Catenary Support Vector Machines

Kin Fai Kan and Christian R. Shelton

Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521, USA
{kkan,cshelton}@cs.ucr.edu

Abstract. Many problems require making sequential decisions. For these problems, the benefit of acquiring further information must be weighed against the costs. In this paper, we describe the *catenary support vector machine* (catSVM), a margin-based method to solve sequential stopping problems. We provide theoretical guarantees for catSVM on future testing examples. We evaluated the performance of catSVM on UCI benchmark data and also applied it to the task of face detection. The experimental results show that catSVM can achieve a better cost tradeoff than single-stage SVM and chained boosting.

1 Introduction

Many problems require making sequential decisions. In product testing, parts are inspected throughout the manufacturing process. Humans or computers must decide whether to continue manufacturing or whether to stop (in case the piece is not salvageable). In medical diagnosis, doctors, patients, and insurers must decide whether the current information is sufficient to make a decision or whether to conduct the next of a bank of tests. In both of these cases, the benefit of further processing must be weighed against the costs. The acquisition of new information is costly.

In object detection in images, a similar problem is faced. Scanning an image for an object of interest takes processing time. If the image can be scanned more quickly or at a lower resolution (reducing the number of pixels to be examined), the detection can be sped up. In doing so, the speed of detection must be weighed against the accuracy of detection.

Most classification methods assume full information about testing examples and are thus not suitable for sequential decision making scenarios. Recently, Shelton et al. [1] proposed chained boosting to solve sequential stopping problem. They assume that the relative costs of stopping at each stage are known and can be made explicit. Given the stopping costs for each training example, the goal is to minimize the cost of the decision rules applied to future examples. The difficulty of the problem lies in the fact that the decisions in later stages depend on what happens in early stages. Motivated by the success of support vector machines (SVMs) in many classification problems, this paper presents the *catenary support vector machine* (catSVM), a margin-based method to solve sequential stopping problems.

2 Related Work

We are interested in direct estimation of a sequence of decision rules. For this reason we are not considering density estimation (like a hidden Markov model) followed by a cost analysis to derive the decision rules. This rules out approaches like influence diagrams [2] as we would like to skip the density estimation step.

Our formulation (see the next section) appears similar to cascade classification [3,4,5] in that there are stages of classification. For applications like face detection, negative examples are far more frequent than positive examples. Rejecting negative examples as quickly as possible is crucial to the speed of the classification process. Viola and Jones [3] propose an iterative approach to train the cascade. In each iteration, a new stage is added to the cascade and a new stage classifier is trained to achieve a very low false negative rate and an approximately 50% false positive rate using a modified AdaBoost algorithm. Stages are added to the cascade until the number of false positives is reduced below a small number on a validation set. Bi et al. [6] propose using 1-norm SVM as the stage classifiers in the cascade. Like Viola and Jones, their approach trains the stage classifiers sequentially from the first stage to the last stage. In every stage, an 1-norm SVM is trained to minimize the sum of the weighted errors and the regularization term.

There are two major differences between our problem formulation and cascade classification. First, although classification speed is important, we are mainly concerned about the costs of gathering information (i.e., the feature costs). Also, while cascade classification requires the user to choose the desired false negative rate and false positive rate at every stage, we assume that the feature costs and the misclassification costs are explicitly specified by the user and our algorithm automatically determines the best tradeoff between the feature cost and the two types of errors.

Second, we optimize the stage classifiers as a group to maximize the overall performance of the processing pipeline. The problem formulation allows the information available to change at each stage. Thus, false-positive and false-negative rates at each stage are not sufficient. It matters *which* positive examples are incorrectly rejected at a stage, not just *how many*. In particular, examples for whom further processing would still result in the incorrect classification should be rejected, while those for whom further information would clarify their classification should be saved.

Cascade classification and catSVM are both "staged" classifiers, but they are more complementary than competitive. The former attempts to speed up the computation of a single classification task (fixed information) by exploiting the asymmetric distribution of examples while the latter attempts to speed up a decision task by exploiting the correlation between data sources gathered at different times. One could well imagine using cascade classifiers at each stage within the framework shown here.

The prior algorithms that are closest to our work are two recent papers, [7] and [1]. Dundar and Bi [7] consider the problem of jointly optimizing cascaded SVM classifiers. However, they ignore the difference of rejecting an example at different stages. They formulate a non-convex and non-linear objective function and propose a cyclic optimization algorithm to optimize it. Shelton et al. [1] consider using boosting to build a classifier pipeline. They formulate a loss function in terms regular costs and propose a upper bound of the conjunction of indicator functions using a product of exponential functions. The resulting upper bound of the loss function is convex and easy to optimize, but it may be too loose to approximate the loss function well. We demonstrate evidence to this effect in our experimental results.

