations that are classified as belonging to class $\mathcal{C}_2$ 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 of the ways in which a decision rule can be found by use of LDA and QDA appears in Ref. 12. When a more-complex decision rule is needed, support vector machines can be applied, for example, with a radial basis function as a kernel. More details of methods for training support vector machines can be found in Ref. 13.

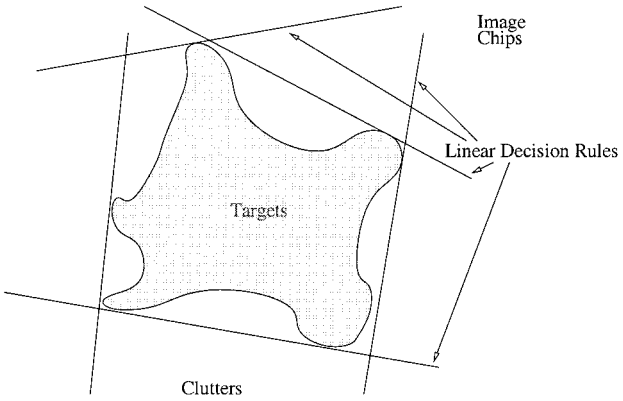

Fig. 1. A set of linear classifiers can quickly find the convex hull of a target region.

3. Geometric Tour: How (and Why) a Cascade Algorithm Works

We imagine the following simple procedure: A cascade starts with simple linear detectors (functions f_i). If targets are contained in a subregion of the space that comprises all image chips, a sequence of linear detection rules will outline a convex hull of the subregion that includes (it is hoped) most of the targets. See Fig. 1 for an illustration of this phenomenon.

There is a point beyond which no linear classifier will be effective in distinguishing a significant proportion of clutter from targets. However, it can still be true that by use of a more-complex rule, e.g., by use of quadratic or radial basis SVMs, it may be possible to distinguish a significant amount of clutter from targets. A cascade approach automatically switches to a more-complex decision rule when necessary. Figure 2 shows that, when the target region has holes, SVMs with radial basis functions can produce an efficient classifier, whereas both LDA and QDA will fail. Note that because many clutters were rejected in previous stages, the sample size at the later stage is small. Hence SVMs can be trained efficiently.

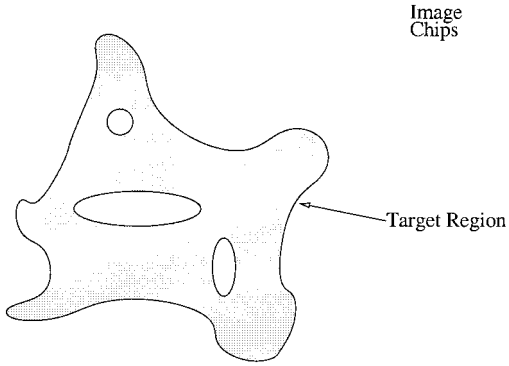

Fig. 2. Target region with holes. A radial basis SVM will find a good classifier in such a situation, whereas a set of linear classifiers will not.

4. Optimal Strategies for Building a Cascade

We consider what the optimal strategies are for building a cascade classifier. We limit our analysis to 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 with which to compare different strategies.

In Section 3.A we formulate the problem. We provide the necessary notation with which to analyze the model. In Section 3.2 we describe three types of heuristic and strategies for their implementation in various circumstances.

A. Cascade Classifier

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

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

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

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

where $\mathbf{x}_j$ is the jth component of 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 inequality (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 number 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 the 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 the decision rule $f_i(x) < t_i$, $1 \leq i \leq L$; and let N_{2i} denote the number of remaining observations. Because of the hierarchical structure of a cascade, we must have

$$\begin{aligned} N &= N_1 + N_2, \\ N_{2i-2} &= N_{2i-1} + N_{2i}, \quad i = 2, 3, \ldots, L, \\ N_{2i-2}{}^j &= N_{2i-1}{}^j + N_{2i}{}^j, \quad i = 2, 3, \ldots, L, \quad j = 0, 1. \end{aligned}$$

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 classes 0 and 1, respectively. The number of 1's that are misclassified as 0's is

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

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

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

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

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

A hierarchical classifier that has architecture (3) is called a level-L cascade. Recall that each simple classifier is a binary classifier. Let $\mathcal{H}(L)$ denote the entire 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 architecture (3), we have

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

We define the following function:

$$\begin{aligned} f(k; L) = \min_{h\in\mathcal{H}(L)} \quad & E_{0\rightarrow 1}(h), \\ \text{subject to} \quad & E_{1\rightarrow 0}(h) \le k, \end{aligned}$$

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

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

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

We consider strategies with which we can compute an optimal cascade. There are three classes of heuristic: frontier-following (Subsection 3.B.1 below), controlled error rates (Subsection 3.B.2), and weighting (Subsection 3.C.3).

