Journal of Machine Learning Research 3 (2002) 303-321 Submitted 11/01; Published 11/02

Minimal Kernel Classifiers

Glenn M. Fung GFUNG@CS.WISC.EDU
Olvi L. Mangasarian OLVI@CS.WISC.EDU
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706, USA

Alexander J. Smola ALEX.SMOLA@ANU.EDU.AU
Australian National University
Department of Engineering and RSISE
Canberra, ACT 0200, Australia

Editor: John Shawe-Taylor

Abstract

A finite concave minimization algorithm is proposed for constructing kernel classifiers that use a minimal number of data points both in generating and characterizing a classifier. The algorithm is theoretically justified on the basis of linear programming perturbation theory and a leave-one-out error bound as well as effective computational results on seven real world datasets. A nonlinear rectangular kernel is generated by systematically utilizing as few of the data as possible both in training *and* in characterizing a nonlinear separating surface. This can result in substantial reduction in kernel data-dependence (over 94% in six of the seven public datasets tested on) and with test set correctness equal to that obtained by using a conventional support vector machine classifier that depends on many more data points. This reduction in data dependence results in a much faster classifier that requires less storage. To eliminate data points, the proposed approach makes use of a novel loss function, the "pound" function $(\cdot)_\#$, which is a linear combination of the 1-norm and the step function that measures both the magnitude and the presence of any error.

Keywords: Support Vector Machines, Sparse Kernels, Data Reduction, Concave Minimization

1. Introduction

Support vector machines have come to play a very dominant role in data classification using a kernel-based linear or nonlinear classifier (Vapnik, 2000, Cherkassky and Mulier, 1998, Schölkopf et al., 1999, Smola et al., 2000). One of the main difficulties that confront large data classification by a nonlinear kernel is the possible dependence of the nonlinear separating surface on the entire dataset. This creates unwieldy storage and computational problems that may preclude the direct use of nonlinear kernels for large datasets or in applications where a very fast classifier is required such as in on-line credit card fraud detection.

For example, let $A \in R^{1000\times 10}$ represent a thousand-point dataset with 10 features characterizing each point. Then the above two difficulties translate into a nonlinear kernel matrix of size 1000×1000 with a million entries that leads to a dense nontrivial mathematical program. Even after solving this problem, the resulting nonlinear separating surface can potentially depend on most of the 1000 points that need to be stored and used in each classification of a new point. Our minimal kernel classifier method completely addresses this difficulty by generating a nonlinear kernel-based classifier that reduces the classifier's data dependence by as much as 98.8%, compared to a conventional support vector machine classifier. In other words, the classifier depends on a very small number of kernel functions.

Such "minimum description length" models (Rissanen, 1986, Blumer et al., 1987, Mitchell, 1997, p. 66), that depend on much fewer data points, often generalize as well or better than models that depend on many more data points, and are useful for incremental and chunking algorithms (Bradley and Mangasarian, 2000, Mangasarian and Musicant, 2002) for massive datasets. In order to address the problem of solving huge mathematical programs, we use RSVM (Lee and Mangasarian, 2000) to generate an initial reduced rectangular kernel that reduces dramatically the size of the problems to be solved while preserving and often, improving training and testing set correctness.

We briefly outline the paper now. In Section 2 we describe linear and nonlinear kernel classifiers and how to generate them using a linear programming formulation. In Section 3 we derive leave-one-out-correctness (*looc*) and leave-one-out-error (*looe*) error bounds that can be obtained by solving a single linear program that generates the nonlinear kernel classifier. In Section 4, using these error bounds we propose a concave programming formulation that utilizes a novel loss function $x_{\#}$ depicted in Figure 2 that measures both the magnitude of the error and its presence. The discontinuous concave minimization program is solved by first smoothing it into a differentiable concave program and then using a finite linear-programming-based successive linear approximation to solve it. Section 5 describes our numerical test results on seven public datasets. These results indicate a major reduction in the number of data points required to generate a rectangular kernel classifier without any loss of testing set correctness. Section 6 concludes the paper.

We briefly describe our notation now and give some background material. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. For a vector x in the n-dimensional real space R^n, the step function x_* is defined as $(x_*)_i = 1$ if $x_i > 0$ else $x_i = 0$, $i = 1,\ldots,n$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. The p-norm of x, $(\sum_{i=1}^{n}|x_i|^p)^{\frac{1}{p}}$, will be denoted by $\|x\|_p$, for $1 \le p < \infty$, and the ∞-norm, $\max_{1\le i\le n}|x_i|$, will be denoted by $\|x\|_\infty$. For $1 \le p, q \le \infty$ and $\frac{1}{p}+\frac{1}{q} = 1$, $\|x\|_p$ and $\|x\|_q$ are dual norms. For a matrix $A \in R^{m\times n}$, A_i is the ith row of A which is a row vector in R^n. A column vector of ones of arbitrary dimension will be denoted by e. For $A \in R^{m\times n}$ and $B \in R^{n\times l}$, the kernel $K(A, B)$ maps $R^{m\times n} \times R^{n\times l}$ into $R^{m\times l}$. In particular, if x and y are column vectors in R^n then, $K(x', y)$ is a real number, $K(x', A')$ is a row vector in R^m and $K(A, A')$ is an $m \times m$ matrix. If $\bar{A}$ is an $\bar{m} \times n$ submatrix of A, then the rectangular $m \times \bar{m}$ matrix $K(A, \bar{A}')$ is referred to as a *rectangular kernel*. The base of the natural logarithm will be denoted by ε. For a vector $z \in R^\ell$, card(z_0) will denote the cardinality of (*i.e.* the number elements in) the set

$\{i \mid z_i = 0\}$ while card(z_+) will denote the cardinality of the set $\{i \mid |z_i| > 0\}$. The symbol $:=$ will denote a definition.

We remark that our notation is exemplified by the nonlinear separating surface (9) and the corresponding linear programming formulation (10) which is consistent with a large body of previous work such as (Ferris and Munson, 2000, Fung and Mangasarian, 2000, 2001, 2002, Lee and Mangasarian, 2000, 2001, Mangasarian, 2000, Mangasarian and Musicant, 2000, 2001a,b). We relate it to the notation used by others (Vapnik, 2000, Cherkassky and Mulier, 1998, Cristianini and Shawe-Taylor, 2000, Schölkopf et al., 1999) as follows. Our matrix A is the matrix X, our kernel $K(A, A')$ is $k(X, X')$, our diagonal matrix D is the diagonal matrix Y, our error vector y is the error vector ξ, and our parameter ν is the parameter C.

