Chapter 1

DATA DISCRIMINATION VIA NONLINEAR GENERALIZED SUPPORT VECTOR MACHINES

O. L. Mangasarian & David R. Musicant
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
olvi,musicant@cs.wisc.edu

Abstract The main purpose of this paper is to show that new formulations of support vector machines can generate nonlinear separating surfaces which can discriminate between elements of a given set better than a linear surface. The principal approach used is that of generalized support vector machines (GSVMs) [21] which employ possibly indefinite kernels. The GSVM training procedure is carried out by either a simple successive overrelaxation (SOR) [22] iterative method or by linear programming. This novel combination of powerful support vector machines [28, 7] with the highly effective SOR computational algorithm [19, 20, 17], or with linear programming, allows us to use a *nonlinear* surface to discriminate between elements of a dataset that belong to one of two categories. Numerical results on a number of datasets show improved testing set correctness, by as much as a factor of two, when comparing the nonlinear GSVM surface to a linear separating surface.

1 INTRODUCTION

Support vector machines (SVMs) discriminate between the elements of a finite set in an n-dimensional real space, each point of which belongs to one of two categories. The SVM constructs a separating plane, in the original n-dimensional space or in a higher dimensional space, that is mid-way between two parallel planes that bound the two categories [6, 28, 7]. A conventional SVM in its dual formulation contains bound constraints, as well as an equality constraint that requires special treatment in iterative procedures such as successive overrelaxation (SOR).

A very simple convex quadratic program with bound constraints *only* can be obtained by taking the dual of the quadratic program associated with a support vector machine that maximizes the margin (distance between bounding separating planes) with respect to *all* the parameters determining the bounding planes [22]. This can be achieved by maximizing the margin (distance between bounding separating planes) with respect to *both* orientation of the separating planes as well as their location relative to the origin. This quadratic program can be solved by successive overrelaxation for datasets with as many as 10 million points [22] to obtain a *linear* separating surface. In the present paper we show how to obtain a *nonlinear* separating surface for more complex datasets by using SOR, thus opening the possibility of solving large scale nonlinear separation problems. The nonlinear surface is induced by a possibly indefinite kernel. We also use linear programming as another approach for nonlinear kernel separation, where the finite simplex algorithm [8] is used.

We note that nonlinear kernel separation is a standard technique in the SVM literature [6, 28, 7]. The techniques that we offer in this paper differ from previous work in some important respects. SOR for support vector machines has been shown to be effective in [22] for linear kernels. In this paper, we show that SOR can be successfully used in conjunction with nonlinear kernels. The SVM literature generally requires that nonlinear kernels be of a certain type, namely kernels that satisfy Mercer's positive definiteness condition [6]. In [21], it is shown theoretically and by means of a small example that linear programming based SVMs do not require Mercer's condition. In this work, we demonstrate the effectiveness and power of this approach using different nonlinear kernels on five publicly available datasets.

In Section 2 we first state our linear discrimination problem as a conventional SVM quadratic program (1.1) and its dual (1.5). We then introduce our modification of the SVM quadratic program (1.1), which is given in problem (1.4) and its dual in (1.6). This modification is an SOR-solvable convex quadratic program with bound constraints only and a positive semidefinite kernel. The dual problems (1.5) and (1.6) for a linear separator are generalized to a kernel-induced nonlinear separator in (1.8) and (1.11). Another version of our generalized nonlinear support vector machine is also given in (1.12) below, in which the kernel is completely arbitrary but the problem is convex and hence also SOR-solvable. We note that in our previous work [22] only a *linear kernel* (polynomial kernel of degree one) was implemented, whereas various *nonlinear kernels* that may be indefinite will be used in this paper. In Section 3 we state our SOR algorithm for nonlinear kernels and establish its linear convergence. In Section 4 we give linear programming formulations of the generalized support vector machine problem, including a novel one that includes two or more arbitrary kernels. In Section 5 we give numerical results for various datasets all of which show the improvement obtained in using the generalized support vector machine with a nonlinear kernel over a linear one. Section 6 draws some conclusions and points out future directions such as parallel SOR implementations that may lead to the solution of very large discrimination problems with nonlinear separating surfaces.

A word about our notation. 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 plus function x_+ is defined as $(x_+)_i = \max\ \{0, x_i\}$, $i = 1, \ldots, n$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For an $m \times n$ matrix A, A_i will denote the ith row of A. The identity matrix in a real space of arbitrary dimension will be denoted by I, while a column vector of ones of arbitrary dimension will be denoted by e. We shall employ the MATLAB "dot" notation [23] to signify application of a function to all components of a matrix or a vector. For example if $A \in R^{m\times n}$, then $A^2_{\bullet} \in R^{m\times n}$ will denote the matrix obtained by squaring each element of A. For $A \in R^{m\times n}$ and $B \in R^{n\times \ell}$, the **kernel** $K(A, B)$ maps $R^{m\times n} \times R^{n\times \ell}$ into $R^{m\times \ell}$. In particular if x and y are column vectors in R^n then, $K(x', A')$ is a row vector in R^m, $K(x', y)$ is a real number and $K(A, A')$ is an $m \times m$ matrix. Note that for our purposes here $K(A, A')$ will be assumed to symmetric, that is $K(A, A') = K(A, A')'$. For a vector x in the n-dimensional real space R^n, the step function $x_{\bullet *}$ is defined as a vector in R^n of minus ones, zeros and plus ones, corresponding to negative, zero and positive components of x respectively. Similarly for an $m \times n$ matrix M, $M_{\bullet *}$ will denote an $m \times n$ matrix of minus ones, zeros and plus ones.