Before describing these heuristics in detail, we 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 our knowledge of structured risk minimization.[14] 3. Because of the difficulty of computing the optimal cascade, we hope to utilize some greedy algorithm with which to find a suboptimal cascade. The heuristics, which are described below, are ways with which to explore various possible solutions to this problem. 4. The frontier gives a set of cascade classifiers. The particular choice of cascade depends on the costs of different types of error. It is problem dependent, and we have chosen not to pursue it here.

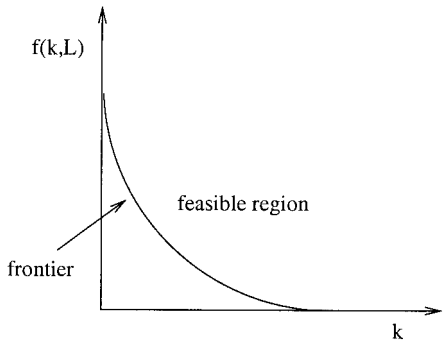

Fig. 3. Illustration of the frontier.

B. Three Heuristics

We describe three types of heuristic. In Subsection 3.B.1 we describe strategies that follow the frontier of the feasible region; in Subsection 3.B.2, a heuristic that is based on controlled error rates; in Subsection 3.B.3, a strategy that is based on the weighted misclassification rate.

1. *Frontier Following*

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

$$\begin{aligned} E_{1\rightarrow 0}([h^*(k, L)] &= k, \\ E_{0\rightarrow 1}[h^*(k, L)] &= f(k; L). \end{aligned}$$

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 with which to compute classifier $h^*(k, 1)$, $k = 0, 1, 2, \ldots, N^1$. For example, if the classifiers are those that are used in CART,

$$\mathbf{x}_j \ge a.$$

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

Propagating. We develop a propagation algorithm with which to approximate the frontier. We start with some notation. 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 rates $E_{1\to 0}$ are no larger than k_1. Apparently, for a classifier $h \in H_{A,k_1}$ we have $E_{1\to 0}(h) \le k_1$. Note that the domain of classifier h is region A. Let $r^*(A, k_1)$ denote the classifier that is in set H_{A,k_1} and takes the minimum of $E_{0\to 1}$:

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

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

Algorithm *Prop*.

1. Let $h'(k, 1) = h^*(k, 1)$, $k = 0, 1, 2, \ldots, N^1$.
2. For $l \ge 1$ let $h'(k, l + 1)$, $k = 0, 1, 2, \ldots$ denote the solution to the following optimization problem:

$$\min_{k_1\le k} E_{0\to 1}[h'(k_1, l)] + E_{0\to 1}(r^*\{R^1[h'(k_1, l)], k - k_1\}). \quad (4)$$

End.

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

Note that the number N^1 in the first row of Algorithm Prop can be reduced to a small value, given that the error rate, $E_{1\to 0}$, can be controlled to be significantly smaller than the quantity N^1. Making such a reduction can save large amount of computation.

Implementational Strategy. We describe a numerically appealing alternative way to implement the algorithm in Subsection 3.B.1. Let K denote a fixed integer. We consider how to compute $h'(k, l + 1)$, $0 \le k \le K$; whereas $h'(k, l)$ is given. The main idea is illustrated by the following table:

	0	1	2	⋯	$K-1$	K
	$h'(0, l)$	$h'(1, l)$	$h'(2, l)$	⋯	$h'(K-1, l)$	$h'(K, l)$
	A_0	A_1	A_2	⋯	A_{K-1}	A_K
0	$r^*(A_0, 0)$	$r^*(A_1, 0)$	$r^*(A_2, 0)$	⋯	$r^*(A_{K-1}, 0)$	$r^*(A_K, 0)$
1	$r^*(A_0, 1)$	$r^*(A_1, 1)$	$r^*(A_2, 1)$	⋯	$r^*(A_{K-1}, 1)$	
⋮	⋮	⋮	⋮			
$K-2$	$r^*(A_0, K-2)$	$r^*(A_1, K-2)$	$r^*(A_2, K-2)$			
$K-1$	$r^*(A_0, K-1)$	$r^*(A_1, K-1)$				
K	$r^*(A_0, K)$					