2. Linear and Nonlinear Kernel Classification

We consider the problem of classifying m points in the n-dimensional real space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the classes +1 or -1 as specified by a given $m \times m$ diagonal matrix D with ones or minus ones along its diagonal. For this problem the standard support vector machine with a linear kernel AA' (Vapnik, 2000, Cherkassky and Mulier, 1998) is given by the following quadratic program for some $\nu > 0$:

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \nu e'y + \frac{1}{2}w'w & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0. \end{array} \tag{1}$$

As depicted in Figure 1, w is the normal to the bounding planes:

$$\begin{array}{ccccc} x'w & - & \gamma & = & +1 \\ x'w & - & \gamma & = & -1, \end{array} \tag{2}$$

and γ determines their location relative to the origin. The first plane above bounds the class +1 points and the second plane bounds the class -1 points when the two classes are strictly linearly separable, that is when the slack variable $y = 0$. The linear separating surface is the plane

$$x'w = \gamma, \tag{3}$$

midway between the bounding planes (2). If the classes are linearly inseparable then the two planes bound the two classes with a "soft margin" determined by a nonnegative slack variable y, that is:

$$\begin{array}{ccccccccl} x'w & - & \gamma & + & y_i & \geq & +1, & \text{for } x' = A_i \text{ and } D_{ii} = +1, \\ x'w & - & \gamma & - & y_i & \leq & -1, & \text{for } x' = A_i \text{ and } D_{ii} = -1. \end{array} \tag{4}$$

The 1-norm of the slack variable y is minimized with weight ν in (1). The quadratic term in (1), which is twice the reciprocal of the square of the 2-norm distance $\frac{2}{\|w\|_2}$ between the two bounding planes of (2) in the n-dimensional space of $w \in R^n$ for a *fixed* γ, maximizes that distance, often called the "margin". Figure 1 depicts the points represented by A, the bounding planes (2) with margin $\frac{2}{\|w\|_2}$, and the separating plane (3) which separates $A+$, the points represented by rows of A with $D_{ii} = +1$, from $A-$, the points represented by rows of A with $D_{ii} = -1$.

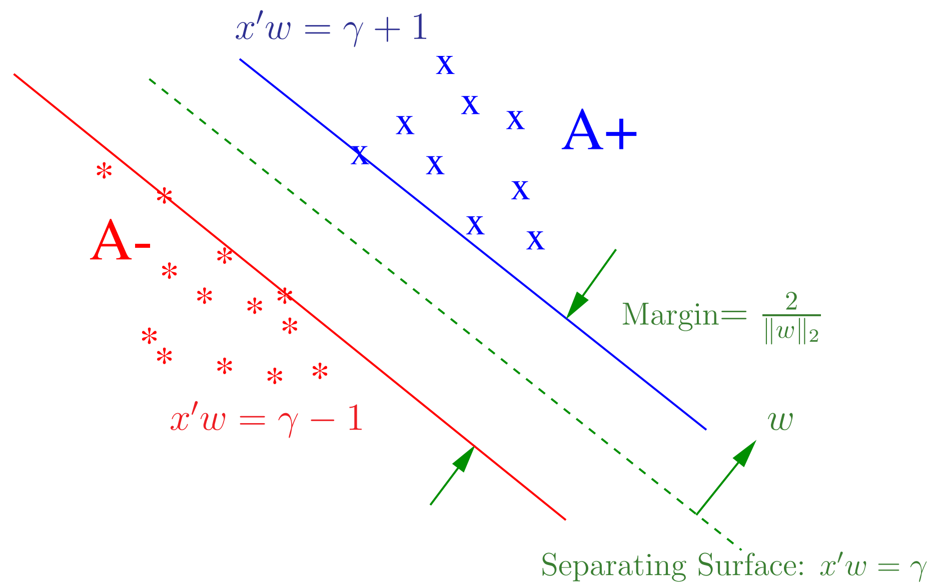

Figure 1: **The bounding planes (2) with margin $\frac{2}{\|w\|_2}$, and the plane (3) separating $A+$, the points represented by rows of A with $D_{ii} = +1$, from $A-$, the points represented by rows of A with $D_{ii} = -1$.**

In order to make use of fast polynomial-time linear-programming-based approaches (Ye, 1997) instead of the standard quadratic programming formulation (1) we reformulate (1) by replacing the 2-norm by a 1-norm as follows (Bradley and Mangasarian, 1998, Mangasarian, 2000):

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \nu e'y + \|w\|_1 & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0. \end{array} \tag{5}$$

This reformulation in effect maximizes the margin, the distance between the two bounding planes of Figures 1, using the different ∞-norm and results with a margin in terms of the 1-norm, $\frac{2}{\|w\|_1}$, instead of $\frac{2}{\|w\|_2}$ (Mangasarian, 1999a). The margin between the bounding planes (2) using a p-norm, is given by $\frac{2}{\|w\|_q}$ (Mangasarian, 1999a), where $\frac{1}{p} + \frac{1}{q} = 1$, that is the p-norm and q-norm are dual norms for $1 \leq p,\, q \leq \infty$. The mathematical program (5) is easily converted to a linear program as follows:

$$
\begin{array}{lrcl}
\min\limits_{(w,\gamma,y,v)\in R^{n+1+m+n}} & \nu e'y + e'v & & \\
\text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\
& v \geq \quad w & \geq & -v \\
& y & \geq & 0,
\end{array}
\tag{6}
$$

where, at a solution, v is the absolute value $|w|$ of w.