2 THE SUPPORT VECTOR MACHINE AND ITS GENERALIZATION: QUADRATIC FORMULATION

We consider the problem of discriminating between 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 linear support vector machine

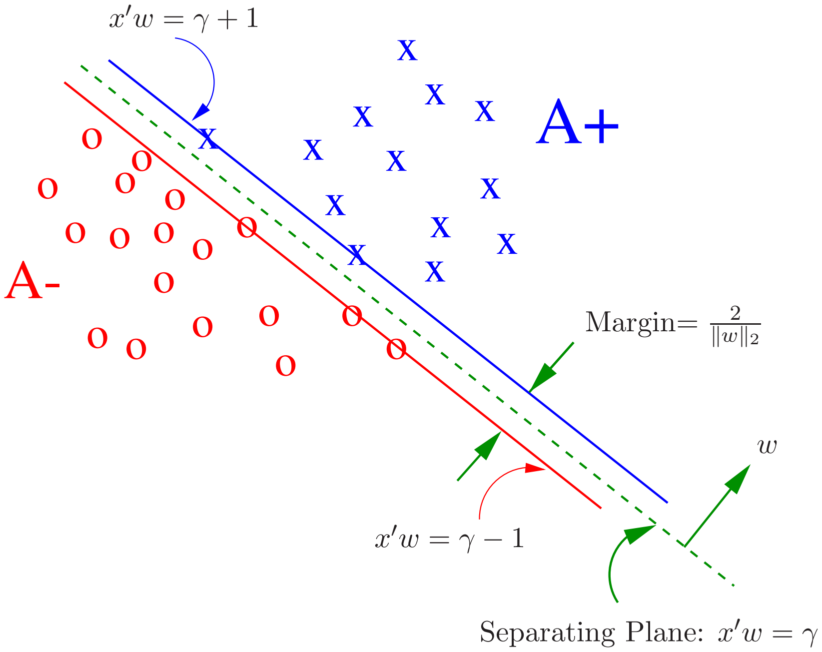

Figure 1.1 **The Linearly Separable Case: The bounding planes of equation (1.2) with margin $\frac{2}{\|w\|_2}$**

with a linear kernel AA' [28, 7] is given by the following for some $\nu > 0$:

$$\begin{array}{lrcl} \min\limits_{w,\gamma,y} & \nu e'y + \frac{1}{2}w'w & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0. \end{array} \tag{1.1}$$

Here w is the normal to the bounding planes:

$$\begin{array}{lcccc} x'w & - & \gamma & = & +1 \\ x'w & - & \gamma & = & -1. \end{array} \tag{1.2}$$

The first plane above bounds the class 1 points and the second plane bounds the class -1 points, if the two classes are linearly separable and $y = 0$. This is illustrated in Figure 1.1.

If the classes are linearly inseparable then the two planes bound the two classes with a "soft margin" determined by a slack variable $y \geq 0$, that is:

$$\begin{array}{lcccccl} x'w & - & \gamma & + & y_i & \geq & +1, \quad \text{for } x = A_i \text{ and } D_{ii} = +1, \\ x'w & - & \gamma & - & y_i & \leq & -1, \quad \text{for } x = A_i \text{ and } D_{ii} = -1. \end{array} \tag{1.3}$$

The one-norm of the slack variable y is minimized with weight ν in (1.1). The quadratic term in (1.1), which is twice the reciprocal of the square of the 2-norm distance $\frac{2}{\|w\|_2}$ between the two planes of (1.2) in the n-dimensional space of $w \in R^n$ for a *fixed* γ, maximizes that distance. In our approach here, which is similar to that of [2, 14, 13], we measure the distance between the planes in the $(n+1)$-dimensional space of $[w; \gamma] \in R^{n+1}$ which is $\frac{2}{\|[w;\gamma]\|_2}$. Using this measure of distance instead results in our modification of the SVM problem as follows:

$$\begin{array}{ll} \min\limits_{w,\gamma,y} & \nu e'y + \frac{1}{2}(w'w + \gamma^2) \\ \text{s.t.} & D(Aw - e\gamma) + y \ \geq \ e \\ & \qquad\qquad\qquad y \ \geq \ 0. \end{array} \tag{1.4}$$

The Wolfe duals [18, Section 8.2] to the quadratic programs (1.1) and (1.4) are as follows.