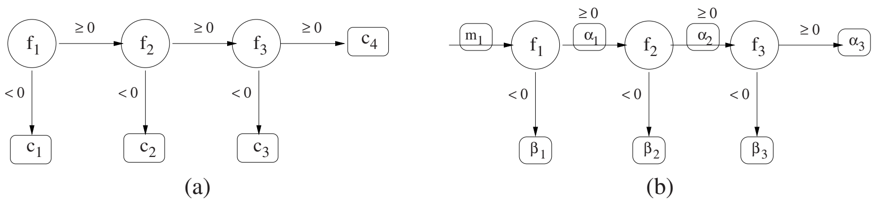


Fig. 1. An example of a three-stage processing pipeline

3 Sequential Stopping Problem

We follow the problem formulation in Shelton et al. [1]. There is no assumption about the structure of the costs. Rather, we assume that each training example carries a cost vector indicating the costs of stopping after each stage. These costs may increase, decrease, or have any other arbitrary relationship with the stage index. The costs might be function of a "label" or might be different for each example.

Let S be the number of stages in the processing pipeline. Denote the feature vector and the costs of an example by x and c, respectively. Let x_j be the components of x that are available at the j-th stage. Let c_j be the total cost of rejecting the example at the j-th stage and c_{S+1} be the total cost of accepting it (allowing it to "pass" at each decision). We assume that c is drawn from a known set $\mathcal{C}$. In the case of binary classification, $\mathcal{C}$ might be of cardinality 2: one sequence of costs for positive examples, and one sequence for negative examples. In general, $\mathcal{C}$ can be of any size. The only requirement is that the maximum magnitude of the members of $\mathcal{C}$ be bounded. Figure 1(a) shows an example of a three-stage processing pipeline.

Denote the classifier at the j-th stage by f_j and the entire processing pipeline by f. A positive value for f_j indicates that processing should continue, while a negative value indicates processing should stop. The loss for an example is therefore

$$L(f(x), c) = \sum_{j=1}^{S} \left(c_j I\left[f_j(x_j) < 0\right] \prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) + c_{S+1} \prod_{k=1}^{S} I\left[f_k(x_k) \geq 0\right] \ . \tag{1}$$

The goal is to find S classifiers, one for each stage, which together minimize $\mathbf{E}[L(f(x), c)]$. Although we do not know the true distribution of (x, c), we can use the empirical loss as a surrogate. We let $\{(X_1, C_1), \ldots, (X_N, C_N)\}$ denote the training set and X_{ij} denote the features of X_i that are available at the j-th stage. Analogously, we let C_{ij} denote the cost associated with X_i at the j-th stage.

4 Catenary Support Vector Machines

We return to the formulation in Section 3 and derive an optimization procedure based on a loss bound.

4.1 Loss Bound

We start by re-writing the loss function (1) in terms of incremental costs.

$$L(f(x), c) = m_1 + \sum_{j=1}^{S} \left(\prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) \Big(\alpha_j I\left[f_j(x_j) \geq 0\right] + \beta_j I\left[f_j(x_j) < 0\right] \Big) \quad (2)$$

where for $j = 1, \ldots, S$,

$$m_j = \begin{cases} \min\left(m_{j+1}, c_j\right) & \text{if } j < S, \\ \min\left(c_{j+1}, c_j\right) & \text{if } j = S; \end{cases}$$

$$\alpha_j = \begin{cases} m_{j+1} - m_j & \text{if } j < S, \\ c_{j+1} - m_j & \text{if } j = S; \end{cases}$$

$$\beta_j = c_j - m_j \quad .$$

In words, m_j is the minimal cost at stage j or later. α_j is the incremental increase in the minimal cost by continuing processing and β_j is the incremental cost of stopping processing. Note that either α_j or β_j is positive but not both. We denote the incremental costs associated with X_i at the j-th stage by m_{ij}, α_{ij}, and β_{ij}. Figure 1(b) shows a three-stage processing pipeline with the incremental costs.

It is hard to minimize (2) directly. We define a upper bound for $L(f(x), c)$ and minimize the upper bound instead.

$$\hat{L}(f(x), c) = m_1 + \sum_{j=1}^{S} \left[\alpha_j \left(U_j^{\alpha}(x) - V_j^{\alpha}(x) \right) + \beta_j \left(U_j^{\beta}(x) - V_j^{\beta}(x) \right) \right] \quad (3)$$

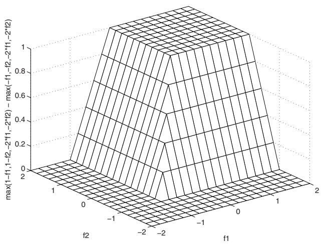
where

$$\begin{aligned} U_j^*(x_j) &= \max\left(1, M_j^*(x)\right) \\ V_j^*(x_j) &= \max\left(0, M_j^*(x)\right) \\ M_j^{\alpha}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), -f_j(x_j)\right) \\ M_j^{\beta}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), f_j(x_j)\right) \quad . \end{aligned}$$

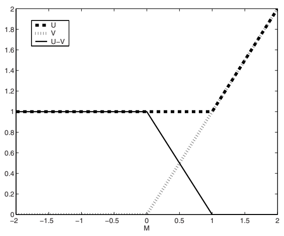
The wildcard '*' represents either α or β. The key idea of deriving Equation (3) is to use the difference of two $\max$ functions to upper bound the conjunction of indicator functions. Figure 2(a) shows an example when the conjunction consists of two indicator functions. Note that we simply bound the 0-1 function by the ramp function. Similar ramp loss functions have been used before to approximate the classification error more closely [8,9] and to improve the scalability of SVMs [10]. Figure 2(b) shows U_j^* and V_j^* as a function M_j^*.