To obtain a more general formulation of an SVM, we make a transformation of the variable w as: $w = A'Du$, where u can be interpreted as an arbitrary variable in R^m (which can be motivated by duality theory (Mangasarian, 2000)), then the SVM (5) becomes:

$$
\begin{array}{lrcl}
\min\limits_{(u,\gamma,y)\in R^{2m+1}} & \nu e'y + \|A'Du\|_1 & & \\
\text{s.t.} & D(AA'Du - e\gamma) + y & \geq & e \\
& y & \geq & 0.
\end{array}
\tag{7}
$$

If we now replace the linear kernel AA' by a completely general kernel $K(A, A')$ and replace $\|A'Du\|_1$ by a general convex function of u, we obtain the generalized support vector machine (GSVM) of (Mangasarian, 2000) as follows:

$$
\begin{array}{lrcl}
\min\limits_{(u,\gamma,y)\in R^{2m+1}} & \nu e'y + f(u) & & \\
\text{s.t.} & D(K(A, A')Du - e\gamma) + y & \geq & e \\
& y & \geq & 0.
\end{array}
\tag{8}
$$

The variables $(u, \gamma) \in R^{m+1}$ define the nonlinear separating surface (9) below, through the kernel K. The function $f(u)$ is some convex function on R^m which suppresses components of the parameter u, while ν is some positive number that weights the classification error $e'y$ versus the suppression of u. Suppressing components of u is justified by the leave-one-out-error bound (13) below. A solution of the mathematical program (8) for u and γ leads to the nonlinear separating surface (Mangasarian, 2000):

$$K(x', A')Du = \gamma \tag{9}$$

The linear formulation (1) of Section 2 is obtained if we let $K(A, A') = AA'$, $w = A'Du$ and $f(u) = \frac{1}{2}u'DAA'Du$.

Setting $f(u)$ in (8) equal to $||u||_1$ leads to the following linear programming formulation:

$$
\begin{array}{lrcl}
\min\limits_{(u,\gamma,y,v)\in R^{m+1+m+m}} & \nu e'y + e'v & & \\
\text{s.t.} & D(K(A, A')Du - e\gamma) + y & \geq & e \\
& v \geq \quad u & \geq & -v \\
& y & \geq & 0.
\end{array}
\tag{10}
$$

The dual (Mangasarian, 1994, p. 130) of this linear program is:

$$
\begin{array}{lrrrcr}
\max\limits_{(t,r,s)\in R^{m+n+n}} & e't & & & & \\
\text{s.t.} & DK(A, A')'Dt & -r & +s & = & 0 \\
& -e'Dt & & & = & 0 \\
& t & & & \leq & \nu e \\
& & r & +s & = & e \\
& t, & r, & s & \geq & 0
\end{array}
\tag{11}
$$

Note that, for any standard simplex algorithm (Dantzig, 1963), solution of either of the dual linear programs (10) or (11), automatically generates a solution of the other program at the termination of the algorithm.

We next derive our leave-one-out-correctness and leave-one-out-error bounds in terms of a solution to the above linear programs. Note that solving a *single* linear program (10) yields these bounds.

3. Leave-One-Out-Correctness (*looc*) and Leave-One-Out-Error (*looe*) Bounds

In this section we derive a lower bound on the leave-one-out-correctness (*looc*) of a solution to a support vector machine with a nonlinear kernel as well as an upper bound on the leave-one-out-error (*looe*), where $looc + looe = 1$. Our bounds, similar to those of (Vapnik and Chapelle, 2000, Opper and Winther, 2000, Weston and Herbrich, 2000, Jaakkola and Haussler, 1999), are however easily and directly computable from a solution of the linear-programming-based formulation of the support vector machine formulation (10) above and are used as a justification for our concave minimization algorithm for a minimal kernel classifier.

Proposition 3.1 Leave-One-Out-Correctness (*looc*) & Leave-One-Out-Error (*looe*) Bounds *Let (u, γ, y, v) be a solution of the linear program (10) and let (t, r, s) be a corresponding solution of its dual (11). The leave-one-out-correctness looc is bounded below as follows:*

$$looc \;\geq\; \frac{card((t \wedge u)_0)}{m} \tag{12}$$

and the leave-one-out-error looe is bounded above as follows:

$$looe \;\leq\; \frac{card((t \vee u)_+)}{m}, \tag{13}$$

where $card((t \wedge u)_0)$ denotes the cardinality of the set $\{i \mid t_i = 0 \text{ and } u_i = 0\}$, while $card((t \vee u)_+)$ denotes the cardinality of the set $\{i \mid t_i > 0 \text{ or } u_i \neq 0\}$.

Proof By the Karush-Kuhn-Tucker optimality conditions for (10) (Mangasarian, 1994, p. 94) we have that:

$$\begin{array}{lcl} t'(D(K(A,A')Du - e\gamma) + y - e) & = & 0 \\ t & \geq & 0 \\ D(K(A,A')Du - e\gamma) + y - e & \geq & 0 \\ y'(\nu e - t) & = & 0 \\ y & \geq & 0 \\ \nu e - t & \geq & 0. \end{array} \tag{14}$$

These optimality conditions lead to the following implications for $i = 1, \ldots, m$:

$$\begin{array}{llcl} y_i > 0 & & & \\ \Longrightarrow & t_i & = & \nu > 0 \\ \Longrightarrow & D_{ii}(K(A_i, A')Du - \gamma) - 1 & = & -y_i < 0. \end{array} \tag{15}$$

Thus, a positive y_i implies a positive multiplier $t_i = \nu > 0$ and a corresponding support vector A_i . Hence the number of support vectors equals or exceeds card(y_+), the number of positive components of y.

To establish (12) we observe that all data points A_i for which *both* the corresponding $t_i = 0$ (*i.e.* A_i is not a support vector) *and* $u_i = 0$ (*i.e.* $K(A, A_i')D_{ii}u_i = K(A, A_i')D_{ii}\cdot 0 = 0$), can be thrown out of the linear program (10) without changing the solution. For all such A_i we have by (14) that $y_i = 0$ and hence these A_i are correctly classified points, and if they were left out of the linear program (10) they would be correctly classified by the linear programming solution. Hence the leave-one-out correctness can be bounded below (because there could be other correctly classified A_i that could be thrown out also without changing the solution) by the cardinality of A_i for which both $t_i = 0$ and $u_i = 0$, that is:

$$looc \;\geq\; \frac{\text{card}((t \wedge u)_0)}{m}, \tag{16}$$

which is the bound (12). Since $looc + looe = 1$ and

$$\text{card}((t \wedge u)_0) + \text{card}((t \vee u)_+) = m,$$

it follows from (16) that

$$1 - looe \;\geq\; \frac{\text{card}((t \wedge u)_0)}{m} = \frac{m - \text{card}((t \vee u)_+)}{m} = 1 - \frac{\text{card}((t \vee u)_+)}{m},$$

from which follows the bound (13).□

Motivated by the above bound (13) on the *looe* and by linear programming perturbation theory (Mangasarian and Meyer, 1979), we present a minimal kernel algorithm that can obtain striking test set correctness results with a minimal use of data points as well as kernel components.