$$\begin{array}{c} \max\limits_{u} \ -\dfrac{1}{2}u'DAA'Du + e'u, \text{ s.t. } e'Du = 0, \ 0 \leq u \leq \nu e \\ (w = A'Du). \end{array} \tag{1.5}$$

$$\begin{array}{c} \max\limits_{u} \ -\dfrac{1}{2}u'DAA'Du - \dfrac{1}{2}u'Dee'Du + e'u, \text{ s.t. } 0 \leq u \leq \nu e \\ (w = A'Du, \ \gamma = -e'Du, \ y = (e - D(Aw - e\gamma))_+). \end{array} \tag{1.6}$$

We note immediately that the variables (w, γ, y) of the primal problem (1.4) can be directly computed from the solution u of its dual (1.6) as indicated. However, only the variable w of the primal problem (1.1) can be directly computed from the solution u of its dual (1.5) as shown. The remaining variables (γ, y) of (1.1) can be computed by setting $w = A'Du$ in (1.1), where u is a solution of its dual (1.5), and solving the resulting linear program for (γ, y). Alternatively, γ can be determined by minimizing the expression for $e'y = e'(e - D(Aw - e\gamma))_+$ as a function of the single variable γ after w has been expressed as a function of the dual solution u as indicated in (1.5), that is:

$$\min_{\gamma \in R} e'(e - D(AA'Du - e\gamma))_+). \tag{1.7}$$

We note that formulations (1.5) and (1.6) can be extended to a general nonlinear kernel $K(A, A') : R^{m\times n} \times R^{n\times m} \to R^{m\times m}$ by replacing AA' by the kernel $K(A, A')$ and the SOR approach can similarly be extended to a general nonlinear kernel [21]. Thus the conventional support vector machine with a positive semidefinite kernel $K(A, A')$ [28, 7] is given by the following dual convex quadratic program for some $\nu > 0$ which is obtained from (1.5) by replacing AA' by $K(A, A')$:

$$\begin{array}{lrcl} \min\limits_{u \in R^m} & \frac{1}{2}u'DK(A,A')Du - e'u & & \\ & \text{s.t.} \quad e'Du & = & 0 \\ & 0 \leq u & \leq & \nu e. \end{array} \tag{1.8}$$

A solution u of this quadratic program leads to the nonlinear separating surface

$$K(x', A')Du = \gamma, \tag{1.9}$$

where γ is defined as in (1.7) above by a solution of:

$$\min_{\gamma \in R} e'(e - D(K(A,A')Du - e\gamma))_+. \tag{1.10}$$

Note that the nonlinear surface (1.9) in R^n obviates the need for computing w, since w is defined only in the higher dimensional space in which the nonlinear surface (1.9) is mapped into a plane.

By a similar approach we obtain the following quadratic dual with bound constraints only and a *nonlinear* kernel $K(A, A')$ by replacing AA' in (1.6) by $K(A, A')$:

$$\begin{array}{c} \min\limits_{u} \dfrac{1}{2}u'D[K(A,A') + ee']Du - e'u, \text{ s.t. } 0 \leq u \leq \nu e \\ (\gamma = -e'Du), \end{array} \tag{1.11}$$

with an explicit expression for γ and the same separating surface (1.9). The formulation (1.11) allows a direct use of the SOR algorithm to solve very large problems.

We mention in passing another possible change in (1.8) allows the use of possibly indefinite kernels. One particular formulation motivated by [21, Equation (10)] is the following one which, as in (1.11), incorporates the equality $e'Du = 0$ into the objective function:

$$\begin{array}{c} \min\limits_{u} \dfrac{1}{2}u'D[K(A,A')K(A,A')' + ee']Du - e'u, \text{ s.t. } 0 \leq u \leq \nu e \\ (\gamma = -e'Du), \end{array} \tag{1.12}$$

with a separating surface [21, Equations (1) & (10)] different from (1.9):

$$K(x', A')K(A, A')'Du = \gamma. \tag{1.13}$$

Note that the kernel $K(A, A')$ in the formulation (1.12) is completely arbitrary and need not satisfy any positive semidefiniteness condition in order for the objective function of (1.12) to be convex. This makes the separating surface (1.13) quite general.

3 SUCCESSIVE OVERRELAXATION FOR NONLINEAR GSVM

Our goal is now to present an algorithm which can be used to efficiently solve the support vector machines presented in the previous section. Traditional quadratic programming techniques do not exploit the structure of this problem, and run exceedingly slowly. Moreover, such techniques expect that all the data is available in memory. This criterion results in significant limitations in scaling these algorithms. Successive overrelaxation was used effectively in [22] for *linear* support vector machines in order to overcome these difficulties. We therefore apply it here to the *nonlinear* SVMs (1.11) and (1.12). Both of these problems can be stated as:

$$\min_{u} \frac{1}{2}u'Mu - e'u, \text{ s.t. } u \in S = \{u \mid 0 \leq u \leq \nu e\}, \tag{1.14}$$

with the symmetric matrix M defined as $D(K(A, A') + ee')D$ and $D(K(A, A')K(A, A')' + ee')D$ respectively for these two problems. Thus M will be positive semidefinite if we assume that $K(A, A')$ is positive semidefinite in the former case and under no assumptions in the latter case. If we decompose M as follows:

$$M = L + E + L', \tag{1.15}$$

where $L \in R^{m \times m}$ is the strictly lower triangular part of the symmetric matrix M, and $E \in R^{m \times m}$ is the positive diagonal of M, then a necessary and sufficient optimality condition for (1.14) for positive semidefinite M is the following gradient projection optimality condition [27, 17]:

$$u = (u - \omega E^{-1}(Mu - e))_{\#}, \ \omega > 0, \tag{1.16}$$

where $(\cdot)_{\#}$ denotes the 2-norm projection on the feasible region S of (1.14), that is:

$$((u)_{\#})_i = \left\{ \begin{array}{l} 0 \ \text{ if } u_i \leq 0 \\ u_i \ \text{ if } 0 < u_i < \nu \\ \nu \ \text{ if } u_i \geq \nu \end{array} \right\}, \ i = 1, \ldots, m. \tag{1.17}$$

Our SOR method, which is a matrix splitting method that converges linearly to a point $\bar{u}$ satisfying (1.16), consists of splitting the matrix M

into the sum of two matrices as follows:

$$M = \omega^{-1}E(B + C), \text{ s.t. } B - C \text{ is positive definite.} \tag{1.18}$$

For our specific problem we take:

$$B = (I + \omega E^{-1}L),\ C = ((\omega - 1)I + \omega E^{-1}L'),\ 0 < \omega < 2. \tag{1.19}$$

This leads to the following linearly convergent [17, Equation (3.14)] matrix splitting algorithm:

$$u^{i+1} = (u^{i+1} - Bu^{i+1} - Cu^i + \omega E^{-1}e)_\#, \tag{1.20}$$

for which

$$B + C = \omega E^{-1}M,\ B - C = (2 - \omega)I + \omega E^{-1}(L - L'). \tag{1.21}$$

Note that for $0 < \omega < 2$, the matrix $B + C$ is positive semidefinite and matrix $B - C$ is positive definite. The matrix splitting algorithm (1.20) results in the following easily implementable SOR algorithm once the values of B and C given in (1.19) are substituted in (1.20).

Algorithm 1 SOR Algorithm *Choose $\omega \in (0, 2)$. Start with any $u^0 \in R^m$. Having u^i compute u^{i+1} as follows:*

$$u^{i+1} = (u^i - \omega E^{-1}(Mu^i - e + L(u^{i+1} - u^i)))_\#, \tag{1.22}$$

until $\|u^{i+1} - u^i\|$ is less than some prescribed tolerance.

Remark 2 *The components of u^{i+1} are computed in order of increasing component index. Thus the SOR iteration (1.22) consists of computing u_j^{i+1} using $(u_1^{i+1}, \ldots, u_{j-1}^{i+1}, u_j^i, \ldots, u_m^i)$. That is, the latest computed components of u are used in the computation of u_j^{i+1}. The strictly lower triangular matrix L in (1.22) can be thought of as a substitution operator, substituting $(u_1^{i+1}, \ldots, u_{j-1}^{i+1})$ for $(u_1^i, \ldots, u_{j-1}^i)$.*

We have immediately from [17, Proposition 3.5] the following linear convergence result.

Theorem 3 SOR Linear Convergence *The iterates $\{u^i\}$ of the SOR Algorithm 1 converge R-linearly to a solution of $\bar{u}$ of the dual problem (1.13), and the objective function values $\{f(u^i)\}$ of (1.13) converge Q-linearly to $f(\bar{u})$. That is for $i \geq \bar{i}$ for some $\bar{i}$:*

$$\begin{array}{rcl} \|u^i - \bar{u}\| & \leq & \mu\delta^i, \text{ for some } \mu > 0,\ \delta \in (0,1), \\ f(u^{i+1}) - f(\bar{u}) & \leq & \tau(f(u^i) - f(\bar{u})), \text{ for some } \tau \in (0,1). \end{array} \tag{1.23}$$

We turn our attention to linear programming formulations of the nonlinear GSVM.

4 THE NONLINEAR GSVM AS A LINEAR PROGRAM

We briefly describe now how the nonlinear separating surface (1.9) can also be generated by a linear program. For greater detail, the reader is referred to [21].

We begin with the standard linear support vector machine (1.1). Motivated by the Wolfe dual (1.5), we make the substitution $w = A'Du$ into the primal and obtain:

$$\begin{array}{rrcl} \min\limits_{u,\gamma,y} & \nu e'y + \frac{1}{2}uDAA'Du & & \\ \text{s.t.} & D(AA'Du - e\gamma) + y & \geq & e \\ & y & \geq & 0. \end{array} \tag{1.24}$$

As in the SOR formulation, we substitute a nonlinear kernel for the AA' term. Additionally, we "simplify" the regularization term in the objective to depend only on the variable u in a linear fashion. We achieve this by introducing a variable s, which takes on the value of the absolute value of u. The substitution surrogates support vector suppression for maximizing the distance between separating bounding planes, which has a similar regularizing effect. This results in the linear program [21, Equation (11)]:

$$\begin{array}{rrcl} \min\limits_{u,\gamma,y,s} & \nu e'y + e's & & \\ \text{s.t.} & D(K(A,A')Du - e\gamma) + y & \geq & e \\ & s \geq u & \geq & -s \\ & y & \geq & 0. \end{array} \tag{1.25}$$

A solution (u, γ, y, s) to this linear program for a completely arbitrary kernel $K(A, A')$ will provide a separating surface given by (1.9). We note that if the linear kernel (i.e. polynomial kernel of degree 1) $K(A, A') = AA'$ is used in the linear program above and w is introduced and set equal to $A'Du$ we obtain the high-performing 1-norm linear SVM proposed in [5] and utilized successfully in [4, 1, 3].