We formulate the following optimization problem.

$$\min \sum_{i=1}^{N} \hat{L}(f(X_i), C_i) + \lambda \Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) \quad (4)$$



(a)



(b)

Fig. 2. (a) Upper bound of $I[f_1 > 0] \cdot I[f_2 > 0]$, (b) U_j^* and V_j^* as a function M_j^*

where Ω is some monotonically increasing function. The first term measures the empirical loss and the second term is the regularization term, measured with respect to a set of reproducing kernel Hilbert spaces $\{\mathcal{H}_j\}$.

4.2 Catenary Support Vector Optimization

We begin with linear classifiers $f_j(x_j) = w_j \cdot x_j + b_j$ and an ℓ_2 regularization term $\Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) = \sum_{j=1}^S \|w_j\|_2^2$. Note that U_j^* and V_j^* are all convex functions in $\{(w_1, b_1), \ldots, (w_S, b_S)\}$. But the difference of two convex functions, $U_j^*(x_j) - V_j^*(x_j)$, is non-convex. Thus, Problem (4) is not a convex optimization problem.

We re-formulate Problem (4) as the following constrained optimization problem.

$$\begin{array}{ll} \min & \sum_{i=1}^N \sum_{j=1}^S \left(\alpha_{ij}\xi_{ij}^\alpha + \beta_{ij}\xi_{ij}^\beta\right) + \lambda \sum_{j=1}^S \|w_j\|_2^2 \\ & \xi_{ij}^\alpha \geq -w_k \cdot X_{ij} - b_k - V_j^\alpha(X_i) \ \forall i, k \leq j \\ \text{s.t.} & \xi_{ij}^\beta \geq -w_k \cdot X_{ij} - b_k - V_j^\beta(X_i) \ \forall i, k < j \\ & \xi_{ij}^\beta \geq w_j \cdot X_{ij} + b_j - V_j^\beta(X_i) \quad \forall i, j \\ & \xi_{ij}^\alpha, \xi_{ij}^\beta \geq 1 \qquad\qquad\qquad\qquad \forall i, j \end{array} \tag{5}$$

Note that we have dropped the constant term, $\sum_{i=1}^N m_{i,1}$, from the objective. In principle, we can leave V_j^* in the objective. But moving V_j^* to the constraints appears to give better empirical results. Also, it is possible to have different tradeoff parameters for each classifier stage.

The Concave-Convex Procedure. The constraints of Program (5) can all be viewed as the difference of two convex functions. We employ the concave-convex procedure (CCCP), first introduced by Yuille and Rangarajan [11] to solve minimization problems whose objective function can be expressed as the sum of a convex part and a concave part. While Yuille and Rangarajan [11] considered only linear constraints, Smola et al. [12] generalized the CCCP to handle concave-convex constraints. The CCCP is an iterative procedure. In each iteration, it replaces the concave parts in the objective function and the constraints by their first-order Taylor approximation. The resulting problem is convex and can be solved using efficient convex minimization algorithms.

Consider the following optimization problem:

$$\begin{array}{ll}\min & f_0(x) - g_0(x) \\ \text{s.t.} & f_i(x) - g_i(x) \leq c_i \quad \forall i\end{array}$$

where f_i and g_i are real-valued convex and differentiable functions on $\Re^n$ for $i \in \{0, \ldots, m\}$, and $c_i \in \Re$ for $i \in \{1, \ldots, m\}$. The CCCP computes $x^{(t+1)}$ from $x^{(t)}$ by solving the following convex optimization problem.

$$\begin{array}{ll}\min & f_0(x) - \left(g_0(x^{(t)}) + \nabla g_0(x^{(t)})^T (x - x^{(t)})\right) \\ \text{s.t.} & f_i(x) - \left(g_i(x^{(t)}) + \nabla g_i(x^{(t)})^T (x - x^{(t)})\right) \leq c_i \quad \forall i\end{array}$$

It can be shown that the CCCP converges to a local minimum [12]. In case of non-global minimum, one may restart the CCCP with a different $x^{(0)}$. However, the CCCP can be considered as a special case of difference of convex functions (D.C.) programming. Tao and An [13] state that the D.C. minimization algorithm (DCA) often converges to a global minimum.