4. The Minimal Kernel Problem Formulation & Algorithm

By using an argument based on a finite perturbation of the objective function of a linear program, we look for solutions of the linear program (10), which in general has multiple solutions, that in addition suppress simultaneously as many components of the error of the primal variable u as well as the dual variable t. Empirical evidence (Bradley and Mangasarian, 1998) indicates that linear-programming-based classification problems are amenable to feature suppression. Since we do not want to solve both the primal and dual problems explicitly, which would be prohibitively costly for very large problems, we propose suppressing components of the error vector y as a reasonable surrogate for suppressing multiplier components t. The justification for this is that from the implication (15) we have that:

$$\{i \mid y_i > 0\} \;\subseteq\; \{i \mid t_i > 0\}.$$

Hence:

$$\frac{\text{card}((t \vee u)_+)}{m} \;\geq\; \frac{\text{card}((y \vee u)_+)}{m} = \frac{\text{card}((y \vee v)_+)}{m}, \tag{17}$$

where card$((y \vee u)_+)$ denotes the cardinality of the set $\{i \mid y_i > 0 \text{ or } u_i \neq 0\}$, card$((y \vee v)_+)$ denotes the cardinality of the set $\{i \mid y_i > 0 \text{ or } v_i > 0\}$, and the equality above follows from the fact that $v = |u|$ at a solution of the linear program (10). We propose to minimize the last term in (17) above instead of the actual upper bound on the *looe* given by the first term in (17). *This works remarkably well in reducing both data points needed and kernel size.* Consequently, we perturb the objective function of (10) by a step function $(\cdot)_*$ of y and of $v = |u|$, thus suppressing as many components of these variables as possible. This leads to the following minimization problem with a concave objective function on its feasible region:

$$\begin{array}{rcl} \displaystyle\min_{(u,\gamma,y,v)\in R^{m+1+m+m}} \nu e'y + e'v + \mu(\nu e'y_* + e'v_*) & = & \nu e'y_\# + e'v_\# \\ \text{s.t. } D(K(A,A')Du - e\gamma) + y & \geq & e \\ v \geq u & \geq & -v \\ y & \geq & 0. \end{array} \tag{18}$$

Here, the "pound" function $(\cdot)_\#$ is defined as the following loss function in terms of the step function $(\cdot)_*$:

$$x_\# = |x| + \mu|x|_*, \text{ for some } \mu > 0. \tag{19}$$

Figure 2 depicts the shape of this loss function $x_\#$, which not only penalizes the amount of deviation from zero, but also *any* deviation from zero no matter how small by an initial penalty of μ which progresses linearly with the amount of deviation thereafter. Using linear programming perturbation theory we can state the following result that justifies our minimal kernel classifier algorithm.

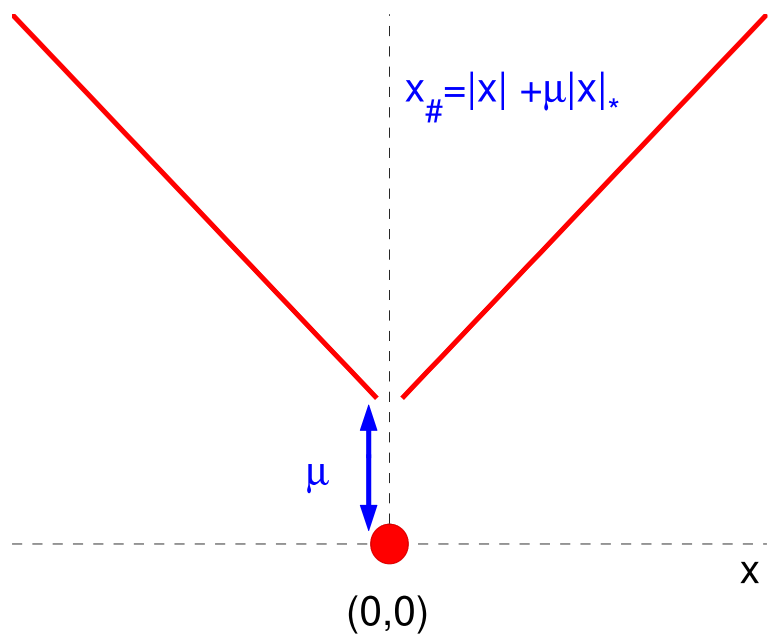

Figure 2: **The loss function $x_\#$**

Proposition 4.1 Minimal Kernel as Perturbed LP *For a given $\nu > 0$ there exist $\bar{\mu} > 0$, such that for all $\mu \in (0, \bar{\mu}]$, each solution of (18) is a solution of the linear programming* *kernel problem (10) with a minimal number of nonzero components of y and u among all its solutions.*

Proof Without loss of generality we assume that the feasible region of the linear program has no lines going to infinity in both directions and forgo the change of variables $w = \tilde{w} - e\zeta,\ \gamma = \tilde{\gamma} - \zeta,\ (\tilde{w}, \tilde{\gamma}, \zeta) \geq 0$. If we make this transformation then there are no lines in the feasible region that go to infinity in both directions, because all the variables would be nonnegative then.

Since the objective function of the problem (18) is concave on the feasible region and bounded below by zero it must have a vertex solution (Rockafellar, 1970, Corollaries 32.3.3 and 32.3.4) for all nonnegative values of ν and μ.