Since the linear programming formulation does not impose restrictions on the kernel employed, we are free to use more than one kernel in a given formulation. For example we can use **both** a linear kernel and a discontinuous neural network kernel in a single linear programming formulation. More generally if we let $K^1(A, A')$ and $K^2(A, A')$ be two arbitrary kernels, then the corresponding linear programming formulation becomes:

$$
\begin{array}{rrcl}
\displaystyle\min_{u^1,u^2,\gamma,y,s^1,s^2} & \nu e'y + e's^1 + e's^2 & & \\
\text{s.t.} \quad D(K^1(A,A')Du^1 + K^2(A,A')Du^2 - e\gamma) + y & \geq & e & \\
s^1 \;\geq\; u^1 & \geq & -s^1 & \\
s^2 \;\geq\; u^2 & \geq & -s^2 & \\
y & \geq & 0. &
\end{array}
\tag{1.26}
$$

The corresponding separating surface to a solution $(u^1, u^2, \gamma, y, s^1, s^2)$ of this linear program is given by:

$$K^1(x', A')Du^1 + K^2(x', A')Du^2 = \gamma \tag{1.27}$$

For example we can use a linear and a discontinuous neural network kernel for $K^1(A, A')$ and $K^2(A, A')$ as follows:

$$K^1(A', A') = (AA' - ee'); \; K^2(A', A') = (AA' - ee')_{\bullet *}, \tag{1.28}$$

where, as defined in the Introduction, the subscript $\bullet *$ denotes replacement of each component of the matrix AA' by 1, 0, or -1, depending on whether it is positive, zero or negative, respectively. Note that in general this neural network kernel is not positive semidefinite and hence standard support vector techniques cannot make use of it. We also note that the linear programming formulation (1.26) can be extended to more than two kernels in an obvious manner.

5 NUMERICAL TESTING

We tested nonlinear kernels versus linear ones by using different formulations of the generalized SVM, namely the quadratic programming formulations as given in (1.11) and (1.12), and the linear programming formulation as given in (1.25). Most of our code development was in the MATLAB environment [23], though the actual solutions to the SVMs were handled externally. We coded the SOR algorithm in C, and accessed it via the MATLAB “mex” interface which allows one to execute external C code as a native MATLAB function [24]. The linear programs were solved with the MINOS solver [26]. All experiments were run on the University of Wisconsin Computer Sciences Department Ironsides cluster. This cluster of four Sun Enterprise E6000 machines consists of 16 UltraSPARC II 250 MHz processors and 2 gigabytes of RAM on each node resulting in a total of 64 processors and 8 gigabytes of RAM.

The SVMs were tested on the following datasets from the University of California at Irvine (UCI) repository [25]:

- The *liver-disorders* dataset contains 345 points, each consisting of six features. Class 1 contains 145 points, and class -1 contains 200 points.

- The *letter-recognition* dataset is used for recognizing letters of the alphabet. Traditionally, this dataset is used for multi-category classification. We used it here in a two class situation by taking a subset of those points which correspond to the letter "A," and a subset of those points which correspond to the letter "B." This resulted in a dataset of 600 points with 6 features, where each class contained 300 points.

- The *mushroom* dataset is a two class dataset which contains a number of categorical attributes. We transformed each categorical attribute into a series of binary attributes, one attribute for each distinct value. For example, this dataset contains an attribute called "cap surface," which can take one one of four categories, namely "fibrous," "grooves," "scaly," or "smooth." We represented this as four attributes. A 1 is assigned to the attribute that corresponds to the actual category, and a 0 to the rest. Thus each instance of this specific categorical attribute corresponds to one row of a 4 x 4 diagonal matrix of ones. A subset of the entire mushroom dataset was used, so as to shorten the running times of our experiments. The final dataset we used contained 22 features with 200 points in class 1 and 300 points in class -1.

- The *tic-tac-toe* dataset is a two class dataset that contains incomplete tic-tac-toe games. All those games with a possible winning move for "X" end up in category 1, and all other games end up in category -1. We have intentionally represented this problem with a poor representation scheme to show the power of non-linear kernels in overcoming this difficulty. Each tic-tac-toe game is represented by 9 attributes, where each attribute corresponds to a spot on the tic-tac-toe board. An "X" is represented by 1, an "O" is represented by 0, and a blank is represented by -1. We used a subset of this dataset with 9 attributes, 200 points in class 1, and 100 points in class -1.