catSVM Program. To formulate Program (5) as a CCCP problem, let $\mathbf{w} = (w_1, \ldots, w_S)$ and $\mathbf{b} = (b_1, \ldots, b_S)$. In each iteration, we need to replace V_j^* in the constraints by its first-order Taylor expansion at the current estimates of $\mathbf{w}$ and $\mathbf{b}$. Notice that V_j^* are non-smooth functions. When we calculate its Taylor expansion, we use its subgradient. For the pointwise maximum function $h(x) = \max_{1 \leq i \leq m} h_i(x)$, its subdifferential at x, $\partial h(x)$, is the convex hull of the subdifferentials of the "active" functions at x, i.e., $\partial h(x) = \mathbf{H_{Convex}}\{\partial h_i(x) | h_i(x) = h(x)\}$. Thus, by simple calculus, we obtain that, for $j = 1, \ldots, S$,

$$\frac{\partial V_j^*(x; \mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = \begin{cases} \{\mathbf{0}\} & \text{if } M_j^*(x) < 0, \\ \Big\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma\tau_j x_j, \mathbf{0}) & \\ \qquad \Big| \tau_k \geq 0,\ \sum_{k=1}^j \tau_k \leq 1\Big\} & \text{if } M_j^*(x) = 0, \\ \Big\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma\tau_j x_j, \mathbf{0}) & \\ \qquad \Big| \tau_k \geq 0,\ \sum_{k=1}^j \tau_k = 1\Big\} & \text{if } M_j^*(x) > 0; \end{cases} \tag{6}$$

where

$$\tau_k = 0 \quad \begin{array}{c}\text{if } k < j \text{ and } M_j^*(x) \neq -w_k \cdot x_k - b_k \\ \text{or if } k = j \text{ and } M_j^*(x) \neq \sigma(w_k \cdot x_k + b_k)\end{array}$$

$$\sigma = \begin{cases} -1 & \text{if } * = \alpha, \\ +1 & \text{if } * = \beta. \end{cases}$$

and $\mathbf{0}$ denotes padding zeroes of appropriate length.

Similarly, we can obtain $\frac{\partial V_j*(x;\mathbf{w},\mathbf{b})}{\partial \mathbf{b}}$ by replacing x_k's by 1 in Equation (6). In the experiments, we pick the subgradient with

$$\tau_k = \begin{cases} c & \text{if } k \text{ is the largest index s.t.} \\ & \quad \text{either } k < j \text{ and } M_j^*(x) = -w_k \cdot x_k - b_k, \\ & \quad \text{or } k = j \text{ and } M_j^*(x) = \sigma(w_k \cdot x_k + b_k), \\ 0 & \text{otherwise,} \end{cases}$$

where $c = \frac{\rho}{\rho+1}$ and ρ is the number of active functions if $M_j^*(x) = 0$, and $c = 1$ if $M_j^*(x) > 0$.

Since only one of α_{ij} or β_{ij} is nonzero, we need only consider the constraints associated with one of ξ_{ij}^α or ξ_{ij}^β. The number of constraints in Program (5) is quadratic in the number of stages. We can re-write it so that the number of constraints depends linearly on the number of stages.

$$\begin{array}{ll} \min \sum_{i=1}^N \sum_{j=1}^S \left(\alpha_{ij}\xi_{ij}^\alpha + \beta_{ij}\xi_{ij}^\beta\right) + \lambda \sum_{j=1}^S \|w_j\|_2^2 & \\ \text{s.t.} \begin{array}{ll} \xi_{ij}^\alpha \geq \eta_{ij} - V_j^\alpha(X_i) & \forall i,j \\ \xi_{ij}^\beta \geq \eta_{i,j-1} - V_j^\beta(X_i) & \forall i,j \\ \xi_{ij}^\beta \geq w_j \cdot X_{ij} + b_j - V_j^\beta(X_i) & \forall i,j \\ \xi_{ij}^\alpha, \xi_{ij}^\beta \geq 1 & \forall i,j \\ \eta_{ij} \geq -w_j \cdot X_{ij} - b_j & \forall i,j \\ \eta_{ij} \geq \eta_{i,j-1} & \forall i,j \end{array} & \end{array} \tag{7}$$

4.3 Extensions

One important advantage of SVM is that it can use kernels to handle data that are not linearly separable. By the Representer Theorem [14], we can also kernelize the stage classifiers. Denote the kernel matrix for the j-th stage by K_j and its i-th column by $K_j(\cdot, i)$. The only changes to Program (7) are (i) replacing the regularization term in the objective by $\lambda \sum_j^S w_j' K_j w_j$, and (ii) replacing the feature vector X_{ij} by $K_j(\cdot, i)$. We are free to use different kernels for different stages. Also, instead of using a ℓ_2 regularization term, we may also use ℓ_1 regularization term to promote sparsity. If we do so, in each iteration of the CCCP, we need to solve a linear program instead of a quadratic program.