Fix ν at some positive value and, to simplify notation and get the basic ideas across with as little detail as possible, let the concave program (18) be represented as follows:

$$\min_{z \in S} f(z) + \mu g(z), \tag{20}$$

where $z = (u, \gamma, y, v)$, S is the polyhedral feasible region of (18), f is linear and g is concave on S, that is:

$$f(z) := \nu e'y + e'v, \quad g(z) := \nu e'y_* + e'v_*.$$

Since the S has a finite number of vertices, for some decreasing sequence of positive numbers $\{\mu_0, \mu_1, \ldots, \} \downarrow 0$, some fixed vertex $\bar{z}$ of S will repeatedly solve the concave minimization problem (18). Hence for any $\mu \in (0, \mu_0]$, $\bar{z}$ is also a solution of (18), because:

$$\mu = (1-\lambda)\mu_i + \lambda\mu_{i+1}, \text{ for some } i \text{ and some } \lambda \in [0,1],$$

and

$$f(\bar{z}) + \mu g(\bar{z}) = (1-\lambda)(f(\bar{z}) + \mu_i g(\bar{z})) + \lambda(f(\bar{z}) + \mu_{i+1} g(\bar{z})).$$

Hence $\bar{z}$ solves (20) for $\mu \in (0, \mu_0]$. We will now show that $\bar{z}$ also solves:

$$\min_{z \in S} f(z), \tag{21}$$

which is equivalent to the linear program (10). Define

$$\bar{f} := \min_{z \in S} f(z) \ \geq \ 0.$$

Suppose now that $f(\bar{z}) > \bar{f}$ and exhibit a contradiction. This will establish that $\bar{z}$ solves (21). Let

$$\epsilon := \frac{f(\bar{z}) - \bar{f}}{4} > 0 \text{ and } z \in S \text{ with } f(z) < \bar{f} + \epsilon, \tag{22}$$

and choose μ such that

$$\frac{1}{\mu} > \max\left\{\frac{g(z) - g(\bar{z})}{f(\bar{z}) - \bar{f} - 2\epsilon}\ ,\ \frac{1}{\mu_0}\right\}. \tag{23}$$

We then have the following contradiction:

$$\mu g(z) + \bar{f} + \epsilon \ > \ \mu g(z) + f(z) \ \geq \ \mu g(\bar{z}) + f(\bar{z}) \ > \ \mu g(z) + \bar{f} + 2\epsilon,$$

where the first inequality follows from the last inequality of (22), the second inequality from the fact that $\bar{z}$ is a solution of (20), and the last inequality from (23). Hence $\bar{z}$ solves (21) which is equivalent to the linear program (10).

It remains to show that $\bar{z}$ also minimizes $g(z)$ as well over the set of minimizers of f on S. That is, we have picked among all solutions of the linear program (10) that one with a minimal number of nonzero components of y and u.

Let z be any solution of (21), that is $z \in S$ and $f(z) = f(\bar{z})$. Then,

$$\mu_0 g(z) = \mu_0 g(z) + f(z) - f(z) \;\geq\; f(z) + \mu_0 g(z) - f(\bar{z}) \;\geq f(\bar{z}) + \mu_0 g(\bar{z}) - f(\bar{z}) = \mu_0 g(\bar{z}).$$

Hence,

$$\bar{z} \in \arg\min_{z \in S}\{g(z) \mid f(z) \;\leq\; f(\bar{z}) = \min_{z \in S} f(z)\}.\Box$$

Because of the discontinuity of the term $e'y_{\#}$, we approximate it by a concave exponential on the nonnegative real line. For $x \geq 0$, we approximate $x_{\#}$ of (19) by the concave exponential depicted in Figure 3. Thus for a vector $y \geq 0$:

$$y_{\#} \approx y + \mu(e - \varepsilon^{-\alpha y}), \;\; \alpha > 0, \tag{24}$$

where ε is the base of natural logarithms. This leads to the following smooth reformulation of problem (18):

$$\begin{array}{rl} \displaystyle\min_{(u,\gamma,y,v) \in R^{m+1+m+m}} & \nu e'(y + \mu(e - \varepsilon^{-\alpha y})) + e'(v + \mu(e - \varepsilon^{-\alpha v})) \\ & \begin{array}{rcl} \text{s.t. } D(K(A, A')Du - e\gamma) + y & \geq & e \\ v \geq u & \geq & -v \\ y & \geq & 0. \end{array} \end{array} \tag{25}$$

It can be shown (Bradley et al., 1998a, Theorem 2.1) that an exact solution to the original discontinuous problem (18) can be obtained by solving the above smooth problem (24) for any sufficiently large value of α, which is typically set to 5 in our numerical computations.

We now prescribe the highly effective and finite successive linearization algorithm (SLA) (Mangasarian, 1996, Bradley et al., 1998b, 1997, 1998a, Mangasarian, 1999b) for solving the above problem.

Algorithm 4.2 Minimal Kernel Algorithm *Start with an arbitrary $(u^0, \gamma^0, y^0, v^0)$. Having $(u^i, \gamma^i, y^i, v^i)$ determine the next iterate $(u^{i+1}, \gamma^{i+1}, y^{i+1}, v^{i+1})$ by solving the following linear program:*

$$\begin{array}{rl} \displaystyle\min_{(u,\gamma,y,v) \in R^{m+1+m+m}} & \nu(e + \mu\alpha\varepsilon^{-\alpha y^i})'(y - y^i) + (e + \mu\alpha\varepsilon^{-\alpha v^i})'(v - v^i) \\ & \begin{array}{rcl} s.t.\ D(K(A, A')Du - e\gamma) + y & \geq & e \\ v \geq u & \geq & -v \\ y & \geq & 0. \end{array} \end{array} \tag{26}$$

Stop when:

$$\nu(e + \mu\alpha\varepsilon^{-\alpha y^i})'(y^{i+1} - y^i) + (e + \nu\alpha\varepsilon^{-\alpha v^i})'(v^{i+1} - v^i) = 0. \tag{27}$$