In this table we have

$$A_i = R^1[h'(i, l)], \qquad i = 0, 1, \ldots, K.$$

Following a description similar to that in Subsection 3.B.1, for entries in one column of the table above,

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

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

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

The complexity of a propagating algorithm can be relatively high, especially when error rate $E_{1\to 0}$ is large. Suppose that the simple classifiers are based on single variate functions, as in inequality (2). For a fixed coordinate, the computational complexity of finding the optimal classifier should be at least of the order of complexity in sorting 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 the table above, which has $O(K^2)$ entries. Overall, the computational complexity is $O[K^2dN \log(N)]$.

2. Controlled Error Rate

Sometimes we may want to quickly compute a suboptimal cascade. The following approach can be adapted:

Algorithm *CER*.

1. Let $\tilde{h}_0$ be a classifier that classifies everything into class 1:

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

2. Set $k = 0$, $l = 0$.
3. While the last improvement is greater than 0 (in the first attempt, this condition was always true)
Find $r^*[R^1(\tilde{h}_l), 0]$;
Let $\tilde{h}_{l+1}$ be the combination of $\tilde{h}_l$ and $r^*[R^1(\tilde{h}_l), 0]$;
Update: $l \leftarrow l + 1$, $\tilde{h}_l \leftarrow \tilde{h}_{l+1}$.
Record the last improvement as $E_{0\rightarrow 1}(\tilde{h}_{l+1}) - E_{0\rightarrow 1}(\tilde{h}_l)$.
End.
4. Set $h'(k) = \tilde{h}_l$.
5. Set $k \leftarrow k + 1$.
Find $r^*[R^1(\tilde{h}_l), 1]$;
Let $\tilde{h}_{l+1}$ be the combination of $\tilde{h}_l$ and $r^*[R^1(\tilde{h}_l), 1]$;
Update: $l \leftarrow l + 1$, $\tilde{h}_l \leftarrow \tilde{h}_{l+1}$.
6. If $k = K$, where K is a predetermined maximal allowable error rate $E_{1\rightarrow 0}$, then stop; otherwise, go back to step 4.
7. Function $h'(k)$, $k = 0, 1, 2, \ldots, K$, gives us an approximation of 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 that corresponds to $f(k)$, or, in other words,

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

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

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

3. Weighting

We propose an alternative to the method of controlled error rate. This alternative includes using Lagrangian multipliers to locate the frontier of a feasible region.

Let λ be a potentially large positive constant. By following an argument similar to that in Subsection 3.B.1 we can solve the following optimization problem efficiently:

$$s^*(A, \lambda) = \text{argmin}_{h\in\mathcal{H}(A)}, \qquad E_{0\rightarrow 1}(h) + \lambda E_{1\rightarrow 0}(h), \tag{5}$$

where A denotes a region, the set $\mathcal{H}$(A) includes all simple classifiers that reside in 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 $E_{0\rightarrow 1}$ and $E_{1\rightarrow 0}$ simultaneously.

We propose the following algorithm:

Algorithm *Weighting*.
1. Let $\tilde{h}_0$ be a classifier that classifies everything into class 1:

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

2. Set $l = 0$.
3. Choose a large initial value (denoted λ_0) for λ: $\lambda = \lambda_0$.
4. While the last improvement in the objective function in problem (5) is significantly nonzero,
find $s^*[R^1(\tilde{h}_l), \lambda]$;
Let $\tilde{h}_{l+1}$ be the combination of $\tilde{h}_l$ and $s^*[R^1(\tilde{h}_l), \lambda]$;
Update: $l \leftarrow l + 1$, $\tilde{h}_l \leftarrow \tilde{h}_{l+1}$;
Record the decreasing of the objective function that is in problem (5).
The decrement above is the last improvement.
End.
5. Let $k = E_{1\rightarrow 0}(\tilde{h}_l)$, $f(k) = E_{0\rightarrow 1}(\tilde{h}_l)$.
6. Let λ take a smaller value: $\lambda \leftarrow \lambda/\theta$, e.g., $\theta = 2$.
7. If $f(k) < 1$, then go back to 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 achieve different trade-offs between the two error rates: $E_{0\rightarrow 1}$ and $E_{1\rightarrow 0}$.