4.4 An Alternative Loss Bound

In the above derivation, we view the loss of an example as the sum of losses incurred in each stage and then derive a upper bound using ramp functions. This is not the only way to do it. Alternatively, we can view the loss of an example as the $\max$ of losses incurred in each stage and obtain the following loss bound:

$$\tilde{L}(f(x), c) = d_0 + \max\left(\left\{d_j(U_j^\alpha(x) - V_j^\alpha(x))\right\}_{j=1}^S, d_{S+1}(U_{S+1}^\beta(x) - V_{S+1}^\beta(x))\right) \tag{8}$$

where $d_0 = \min(c_1, \ldots, c_{S+1})$ and $d_j = c_j - d_0$.

Note that $\tilde{L}$ is no greater than $\hat{L}$. It is not difficult to see that we can formulate a constrained optimization problem with $\hat{L}$ and employ the CCCP to solve it. Unfortunately, our preliminary experimental results showed that the CCCP is not effective in optimizing $\tilde{L}$. We leave the problem of optimizing $\tilde{L}$ as an open problem.

5 Performance Bounds

We are able to provide theoretical bounds on how well catSVM will perform on new testing data. In fact, Shelton et al. [1] gave a risk bound for chained boosting and a similar bound holds for catSVM. We will state the theorem below. The proof is similar to the one in [1] and is omitted. We need the following definition.

Definition 1. *Let μ be a probability distribution on a set $\mathcal{X}$ and suppose that $X_1, \ldots, X_n$ are independent samples selected according to μ. Let F be a class of functions mapping from $\mathcal{X}$ to $\Re$. Define the random variable*

$$\hat{G}_n(F) = \mathbf{E}\left[\sup_{f \in F}\left|\frac{2}{n}\sum_{1}^{n} g_i f(X_i)\right| \,\middle|\, X_1, \ldots, X_n\right] ,$$

where $g_1, \ldots, g_n$ are independent Gaussian $N(0,1)$ random variables. The Gaussian complexity of F is $G_n(F) = \mathbf{E}\hat{G}_n(F)$.

We can now state the theorem, bounding the true risk by the empirical risk and the Gaussian complexity of the classes of the stage classifiers:

Theorem 1. *Let L and $\hat{L}$ be as in Equations (1) and (3). Let $\gamma_j = \max_{c \in \mathcal{C}}(\alpha_j + \beta_j)$ and $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. Let $F_1, \ldots, F_S$ be the sequence of the classes of the stage classifiers. Let $(X_i, C_i)_{i=1}^{N}$ be independently selected according to some fixed probability measure P. Then, for any integer N and any $0 < \delta < 1$, with probability at least $1 - \delta$ over samples of size N, every sequence $f_1, \ldots, f_S$ in $F_1 \times \ldots \times F_S$ satisfies*

$$\mathbf{E}[L] \leq \hat{\mathbf{E}}_N[\hat{L}] + \kappa \sum_{j=1}^{S}\left(\sum_{\ell=j}^{S}\gamma_\ell\right) G_N(F_j) + \Lambda\sqrt{\frac{8\ln\frac{2}{\delta}}{N}}$$

for some constant κ.

Note that the second term in the bound does not depend on the regular costs c_j's directly, and it does not depend on m_1 at all. Quite often, the incremental costs α_j's and β_j's are smaller than the c_j's. Additionally, for $j < j'$, the complexity of F_j has a larger weight than that of $F_{j'}$. This may suggest that it is advantageous to use simple stage classifiers in early stages and use complex stage classifiers in later stages. We can further bound the true risk in terms of kernel functions of the stage classifiers. We need the following lemma which follows from McDiarmid's inequality [15].

Lemma 1. *Let F be a class of functions mapping to $[-1, 1]$. For any integer n,*

$$P\left\{|G_n(F) - \hat{G}_n(F)| \geq \epsilon\right\} \leq 2\exp\left(\frac{-n\pi\epsilon^2}{4}\right) .$$

Theorem 1 and Lemma 1, combined with Lemma 22 in Bartlett and Mendelson [16], imply the following theorem.

Theorem 2. *Let L and $\hat{L}$ as in Equations (1) and (3). Let $\gamma_j = \max_{c\in\mathcal{C}}(\alpha_j + \beta_j)$ and $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. Let $F_1, \ldots, F_S$ be the sequence of the classes of stage classifiers. Let $\mathcal{X}_j$ be the feature space in the j-th stage. For $j = 1, \ldots, S$, fix B_j, and let $K_j : \mathcal{X}_j \times \mathcal{X}_j \to \Re$ be a kernel with $\sup_{x\in\mathcal{X}_j} |K_j(x, x)| < \infty$. Let $\mathcal{X}$ be the full feature space, i.e., $\mathcal{X} = \bigcup_{j=1}^{S} \mathcal{X}_j$. Suppose that $\{X_i, C_i\}_{i=1}^{N}$ are selected at random and independently according to some probability distribution P on $\mathcal{X} \times \mathcal{C}$. Then with probability at least $1 - \delta$, every function sequence $f_1, \ldots, f_S$ of the form*