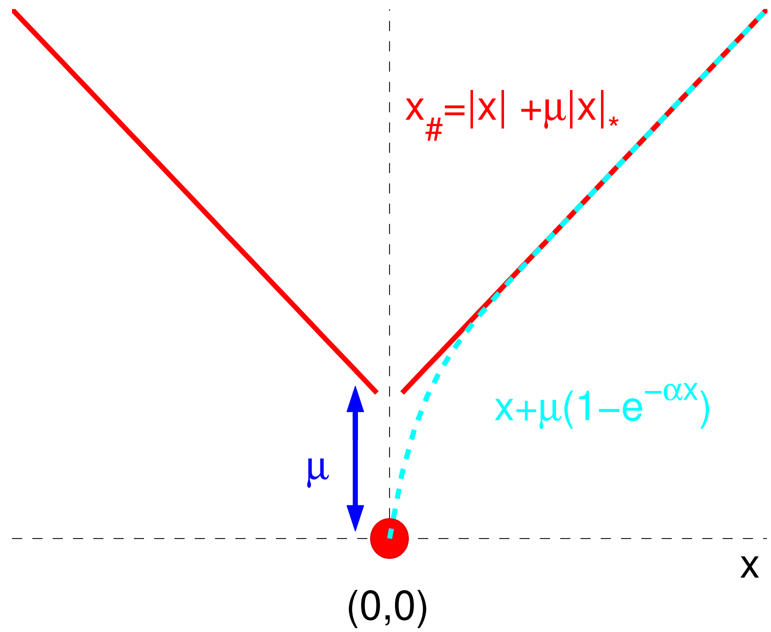

Figure 3: **Loss function $x_\#$ approximation by $x + \mu(1 - \varepsilon^{\alpha x})$ on $x \geq 0$.**

It can be shown (Mangasarian, 1996) that this algorithm, which is essentially the repeated solution of the linear program (10), terminates in a finite number of steps (typically 5 to 8) at a point that satisfies the necessary optimality condition that the current iterate is a fixed point of the linearized problem.

Remark 4.3 *When $K(A, A')$ is large, for example where $m > 500$, it is convenient to use RSVM (Lee and Mangasarian, 2000) in order to utilize a smaller reduced kernel $K(A, \bar{A}')$ where $\bar{A}' \in R^{\bar{m} \times n}$ and $\bar{m}$ is 5%-20% of m. In all cases, however, the initial $(u^0, \gamma^0, y^0, v^0)$ were obtained using the 1-norm formulation (10).*

Remark 4.4 *The Minimal Kernel Algorithm 4.2 terminates at a primal solution (u, v, y, γ) and a corresponding solution $(\tilde{t}, \tilde{r}, \tilde{s})$ to the dual of the last linearized problem (26). The reduced $m_1 \times m_2$ rectangular kernel $K(\bar{A}_{m_1}, \bar{A}_{m_2})$ for this last linearized problem has dimensions corresponding to:*

$$m_1 = card(\tilde{t}_+), \;\; m_2 = card(u_+), \tag{28}$$

where:

$$\bar{A}_{m_1} := \{A_i \mid \tilde{t}_i > 0\} \; and \; \bar{A}_{m_2} := \{A_i \mid |u|_i > 0\}. \tag{29}$$

Typically, m_1 and m_2 are considerably smaller than m. The number m_1 is the number of the first m constraints of the linear program (26) with positive dual multipliers at the solution of the last linear program solved by Algorithm 4.2. The number m_2 determines the number of data points that the classifier:

$$K(x', \bar{A}'_{m_2})Du_{m_2} - \gamma = 0, \tag{30}$$

depends on. We refer to m_2 as the number of kernel support vectors. *For a standard support vector machine (1), $m_1 = m_2$. If the final linearization of (26) is re-solved with the smaller kernel $K(\bar{A}_{m_1}, \bar{A}_{m_2})$, the solution is the same as that obtained by Algorithm 4.2. However, if the standard 1-norm SVM formulation (10) is solved directly with the reduced kernel $K(\bar{A}_{m_1}, \bar{A}'_{m_2})$, then a result close to but not identical to that of Algorithm 4.2 is obtained because the objective function of (10) does not contain the linearization of the pound function.*

5. Computational Results

All our computations were performed on the University of Wisconsin Data Mining Institute "locop1" Windows NT machine using MATLAB 6 (MATLAB, 1994-2001). The machine utilizes a 400 Mhz Pentium II and allows a maximum of 2 Gigabytes of memory for each process.

We tested Algorithm 4.2 on eight publicly available datasets in order to demonstrate that the algorithm gives equally correct test set results by using a drastically reduced kernel size compared to a kernel that uses the full dataset. A Gaussian kernel (Vapnik, 2000, Cherkassky and Mulier, 1998) was used throughout the numerical tests:

$$K(A_i, A'_j) = \varepsilon^{-\sigma\|A_i - A_j\|_2^2},$$

where σ is a positive parameter determined by using a tuning set for each dataset.

5.1 Results for the Checkerboard

The first dataset used was the Checkerboard (Ho and Kleinberg, 1996a,b) consisting of 486 black points and 514 white points taken from a 16-square checkerboard. These 1000 points constituted the training dataset while the test set consisted of 39,601 points taken from a uniform 199×199 grid of points. This example was picked in order to demonstrate visually how a small subset of the original data can achieve an accurate separation.

5.2 Results on the USPS Dataset

Figure 4 depicts the separation obtained using a reduced kernel $K(A_{m_1}, A_{m_2})$ with $m_1 = 30$ and $m_2 = 27$. The 30 points constituting A_{m_1} and the 27 points constituting A_{m_2} are depicted in Figure 4 as circles and stars respectively. The total of these points is 5.7% of the original data and the rather accurate separating surface (30) depends merely on 27 points, that is 2.7% of the original data.

Although our algorithm is primarily intended for two-class problems, we have also applied it to the ten-category USPS (US Postal Service) dataset of hand-written numbers (Vapnik, 2000, Bottou et al., 1994). This well-known dataset consists of 7291 training patterns and 2007 testing patterns, collected from real-life zip codes. Each pattern consists of one of the ten numbers 0 to 9 and is represented by a digital image consisting of 16×16 pixels, which results in a 256-dimensional input space. In our tests here we used a one-from-the-rest approach, which led to the construction of 10 classifiers, each separating one class from the rest. The final classification was done by selecting the class corresponding to the classifier with the largest output value. The number of kernel support vectors reported