5. Experiments

In this section our objective is to assess the performance of the three heuristic algorithms above. Experiments were carried out with two synthetic data sets and a set of infrared image data. For simplicity we stayed with univariate functions employed in CART. In the results, the approximate frontier f curves reflected the empirical performance with the training data. And the generalization performance of the testing data was evaluated by the complementary ROC (c-ROC) curves, with the x axis representing the missed detection rate and the y axis representing the false-alarm rate. Each point on an f curve corresponded to a cascade, which we used to generate a point on the c-ROC curve.

All experiments were conducted in Matlab codes.

A. Synthetic Data

In the following two examples we simulated two data sets with different underlying distributions and compared the performance of the three heuristic algorithms. For the propagating method 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 that of $y = 0$ 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

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

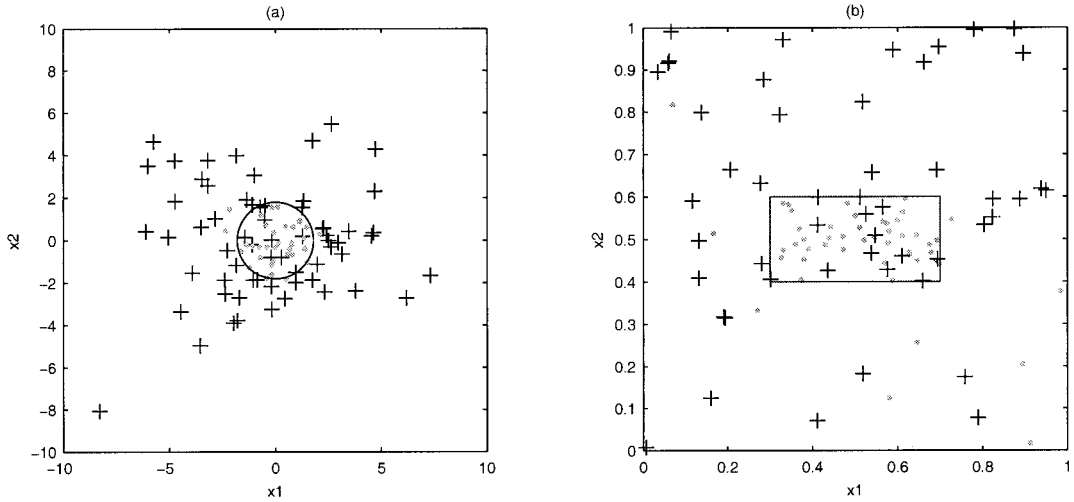

Fig. 4. Illustrations of two artificial data sets with theoretical boundaries. (a) Example 1: two Gaussian distributions with different variances. (b) Example 2: two mixed uniform distributions, viewed from the first two dimensions, with different mixing parameters.

An explanation of the above decision rule appears in Appendix A.

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

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

In the first example we generated 1000 observations for training and 1000 for testing. In the second example we assigned $\alpha_1 = 0.2$ and $\alpha_2 = 0.8$, and both training and testing data sets contained 1000 observations for each class. Simulated data sets for these two examples are shown in Fig. 4. The propagation, controlled error rate and weighting training times (in seconds) were 40.8, 0.3, and 1.2, respectively, for example 1 and 3861.8, 6.7, and 21.2 for example 2. Figures 5(a) and 6(a) contain the f curves based on training data, and Figs. 5(b) and 6(b) are the c-ROC curves for the testing data. For both examples, the weighting and propagating techniques approximate the theoretical boundaries well, which implies their good generalization properties. A small zigzag phenomenon was observed on the c-ROC curves generated by the propagating technique. In Fig. 7 we illustrate the phenomenon by connecting all pairs of corresponding points, and it can be clearly seen that the whole trend is homogeneous despite some small random movements.