We additionally used some synthetic datasets, namely the checkerboard dataset [16, 15] and also a synthetic dataset of Gaussian distribution. The Gaussian synthetic dataset, suggested by Usama Fayyad of Microsoft Research, was created by constructing 20 random centers, then generating "clouds" of points around these centers in a Gaussian fashion based on random covariance matrices. An arbitrary hyperplane

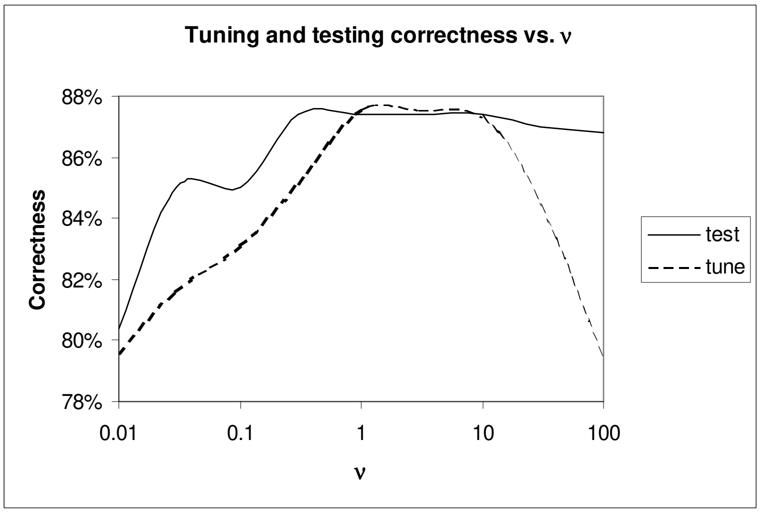

Figure 1.2 **Tuning and testing set accuracy for SOR with a linear kernel on the mushroom dataset.**

was used to separate the centers into two classes. Each center was assigned to the class corresponding to the side of the hyperplane it fell on; furthermore, each randomly generated point was assigned to the same class as the center from which it came. We then used a scaling parameter to size the clouds so that the data was not linearly separable.

All experiments were conducted using tenfold cross validation, which is is a standard technique used for evaluating machine learning algorithms. We randomly partitioned each dataset into ten equal parts, and ran the algorithm ten times on 90% of this dataset. Each time, a different one of the ten segments was held out to serve as a test set. We used this test set to measure the generalization capability of the learning method. Kernels and parameters were chosen via the use of a *tuning* set, an artificially created testing set surrogate taken completely from the training set. We ran our experiments for a variety of parameters and kernels on the training data with the tuning set removed, and found those parameters that performed best on the tuning set. We then used these settings to retrain on the training set as a whole (tuning set included) to report our generalization results on the test set. Tuning set accuracy is usually a good predictor of test accuracy, as shown in Figure 1.2.

Table 1 shows results on a number of datasets, using the SOR Algorithm 1 with both linear and nonlinear kernels. In each of the cases shown, the chosen nonlinear kernel generalizes better than the linear

Dataset	Kernel Type	Training Correctness	Testing Correctness	Time (secs)
Letter Recognition	Linear	85.00%	84.83%	11.1
	6th degree polynomial	92.65%	86.33%	11.0
Mushroom	Linear	87.40%	87.40%	6.6
	Negative squared cubic	89.38%	88.80%	6.7
Tic Tac Toe	Linear	60.37%	56.33%	1.4
	Quadratic	100.00%	83.33%	1.4
Gaussian	Linear	83.20%	81.70%	43.0
	Quadratic	97.68%	93.41%	43.0

Table 1.1 **SOR training and testing set correctness for linear and quadratic kernels**

kernel. This is particularly noticeable for the tic-tac-toe and Gaussian datasets. In general, the kernels we used are as follows:

Kernel 1 Polynomial $((\frac{A}{\lambda} - \rho)(\frac{A}{\lambda} - \rho)' - \mu)^{d}_{\bullet}$

Kernel 2 Negative Polynomial $-((\frac{A}{\lambda} - \rho)(\frac{A}{\lambda} - \rho)' - \mu)^{d}_{\bullet}$

The variables λ, ρ, μ are scalar parameters and d is a positive integer. When a parameter is added or subtracted from a matrix or vector, it means componentwise addition or subtraction of the parameter in conformity with MATLAB notation. The "negative squared cubic" kernel listed for the mushroom dataset is a negative polynomial kernel of degree 3, with the problem formulation as in (1.12). Other kernels use the problem formulation shown in (1.11).

Table 2 shows training and testing set correctness, using the linear programming formulation (1.25) with various kernels, under tenfold cross validation for the above mentioned datasets. For these experiments, we again implemented a linear kernel and various polynomial kernels. Note that these polynomial kernels are indefinite in general, and therefore do not satisfy Mercer's condition.

As in the SOR case above, the nonlinear kernels gave improved testing set results over linear kernels. The most striking results are those for the checkerboard dataset of Figure 1.3. An initial attempt using a k-nearest-neighbor algorithm yields the very coarse separation shown in Figure 1.4. We instead used a nonlinear SVM, and obtained a tenfold testing set correctness of over 97.70% for our nonlinear kernel. This is reflected in the very realistic rendering in Figure 1.5 depicting the resulting separation of the original dataset. Our experiments were inspired by those of Kaufman in [16] where eighth degree polynomial and radial basis kernels were used in a standard support vector machine formulation to separate the checkerboard dataset. However, our separation depicted in Figure 1.5 is considerably sharper than those of [16].