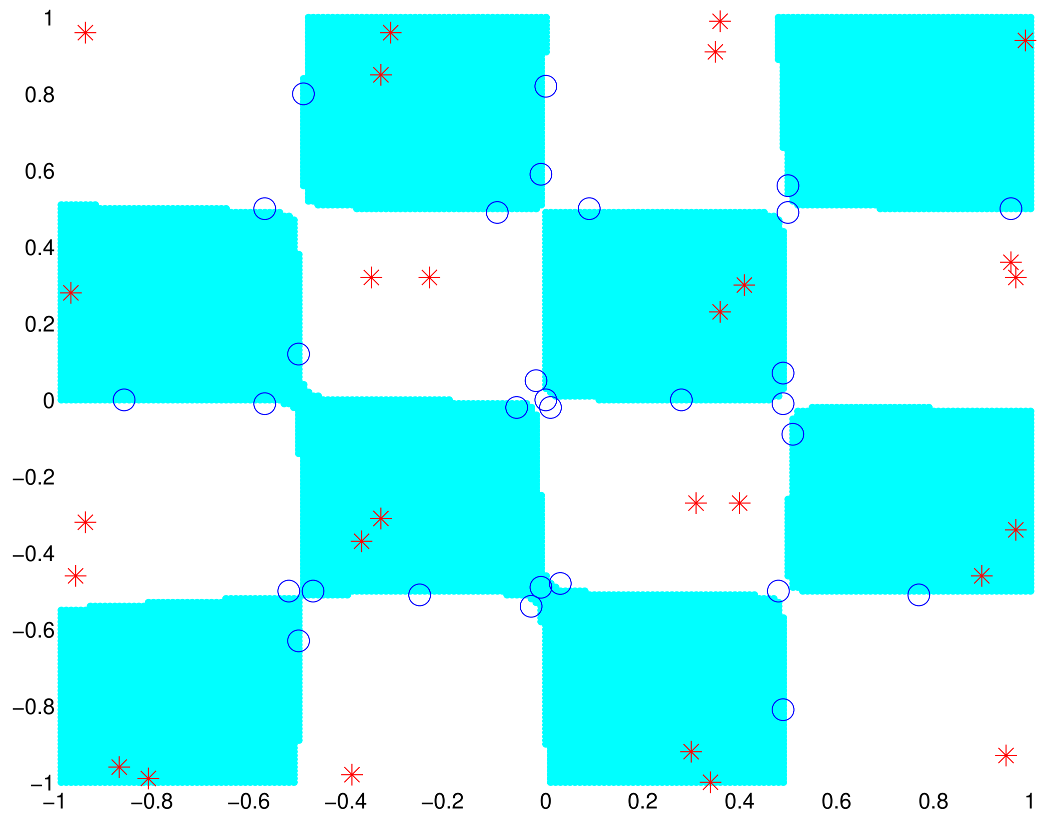

Figure 4: **The checkerboard classifiier depends on only 2.7% of the original data depicted as 27 stars, and is trained on only 3.0% of the original data depicted as 30 circles. The reduced kernel here is 30×27 compared to an original kernel for the full dataset of size $1,000 \times 1,000$. Running times were 109.9 seconds for the full kernel and 0.1 seconds for the reducedkernel.**

in Table 1 is the total over the ten classifiers. We compare our algorithm with other kernel reducing methods such as the Sparse Greedy Matrix Approximation (**SGMA**) (Smola and Schölkopf, 2000) and the Relevance Vector Machine (**RVM**) (Tipping, 2000) as well as with the standard Support Vector Machine (**SVM**) (Vapnik, 2000). In all the experiments in this section a Gaussian kernel was used. Table 1 compares the number of kernel support vectors for all these methods as well as the test set error. We note that our error rate is somewhat higher than that of the other methods. However, our number of kernel support vectors is the second smallest, in line with objectives of the paper. The average time to compute each of the ten classifiers for the USPS dataset was 24.6 minutes on our Pentium II 400MHz machine.

5.3 Results on Six Public Datasets

The next set of problems were from the University of California Irvine Repository (Murphy and Aha, 1992) and varied in size between 297 to 8124 points, with input space dimen-

Method	No. of Kernel Support Vectors	Test Error %
MKC	376	6.7%
SVM (Vapnik, 2000)	1727	4.1%
SGMA (Smola and Schölkopf, 2000)	590	*
RVM (Tipping, 2000)	316	5.1%

Table 1: **Comparison of total number of kernel support vectors (in ten classifiers) and test set error for 4 methods: Minimal Kernel Classifier (MKC), standard Support Vector Machine (SVM), Sparse Greedy Matrix Approximation (SGMA) and Relevance Vector Machine (RVM).**
*** No test error is reported in (Smola and Schölkopf, 2000).**

sionality between 6 to 34. The purpose of the experiments was to show that the proposed Minimal Kernel Algorithm 4.2 can achieve three objectives:

(i) It can generate a nonlinear separating surface with less than 10% of the original data. This is a key property for incremental algorithms (Fung and Mangasarian, 2002) where obsolete old data is retired before merging it with new incoming data.

(ii) Accuracy of ten-fold cross validation is as good or better than that of a nonlinear classifier that depends on a much bigger subset of the original training set.

(iii) Since the reduced kernel classifier depends on a very small subset of the original data, classification of a new point is done very quickly. This makes this method very attractive for applications where there are time constraints on the testing procedure or where there are storage constraints on the classifier.

The above and other results are given in Table 2 averaged over ten-fold runs for each dataset.

6. Conclusion

When one is confronted with a complex classification problem for which a linear classifier fails, e.g. the checkerboard example of Figure 4, one needs to resort to a nonlinear classifier. We have addressed one of the serious computational difficulties associated with such problems when we attempt to use a nonlinear kernel classifier on a large training dataset. Such problems result in the unwieldy explicit dependence of the nonlinear classifier on almost all the entries of a huge dataset. By utilizing a leave-one-out error bound, we have proposed an algorithm, based on solving a few linear programs, that generates an accurate nonlinear kernel classifier that typically depends on less than 10% of the original data. With the exception of the multiple class USPS dataset, the nonlinear separator is as or more accurate than classifiers using the full dataset and is much faster to evaluate, making it suitable for fast on-line decision making. This allows us to tackle nonlinear classification problems that hitherto were very difficult to solve. The fact that our formulation also reduces the number of data points needed if we have to re-solve the problem, suggests promising new applications, such as incremental classification of massive datasets where only a small fraction of the data is kept before merging it into incrementally arriving data.