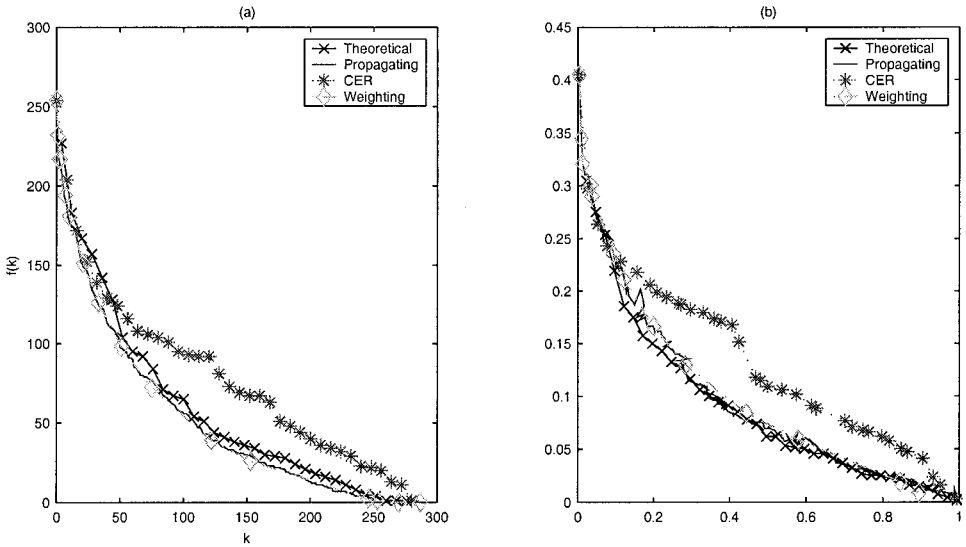

Fig. 5. Comparison of the three algorithms for Example 1: (a) f curves for training data, (b) c-ROC curves for testing data. The controlled error rate (CER) algorithm always is the worst, and the weighting is nearly as good as the propagating algorithm. For the testing data, the similarity of the weighting and the propagating algorithms implies their good generalization properties in this setting.

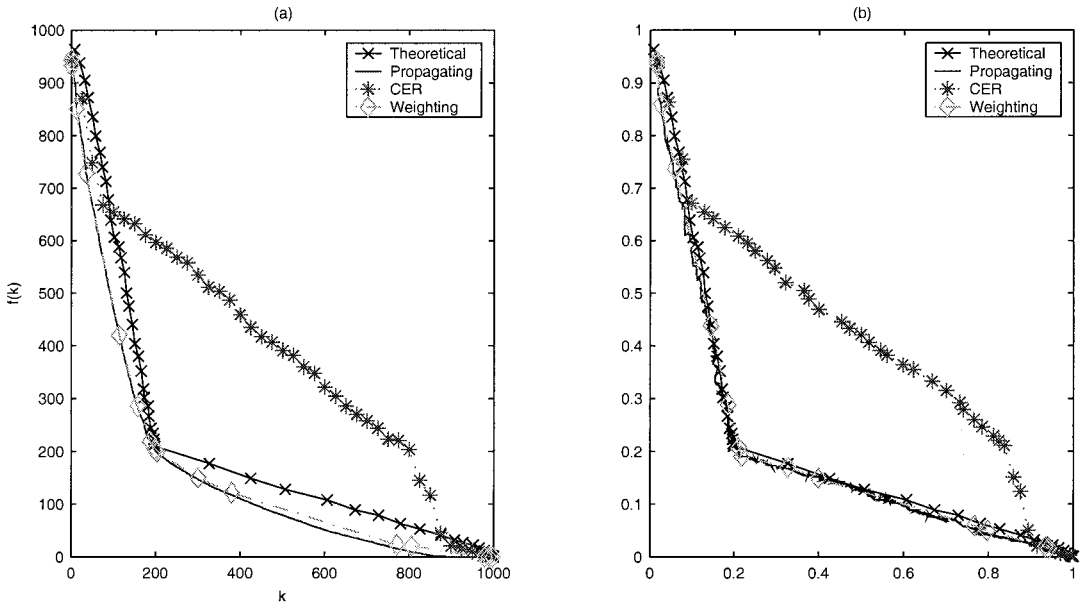

Fig. 6. Comparison of the three algorithms described for Example 1 in Fig. 5 for Example 2: (a) f curves for training data, (b) c-ROC curves for testing data. Similar conclusions to those for Example 1 can be drawn here.

B. Infrared Data

The weighting method seems most promising when both performance and running time are taken into consideration. We applied the weighting method to an infrared image data set. The data set consisted of 20,000 training examples and 6898 testing examples of 300 dimensions. The results are shown in Fig. 8. The c-ROC curve plot compares the performance of the infrared data set with the MRC. It is clear that our method outperforms the MRC.

6. Discussion

We discuss techniques that can be integrated into the framework described above. We also compare our approach with those of other existing methodologies.