Dataset	Kernel Type	Training Correctness	Testing Correctness	Time (secs)
Liver Disorders	Linear	71.21%	68.70%	23.5
	Quadratic	78.33%	73.37%	46.8
Letter Recognition	Linear	85.43%	84.33%	62.5
	Negative Cubic	97.15%	87.67%	553.7
Tic Tac Toe	Linear	66.67%	66.67%	17.1
	Quadratic	98.78%	90.67%	21.4
Mushroom	Linear	87.40%	87.40%	45.5
	Negative 7th degree polynomial	89.56%	88.80%	35.1
Checkerboard	Linear	51.12%	48.60%	206.8
	6th degree polynomial	99.97%	98.50%	289.0

Table 1.2 **Linear programming training and testing set correctness for linear and nonlinear support vector machines**

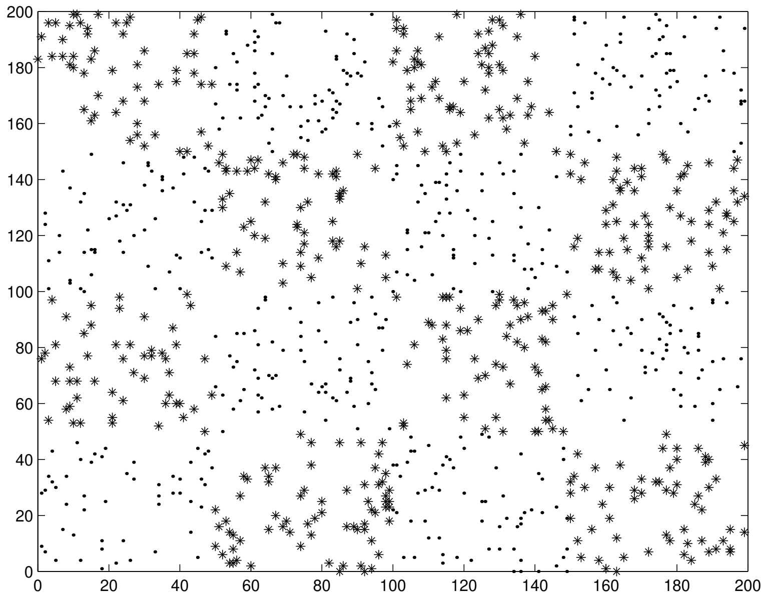

Figure 1.3 **Checkerboard training dataset**

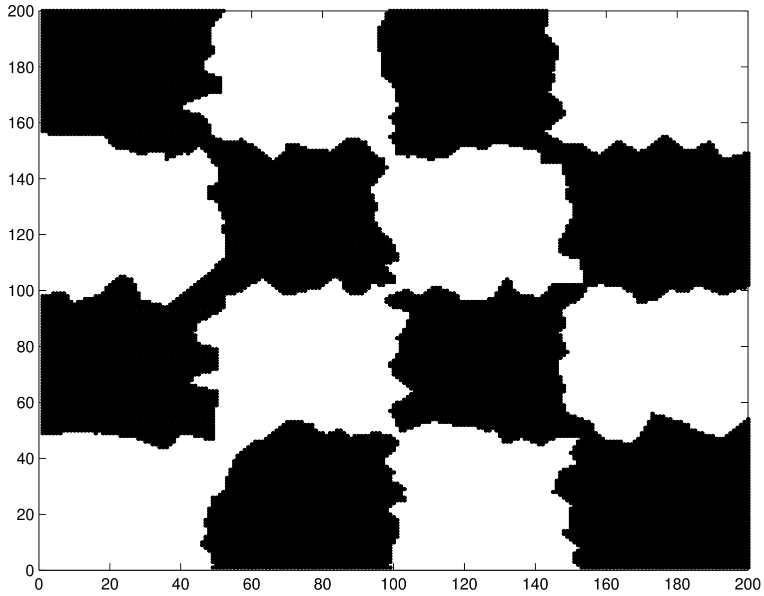

Figure 1.4 **k-nearest-neighbor separation of the checkerboard dataset (k=1, 2-norm)**