$$f_j(x) = \sum_{i=1}^{N} \alpha_i K_j(x_{ij}, x)$$

with $\sum_{i_1,i_2} \alpha_{i_1}\alpha_{i_2} K_j(x_{i_1,j}, x_{i_2,j}) \le B_j^2$ satisfies

$$\mathbf{E}[L] \le \hat{\mathbf{E}}_N[\hat{L}] + \frac{\kappa}{N} \sum_{j=1}^{S} \left(\sum_{\ell=j}^{S} \gamma_\ell \right) B_j \sqrt{\sum_{i=1}^{N} K_j(x_{ij}, x_{ij})}$$
$$+ \left(\Lambda + \frac{1}{\sqrt{2\pi}} \sum_{j=1}^{S} j\gamma_j \right) \sqrt{\frac{8 \ln \frac{2(S+1)}{\delta}}{N}}$$

for some constant κ.

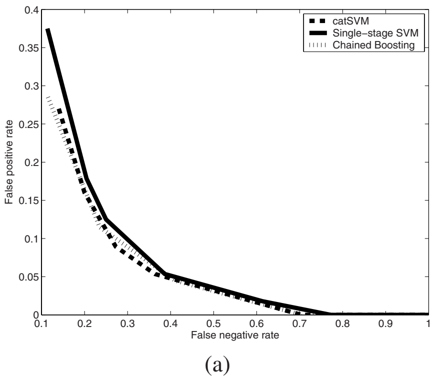
6 Experimental Results

We tested catSVM on UCI benchmark data and the MIT face database. We compared the performance of catSVM to chained boosting[1] and single-stage SVM. For chained boosting, we used decision stumps as weak classifiers and set the number of rounds before the algorithm stops to 2000. For single-stage SVM and catSVM, we used the RBF kernel and set the kernel width to the median of the pairwise distance in the training set. We set the regularization parameter λ to be 1 and did not adjust it. Furthermore, for catSVM, we initialized the SVM in every stage to be zero (i.e., for $j = 1, \ldots, S,\ w_j = \mathbf{0}$ and $b_j = 0$). We used Mosek to solve the quadratic programs generated by catSVM.

The single-stage SVM and our catSVM algorithms run on the same hypothesis space of RBF kernels. We ran chained boosting on a different feature space. It was not possible to use the same feature space. The boosting algorithm constructs a linear surface in the space of features that are decision stumps. That does not correspond to any easily constructed kernel. The feature space dimensions of the RBF kernel could be defined as the set of all kernel functions with one point as a training point. However, those dimensions have real values and the boosting algorithm is designed for thresholded features (or weak learners). However, our experience with these datasets suggests that boosting decision stumps and RBF kernels for SVMs have roughly the same performance on single-stage problems.

We did not compare to constructing three independent SVM classifiers and then connecting them in a chain; this would have required selecting the false-positive and

[1] The original chained boosting algorithm uses regular costs. We modified it to use incremental costs as well, which improved its performance.



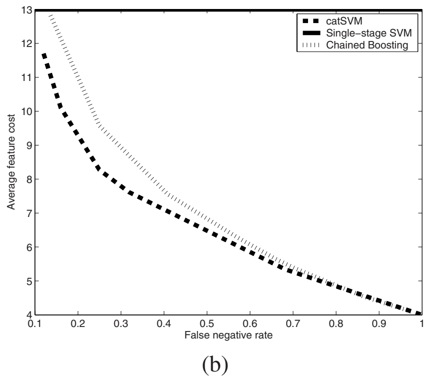


Fig. 3. UCI heart tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

Table 1. For four different cost settings for the heart dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples

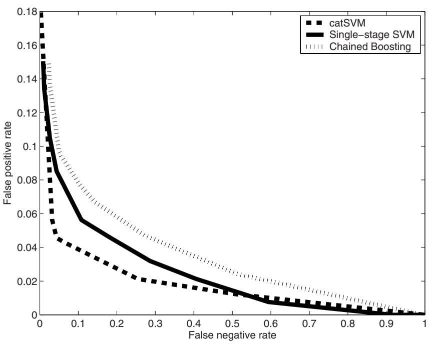
			fn: 9, fp: 18				fn: 18, fp: 18				fn: 36, fp: 18				fn: 72, fp: 18			
boosting	train	+	76	0	0	0	70	0	0	6	38	0	0	38	16	0	0	60
		–	94	0	0	0	94	0	0	0	82	8	4	0	74	4	16	0
	test	+	43	0	0	1	39	0	0	5	24	2	0	18	18	2	1	23
		–	56	0	0	0	52	1	2	1	45	5	4	2	42	7	4	3
catSVM	train	+	76	0	0	0	6	2	6	62	2	1	1	72	0	0	2	74
		–	94	0	0	0	68	3	19	4	54	8	20	12	25	20	34	15
	test	+	44	0	0	0	9	0	5	30	7	1	3	33	7	0	4	33
		–	56	0	0	0	49	1	5	6	35	5	9	7	20	9	19	8

false-negative costs for each classifier. One of the main purposes of our approach is to automatically adjust the classifier to achieve the desired cost results without having to manually search over such trade-off parameters.