A. Related Works

1. *Connection with MART and Boosting*

MART[15,16] is essentially a boosting method for tree models. Boosting has proved to be an effective

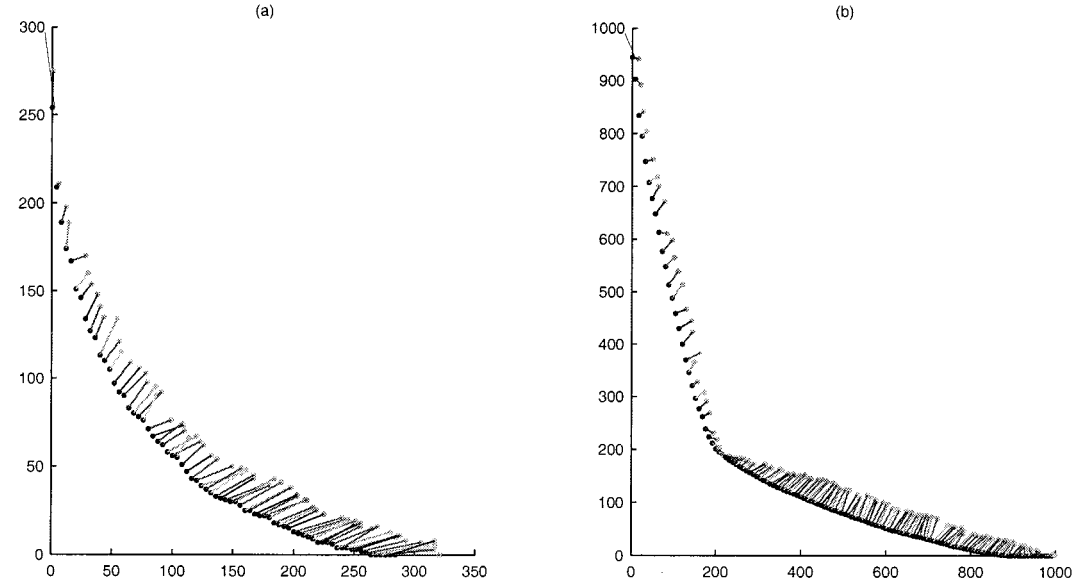

Fig. 7. Filled circles, point on the f curve by means of propagating on the training data; and open circles, points on the c-ROC curve for the same classifiers and the testing data. Curves have been drawn to connect corresponding pairs. (a) Example 1, (b) Example 2. The way in which the error rates change as the trained classifiers, which are trained by the training data, are applied to the testing data is shown.

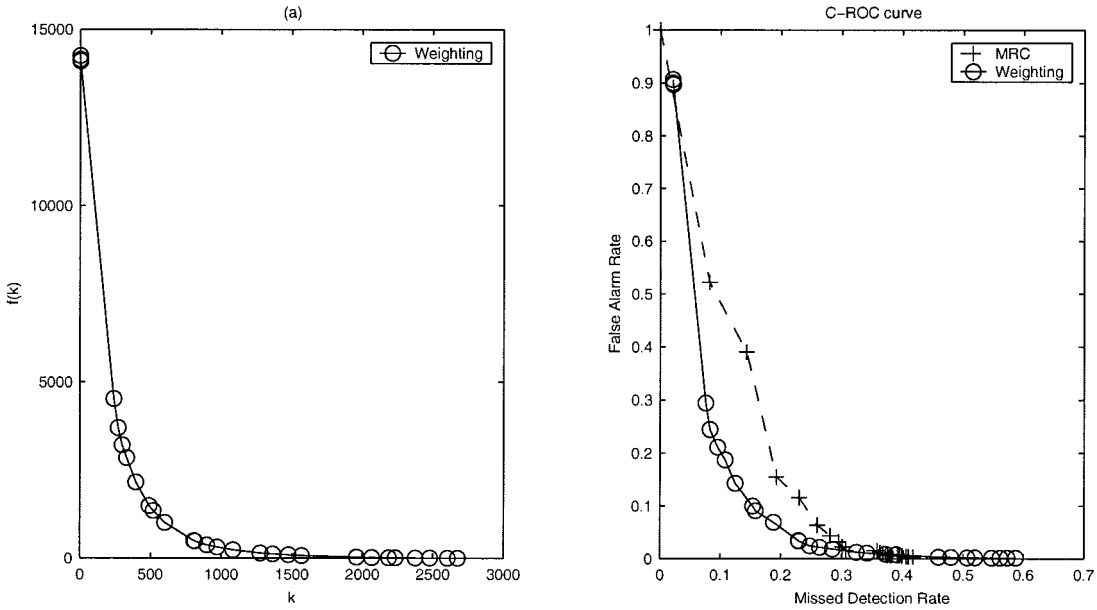

Fig. 8. Infrared image data: (a) f curve; c-ROC curve, comparison with the MRC.