Data Set $m \times n$	Reduced rectangular kernel $m_1 \times m_2$	MKC Ten-fold test set correctness % (Ten-fold time sec.)	SVM Ten-fold test set correctness % (SV)*	Kernel Support vector reduction ** %	Testing time reduction †% (SVM-MKC time sec.)
Ionosphere 351×34	30.2×15.7	94.9% (172.3)	92.9% (288.2)	94.6%	94.9% (3.05-0.16)
Cleveland Heart 297×13	64.6×7.6	85.8% (147.2)	85.5 % (241.0)	96.9 %	96.3 % (0.84-0.03)
Pima Indians 768×8	263.1×7.8***	77.7 % (303.3)	76.6 % (637.3)	98.8%	98.8 % (3.95-0.05)
BUPA Liver 345×6	144.4×10.5	75.0 % (285.9)	72.7 % (310.5)	96.6%	97.5 % (0.59-0.02)
Tic-Tac-Toe 958×9	31.3×14.3***	98.4 % (150.4)	98.3 % (861.4)	98.3%	98.2 % (6.97-0.13)
Mushroom 8124×22	933.8×47.9***	89.3 % (2763.5)	oom (NA)	NA	NA

Table 2: **Results for six UC Irvine datasets showing the percentage of reduction achieved over ten-fold runs. The last column gives testing time reduction resulting from using our minimal kernel classifier (MKC) instead of a regular full kernel classifier. The numbers m_1 and m_2 are averages over ten folds and refer to the dimensionality of the reduced kernel $K(\bar{A}_{m_1}, \bar{A}_{m_1})$ as explained in Remark 4.4. All linear programs were solved using CPLEX 6.5 (CPL, 1992). NA denotes "Not Available", while oom denotes "out of memory".**

* **Number of support vectors obtained using a standard quadratic programming support vector machine (1).**

** **Comparison between the number of MKC kernel support vectors m_2 defined by (28) and the number of support vectors (SV) using the standard quadratic programming support vector machine (1).**

*** **RSVM (Lee and Mangasarian, 2000) was used here in order to obtain a smaller *initial* kernel for each of the Pima Indians dataset (768×150 instead of 768×768), the Tic-Tac-Toe dataset (958×96 instead of 958×958) and the Mushroom dataset (8124×400 instead of 8124×8124).**

† If T_{svm} **is the average single fold time over ten folds testing for the standard SVM classifier that depends on SV data points, and** T_r **is the average single fold time over ten folds testing using an MKC classifier that depends only on** m_2 **data points, then this percentage is given by:** $100 \times (1 - \frac{T_{svm}}{T_r})$

Acknowledgments

The research described in this Data Mining Institute Report 00-08, November 2000, was supported by National Science Foundation Grants CCR-9729842, CCR-0138308 and CDA-9623632, by Air Force Office of Scientific Research Grant F49620-00-1-0085 and by the Microsoft Corporation. Revised July and September 2002.

References

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *International Conference on Pattern Recognition*, pages 77–87. IEEE Computer Society Press, 1994.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

P. S. Bradley and O. L. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13:1–10, 2000. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

P. S. Bradley, O. L. Mangasarian, and J. B. Rosen. Parsimonious least norm approximation. *Computational Optimization and Applications*, 11(1):5–21, October 1998a. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-03.ps.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-03.ps.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998b. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-21.ps.

V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.

Using the CPLEX(TM) Linear Optimizer and CPLEX(TM) Mixed Integer Optimizer (Version 2.0). CPLEX Optimization Inc., Incline Village, Nevada, 1992.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.

M. C. Ferris and T. S. Munson. Interior point methods for massive support vector machines. Technical Report 00-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 2000. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-05.ps.

G. Fung and O. L. Mangasarian. Data selection for support vector machine classification. In R. Ramakrishnan and S. Stolfo, editors, *Proceedings KDD-2000: Knowledge Discovery and Data Mining, August 20-23, 2000, Boston, MA*, pages 64–70, New York, 2000. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-02.ps.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA*, pages 77–86, New York, 2001. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps.

G. Fung and O. L. Mangasarian. Incremental support vector machine classification. In H. Mannila R. Grossman and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, Virginia, April 11-13,2002*, pages 247–260, Philadelphia, 2002. SIAM. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-08.ps.

T. K. Ho and E. M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 880–885, Vienna, Austria, 1996a. http://cm.bell-labs.com/who/tkh/pubs.html. Checker dataset at: ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/checker.

T. K. Ho and E. M. Kleinberg. Checkerboard dataset, 1996b. http://www.cs.wisc.edu/math-prog/mpml.html.

T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, San Mateo, CA, 1999. Morgan Kaufmann.

Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. Technical Report 00-07, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, July 2000. Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM Proceedings. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

Yuh-Jye Lee and O. L. Mangasarian. SSVM: A smooth support vector machine. *Computational Optimization and Applications*, 20:5–22, 2001. Data Mining Institute, University of Wisconsin, Technical Report 99-03. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-03.ps.

O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.

O. L. Mangasarian. Machine learning via polyhedral concave minimization. In H. Fischer, B. Riedmueller, and S. Schaeffler, editors, *Applied Mathematics and Parallel Computing - Festschrift for Klaus Ritter*, pages 175–188. Physica-Verlag A Springer-Verlag Company, Heidelberg, 1996. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-20.ps.

O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24: 15–23, 1999a. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07r.ps.

O. L. Mangasarian. Minimum-support solutions of polyhedral concave programs. *Optimization*, 45:149–162, 1999b. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-05.ps.

O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

O. L. Mangasarian and R. R. Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, November 1979.

O. L. Mangasarian and D. R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):950–955, 2000. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-09.ps.

O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 577–583, Cambridge, MA, 2001a. MIT Press. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001b. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.

O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46:255–269, 2002. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-02.ps.

MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, 1997.

P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.

M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326, Cambridge, MA, 2000. MIT Press.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

B. Schölkopf, C. Burges, and A. Smola (editors). *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3.

A. Smola, P. L. Bartlett, B. Schölkopf, and J. Schuurmann (editors). *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000. URL `citeseer.nj.nec.com/smola00sparse.html`.

M. Tipping. The relevance vector machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 652–658, Cambridge, MA, 2000. MIT Press.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.

V. N. Vapnik and O. Chapelle. Bounds on expectation for SVM. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 261–280, Cambridge, MA, 2000. MIT Press.

J. Weston and R. Herbrich. Adaptive margin support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 281–295, Cambridge, MA, 2000. MIT Press.

Y. Ye. *Interior Point Algorithms Theory and Analysis*. John Wiley & Sons, New York, 1997.