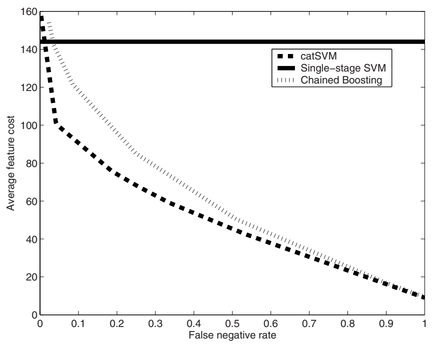
We construct the vectors of stage costs as follows. We assign a constant feature cost to each stage that is the same for all examples (as specified in the problem set up below). It represents the cost of collecting the features, regardless of the final outcome. If the example is positive, we add an extra cost to c_i for $i \leq s$, representing an extra penalty if a positive example is rejected at any stage. If the example is negative, we add an extra cost to c_{s+1}, that is we penalize the classifier if it allows the example to pass through every stage (and therefore wrongly accepts it as a positive example).

6.1 UCI Data

We report the results on the heart disease dataset from the UCI machine learning repository. We assigned the 13 attributes to three stages in descending order according to



(a)



(b)

Fig. 4. Face detection tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

Table 2. For four different cost settings for the face detection dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples

			fn: 150, fp: 250				fn: 250, fp: 250				fn: 500, fp: 250				fn: 1000, fp: 250			
boosting	train	+	224	0	0	0	224	0	0	0	220	0	0	4	33	0	0	191
		–	376	0	0	0	376	0	0	0	375	1	0	0	342	34	0	0
	test	+	341	0	0	1	341	0	0	1	342	4	0	36	143	32	1	166
		–	654	1	3	0	653	1	3	1	646	4	4	4	533	88	19	18
catSVM	train	+	224	0	0	0	35	8	0	181	0	0	1	223	0	0	0	224
		–	376	0	0	0	278	86	10	2	104	233	36	3	96	240	37	3
	test	+	342	0	0	0	42	16	6	278	1	10	7	324	1	10	7	324
		–	658	0	0	0	457	143	28	30	170	392	63	33	152	410	62	34

their correlation with the predicted output: four attributes to each of the first two stages and five to the last stage. The single-stage SVM was trained and tested on all the 13 attributes. We set the feature cost of each stage to the number of attributes assigned to that stage and all the preceding stages. Early rejections are treated as "normal" whereas an example that passes all stages is treated as "disease." Figure 3(a) shows the false negatives and false positives as the misclassification penalties vary. The three methods give very similar tradeoff between the two types of errors. Figure 3(b) shows the false negative and the average feature cost as the misclassification penalties vary. The average feature cost is the average number of features that must be examined before the classifier makes a decision. The feature cost of single-stage SVM is fixed at 13. We observe that catSVM does a better job in trading false negative rate for feature cost than chained boosting.

Table 1 shows at which stage the examples are rejected or accepted. The feature costs are 4, 8, 13, and 13. 'fn: 9, fp: 18' means the penalties for false negative and false positive are 9 and 18, respectively; therefore the cost vectors would be $[13, 17, 22, 13]$

and $[4, 8, 13, 31]$ for positive and negative examples, respectively. Note that for every penalty setting, the first three columns are the number of examples rejected in the three stages and the last column is the number of examples accepted. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples.

6.2 Face Detection

We also validated the catenary SVM by applying it to face detection. We tested on the MIT face database [17] which contains 19-by-19 gray-scale images of faces and non-faces. For face detection, the non-face is usually the majority. Therefore, our goal is to produce a classifier that can identify non-face images by examining as low a resolution patch as possible. We built a multi-stage detection system where any early rejection is labeled as a non-face. The first stage looks at down-sampled versions of the images at a resolution of 3-by-3. The next stages do the same, at resolutions of 6-by-6 and 12-by-12. We did not examine the full 19-by-19 resolution as it did not provide significant improvement over the 12-by-12 resolution.

We assign a feature cost to each stage proportional to the total number of pixels at that stage and all the preceding stages. There are three free parameters in the problem formulation: the per pixel cost, the penalty for an incorrect face classification, and the penalty for an incorrect non-face classification. Changing these quantities will control the tradeoff between false negatives and false positives, and between classification error and feature cost. In the experiments, we fix the per pixel cost and vary the other two quantities.