method for training classifiers; see Refs. 17 and 18. A disadvantage of both MART and a direct output of boosting is that the trained classifier can be highly complex: It is an additive model that has a large number of components. Compared with MART, a hierarchical model is much easier to implement and is guaranteed to be fast. The training of a hierarchical model can be faster than the boosting approach. However, we expect boosting to provide a better performance.

It is interesting to note that in a successful application in computer vision (cf. Ref. 10) the researchers first used a boosting approach to find an ideal model and then used a cascade (hierarchical model) to approximate the ideal model. Superior performance in simulations has been reported.

2. *Connection with the MRC*

The MRC[7] and an enhanced approach[8,9] use the structure of a cascade detector. The difference from our proposed method is that in training an intermediate detector f_i we allow the use of different types of classifier, e.g., SVM. Function f_i can be highly nonlinear. This phenomenon has not been systematically studied in an existing framework. Moreover, we shall examine ways to choose the function type for f_i adaptively. Conceptually, the MRC is similar to the controlled error rate model.

B. Theoretical Questions

Following are some theoretical questions that need to be answered in constructing a cascade:

1. What is the optimal strategy for deciding the single classifier at each step of a cascade?
2. Given a cascade, how does one evaluate its rate of convergence and in which sense?
3. How does one characterize the generalization error of a cascade?
4. How does one characterize the rate of approximation of a searching method to the frontier of (a class of) cascades?
5. How does one guarantee the consistency of a computed cascade?

C. Nonparametric Approaches

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

D. More on Decision Rules That Are Based on Marginal Distributions

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

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

implies a simple cutoff decision rule. Such a classifier is optimal only when the likelihood ratio is a monotonic function of the variable. We address the

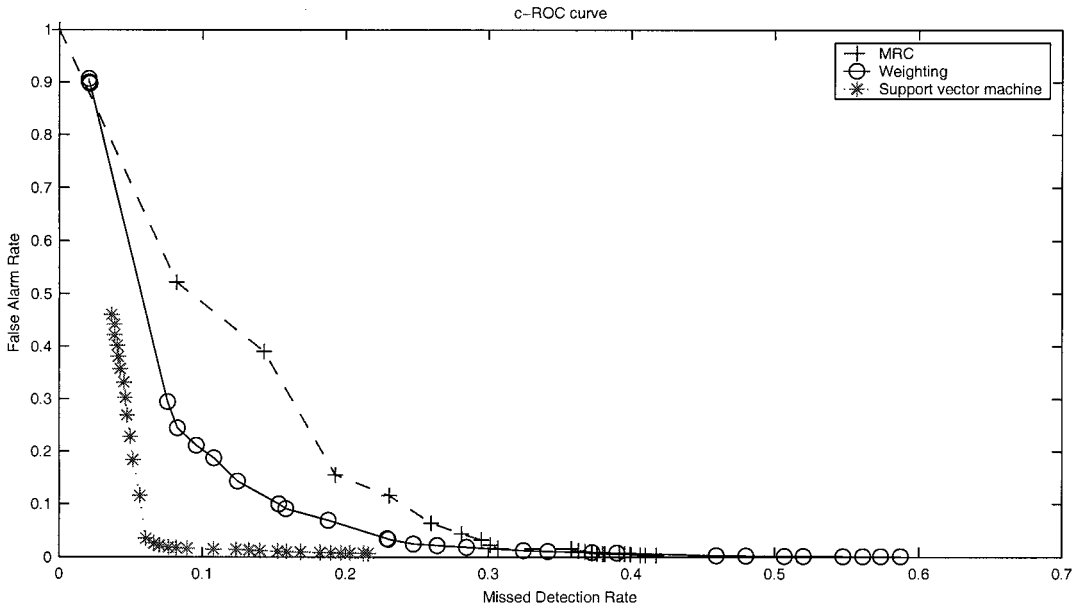

Fig. 9. The comparison among MRC, weighting, and SVM algorithms for infrared data. In this case the SVM yields the best performance. The weighting algorithm uses a single-coordinate-based classifier: $\mathbf{x}_j > a$. This demonstrates the importance of using a more-complex classifier at each stage.

case with more general assumptions and more data-driven methods.