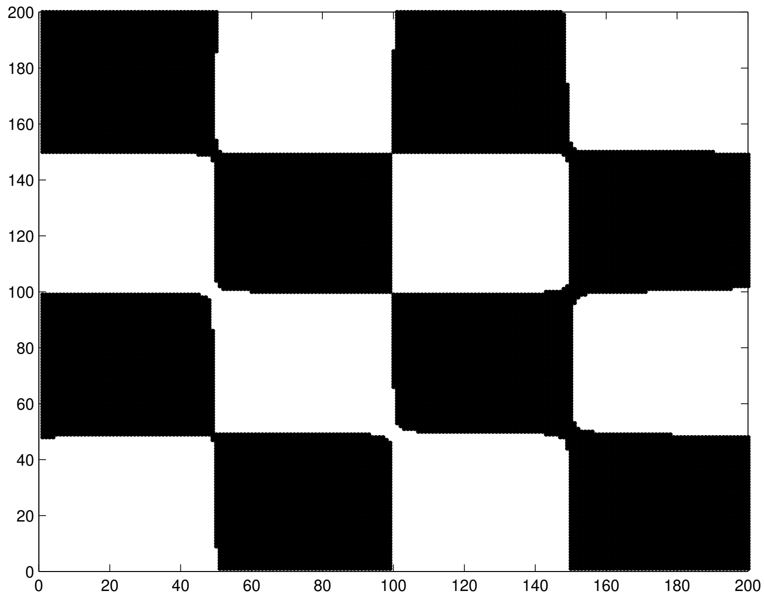

Figure 1.5 **Indefinite sixth degree polynomial kernel separation of the checkerboard dataset ($\nu = 10,000,\ \lambda = 100,\ \rho = 1,\ d = 6,\ \mu = 0.5$)**

6 CONCLUSION

We have implemented a simple iterative method, successive overrelaxation, and a linear programming approach for the solution of discrimination problems using generalized support vector machines with nonlinear kernels. Both of these methods yield advantages over previous techniques for the solution of SVMs. SOR has significantly smaller memory requirements than previous techniques, and can therefore be used on larger datasets than previously considered. In fact, it can be used on some datasets that do not fit in memory [22]. The linear programming formulation yields the advantage that linear programs can be solved significantly more efficiently than quadratic programs. The methods scale up and can be parallelized by using techniques already implemented [10, 11, 9, 12]. Future work includes multicategory nonlinear discrimination via kernel methods using successive overrelaxation and linear programming, as well as parallel implementation on multiprocessor machines.

Acknowledgements

We are indebted to our colleagues Usama Fayyad, Microsoft Research, for suggesting the Gaussian dataset, to Linda Kaufman, Bell Labs, for supplying us with the checkerboard dataset and to Tin Kam Ho, Bell Labs, for allowing us to make the checkerboard dataset public. This research is supported by National Science Foundation Grants CCR-9729842 and CDA-9623632, and by Air Force Office of Scientific Research Grant F49620-97-1-0326 as Mathematical Programming Technical Report 99-03, March 1999, and revised March 2000.

References

[1] K. P. Bennett, D. Hui, and L. Auslender. On support vector decision trees for database marketing. Department of Mathematical Sciences Math Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY 12180, March 1998. http://www.math.rpi.edu/∼bennek/.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

[3] 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.

[4] E. J. Bredensteiner. *Optimization Methods in Data Mining and Machine Learning*. PhD thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, 1997.

[5] E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. *Computational Optimizations and Applications*, 10:111–126, 1998.

[6] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

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

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

[9] R. De Leone and O. L. Mangasarian. Serial and parallel solution of large scale linear programs by augmented Lagrangian successive overrelaxation. In A. Kurzhanski, K. Neumann, and D. Pallaschke, editors, *Optimization, Parallel Processing and Applications*, pages 103–124, Berlin, 1988. Springer–Verlag. Lecture Notes in Economics and Mathematical Systems 304.

[10] R. De Leone, O. L. Mangasarian, and T.-H. Shiau. Multi-sweep asynchronous parallel successive overrelaxation for the nonsymmetric linear complementarity problem. *Annals of Operations Research*, 22:43–54, 1990.

[11] R. De Leone and M. A. Tork Roth. Massively parallel solution of quadratic programs via successive overrelaxation. *Concurrency: Practice and Experience*, 5:623–634, 1993.

[12] M. C. Ferris and O. L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, 1(4):487–500, 1991.

[13] T.-T. Friess. Support vector neural networks: The kernel adatron with bias and soft margin. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, England, 1998. Revised Version: www.brunner-edv.com/friess/.

[14] T.-T. Friess, N. Cristianini, and C. Campbell. The kernel-adatron algorithm: A fast and simple learning procedure for support vector machines. In Jude Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference (ICML'98)*, pages 188–196, San Francisco, 1998. Morgan Kaufmann. http://svm.first.gmd.de/papers/FriCriCam98.ps.gz.

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

[16] L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In Bernhard Schölkopf, Christopher

J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 147–167. MIT Press, 1999.

[17] Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, 46:157–178, 1993.

[18] O. L. Mangasarian. *Nonlinear Programming*. McGraw–Hill, New York, 1969. Reprint: SIAM Classic in Applied Mathematics 10, 1994, Philadelphia.

[19] O. L. Mangasarian. Solution of symmetric linear complementarity problems by iterative methods. *Journal of Optimization Theory and Applications*, 22(4):465–485, August 1977.

[20] O. L. Mangasarian. On the convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem. *SIAM Journal on Optimization*, 1:114–122, 1991.

[21] 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.

[22] O. L. Mangasarian and David R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-18.ps.

[23] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1992.

[24] MATLAB. *Application Program Interface Guide*. The MathWorks, Inc., Natick, MA 01760, 1997.

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

[26] B. A. Murtagh and M. A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83.20, Stanford University, December 1983. MINOS 5.4 Release Notes, December 1992.

[27] B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.

[28] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.