We used 600 images as the training set and 1000 images as the testing set. The single-stage SVM was trained and tested on image patches at the highest resolution, 12-by-12. Figure 4(a) shows the false negatives and false positives as the misclassification penalties vary. Note that catSVM can achieve better tradeoff than single-stage SVM and chained boosting. The processing pipeline successfully improves the ability of SVM to tradeoff between the two types of errors. Figure 4(b) shows the false negative and the average feature cost as the misclassification penalties vary. The feature cost of single-stage SVM is fixed at 144. Chained boosting and catSVM give higher average feature costs for lower false negative rates. Note that catSVM requires a lower average feature cost than chained boosting for most false negative rates. The advantage of catSVM becomes more obvious when the false negative rate is small.

Table 2 shows at which stage the examples are rejected or accepted. The feature costs are 9, 45, 189, and 189. ‘fn: 150, fp: 250’ means the penalties for false negative and false positive are 150 and 250, respectively. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples. It is clear that catSVM is more effective in pushing the positive examples forward.

It is interesting to note that the performance of catSVM is superior to that of a single-stage SVM (which is a regular SVM trained on the full set of features, varying the false-positive and false-negative costs) in terms of testing error. We believe this is because the hypothesis class of the earlier stages are simpler. Therefore, those decision rules have less variance for a fixed number of samples. Our algorithm then has a natural bias that helps reduce the variance with few numbers of samples. Our generalization bounds also point to this advantage.

7 Conclusion

We believe that for some decision-making problems, it is important to weigh the benefit against the cost of acquiring more information. We present the catenary SVM to solve one-sided early detection for binary classification. We formulate the problem as a constrained concave-convex optimization problem and solve it using CCCP. In addition, we are able to provide data-dependent theoretical guarantee for catSVM. The experimental results show that catSVM can tradeoff misclassification error and feature cost more effectively than single-stage SVM and chained boosting.

The main drawback of catSVM is its scalability. Currently, we use a generic solver to solve the linear or quadratic programs generated by CCCP. Although the number of constraints is only linear in the number of stages, the generic solver runs out of memory even for medium-size datasets. We are planning to explore other more scalable algorithms (e.g., cutting-plane methods). Moreover, it would be interesting to extend catSVM to two-sided early detection.

Acknowledgments

This research was supported through the grant “Adaptive Decision Making for Silicon Wafer Testing” from Intel Research and UC MICRO. We thank the anonymous reviewers for their careful and thoughtful comments.

References

1. Shelton, C.R., Huie, W., Kan, K.F.: Chained boosting. In: NIPS 2007, pp. 1281–1288 (2007)
2. Howard, R.A., Matheson, J.E.: Influence diagrams. In: Howard, R.A., Matheson, J.E. (eds.) Readings on the Principles and Applications of Decision Analysis. Strategic Decision Group, vol. 2, pp. 719–762 (1984) article dated 1981
3. Viola, P., Jones, M.: Fast and robust classification using asymmetric adaboost and a detector cascade. In: NIPS, pp. 1311–1318 (2002)
4. Wu, J., Mullin, M.D., Rehg, J.M.: Linear asymmetric classifier for cascade detectors. In: ICML, pp. 988–995 (2005)
5. Šochman, J., Matas, J.: WaldBoost — learning for time constrained sequential detection. In: CVPR, pp. 150–156 (2005)
6. Bi, J., Periaswamy, S., Okada, K., Kubota, T., Fung, G., Salganicoff, M., Rao, B.: Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. In: SIGKDD, pp. 837–844 (2006)
7. Dundar, M., Bi, J.: Joint optimization of cascaded classifiers for computer aided detection. In: CVPR, pp. 1–8 (2007)
8. Shen, X., Tseng, G.C., Zhang, X., Wong, W.H.: On ψ-learning. Journal of the American Statistical Association 98(463), 724–734 (2003)
9. Liu, Y., Shen, X., Doss, H.: Multicategory ψ-learning and support vector machine: computational tools. J. of Comp. and Graphical Statistics 14(1), 219–236 (2005)

10. Collobert, R., Weston, J., Bottou, L.: Trading convexity for scalability. In: ICML, pp. 201–208 (2006)
11. Yuille, A.L., Rangarajan, A.: The concave-convex procedure (CCCP). In: NIPS, pp. 1033–1040 (2002)
12. Smola, A.J., Vishwanathan, S., Hofmann, T.: Kernel methods for missing variables. In: AI-STATS, pp. 325–332 (2005)
13. Tao, P.D., An, L.T.: A D.C. optimization algorithm for solving the trust-region subproblem. SIAM Journal on Optimization 8(2), 476–505 (1998)
14. Schölkopf, B., Smola, A.J.: Learning with kernels. The MIT Press, Cambridge (2002)
15. McDiarmid, C.: On the method of bounded differences. Surveys in Combinatorics, 148–188 (1989)
16. Bartlett, P.L., Mendelson, S.: Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research 2, 463–482 (2002)
17. MIT: CBCL face database #1 (2000), `http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html`