E. Regularization

Another interesting problem is to determine whether the regularization (or the penalized likelihood) method can be a framework with which to build a cascade model. If the answer is yes, what are the advantages?

F. How To Integrate Multiscale Features

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

As a matter of fact, if we choose LDA or QDA to train detector f_i as in Section 2, the resultant linear or quadratic decision rule gives us an outcome that we shall call a feature. Because multiscale analysis has provided a set of features, we should take advantage of them. Let $\mathbf{z} = \{z_1, z_2, \ldots, z_T\}$ 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. By doing so, we avoid retraining from observation $\mathbf{x}$. To determine function f_i efficiently, one can adopt the algorithm that was used in CART.[1] In fact, for our purpose, a simplified version will suffice.

Note that in many multiscale transforms, e.g. wavelets, beamlets, wedgelets, and ridgelets, there are fast discrete transforms. Because searching of an optimal univariate function f_i is fast too, the entire procedure can be quickly implemented.

It is possible to consider f_i a function of a small number of features z_j. By doing so, we can incorporate the interaction between features. Note that function f_i can be highly nonlinear and nonseparable.

G. Necessity of Using More-Flexible Classifiers

We compared the results between a cascade (using simple classifiers as $\mathbf{x}_j > c$) and a support vector machine, using radial basis functions. Both methods were applied to an infrared data set. The results are plotted in Fig. 9. We found that SVM outperforms a cascade. We mentioned above that a set of linear classifiers can detect only 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 we are likely to overcome this underperformance. We leave this issue for future study.

6. Conclusions

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

In this paper we have initiated a systematic comparison of several potential heuristics that can be utilized in building a hierarchical model. We pointed out the implications and advantages of the cascade architecture. We described the feasibility of integrating a range of techniques into this framework. Many future research directions have been pointed out.

Appendix A. Theoretical Boundary for Example 1

The class conditional probability has the Gaussian distribution

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

The posterior distribution is

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

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

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

So a decision boundary can be described as

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

References

1. L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, *Classification and Regression Trees* (Wadsworth, Belmont, Calif., 1984).
2. J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, Los Altos, Calif., 1993).
3. K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu, "Enlarging the margins in perceptron decision trees," Mach. Learning **41,** 295–313 (2000).
4. A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. Video Technol. **6,** 243–250 (1996).
5. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Trans. Signal Process. **41,** 3445–3462 (1993).
6. P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," IEEE Trans. Commun. **9,** 532–540 (1983).
7. M. Elad, Y. Hel-Or, and R. Keshet, "Pattern detection using a maximal rejection classifier," Pattern Recogn. Lett. **23,** 1459–1471 (2002).
8. Y. Hel-Or and H. Hel-Or, "Real time pattern matching using projection kernels," interdisciplinary Tech. Rep. CS-2002-1 (2002), http://www.faculty.idc.ac.il/toky/Publications/publications.htm.
9. Y. Hel-Or and H. Hel-Or, "Generalized pattern matching using orbit decomposition," presented at the International Conference on Image Processing, Barcelona, Spain (September 2003), http://www.faculty.idc.ac.il/toky/Publications/publications.htm.
10. P. Viola and M. Jones, "Robust real-time object detection," presented at the ICCV Workshop on Statistical and Computation Theories of Vision, Vancouver, B.C., Canada (July 2001).
11. B. Heisele, T. Serre, S. Mukherjee, and T. Poggio, "Feature reduction and hierarchy of classifiers for fast object detection in video images," in *Proceedings of 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (*CVPR 2001*) (IEEE Computer Society Press, Los Alamitos, Calif., 2001), Vol. 2, pp. 18–24.
12. T. Hastie, J. Friedman, and R. Tibshirani, *Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer-Verlag, Berlin, 2001).
13. N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge U. Press, New York, 2000).
14. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer-Verlag, Berlin, 1995).
15. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Ann. Statist. **29,** 1189–1232 (2001).
16. J. H. Friedman, "Getting Started with MART in R," tutorial (April 2002), http://www-stat.stanford.edu/~jhf/.
17. J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," Ann. Statist. **28,** 337–407 (2000).
18. R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," Ann. Statist. **26,** 1651–1686 (1998).
19. M. Zhu and T. Hastie, "Feature extraction for non-parametric discriminant analysis," J. Comput. Graphic. Statist. (to be published).