Proximal Support Vector Machine Classifiers

Glenn Fung and Olvi L. Mangasarian
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
(gfung,olvi)@cs.wisc.edu

ABSTRACT

Instead of a standard support vector machine (SVM) that classifies points by assigning them to one of two disjoint halfspaces, points are classified by assigning them to the closest of two parallel planes (in input or feature space) that are pushed apart as far as possible. This formulation, which can also be interpreted as regularized least squares and considered in the much more general context of regularized networks [8, 9], leads to an extremely fast and simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations. In contrast, standard SVMs solve a quadratic or a linear program that require considerably longer computational time. Computational results on publicly available datasets indicate that the proposed proximal SVM classifier has comparable test set correctness to that of standard SVM classifiers, but with considerably faster computational time that can be an order of magnitude faster. The linear proximal SVM can easily handle large datasets as indicated by the classification of a 2 million point 10-attribute set in 20.8 seconds. All computational results are based on 6 lines of MATLAB code.

Keywords

data classification, support vector machines, linear equations

1. INTRODUCTION

Standard support vector machines (SVMs) [33, 6, 3, 5, 20], which are powerful tools for data classification, classify points by assigning them to one of two disjoint halfspaces. These halfspaces are either in the original input space of the problem for linear classifiers, or in a higher dimensional feature space for nonlinear classifiers [33, 6, 20]. Such standard SVMs require the solution of either a quadratic or a linear program which require specialized codes such as [7]. In contrast we propose here a *proximal* SVM (PSVM) which classifies points depending on proximity to one of two parallel planes that are pushed as far apart as possible. In fact our geometrically motivated proximal formulation has been considered in the much more general context of regularization networks [8, 9]. These results, which give extensive theoretical and statistical justification for the proximal approach, do not contain the extensive computational implementation and results given here. Furthermore, our specific formulation leads to a strongly convex objective function which is not always the case in [8, 9]. Strong convexity plays a key role in the simple proximal code provided here as well the very fast computational times obtained. Obtaining a linear or nonlinear PSVM classifier requires nothing more sophisticated than solving a single system of linear equations. Efficient and fast linear equation solvers are freely available [1] or are part of standard commercial packages such as MATLAB [26], and can solve large systems.

We briefly summarize the contents of the paper now. In Section 2 we introduce the proximal linear support vector machine, give the Linear Proximal Algorithm 2.1 and an explicit expression for the leave-one-out-error in terms of problem data (16). In Section 3 we introduce the proximal kernel-based nonlinear support vector machine, the corresponding nonlinear classifier (28) and the Nonlinear Proximal Algorithm 3.1. Section 3 contains many numerical testing results for both the linear and nonlinear classifiers based on an extremely simple MATLAB [26] code of 6 lines for both the linear and nonlinear PSVM. The results surpass all other algorithms compared to in speed and give very comparable testing set correctness.

A word about our notation and 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 $step(x)$ is defined as $step(x)_i = 1$ if $x_i > 0$ else $step(x)_i = 0$ if $x_i \leq 0$ for $i = 1, \ldots, n$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$ and the 2-norm of x will be denoted by $\|x\|$. 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, while $A_{\cdot j}$ is the jth column of A. A column vector of ones of arbitrary dimension will be denoted by e. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$, the kernel $K(A, B)$ maps $R^{m\times n} \times R^{n\times k}$ into $R^{m\times k}$. 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. The base of the natural logarithm will be denoted by ε. We will make use of the following Gaussian kernel [33, 6, 20] that is frequently used in the SVM literature:

$$K(A,B) = \varepsilon^{-\mu\|A_i{}'-B_{\cdot j}\|^2},\ i=1\dots,m,\ j=1\dots,k, \quad (1)$$

where $A \in R^{m\times n}$, $B \in R^{n\times k}$ and μ is a positive constant. The identity matrix of arbitrary dimension will be denoted by I.

2. THE LINEAR PROXIMAL SUPPORT VECTOR MACHINE

We consider the problem, depicted in Figure 1, 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 class $A+$ or $A-$ as specified by a given $m\times m$ diagonal matrix D with plus ones or minus ones along its diagonal. For this problem, the standard support vector machine with a linear kernel [32, 6] is given by the following quadratic program with parameter $\nu > 0$:

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

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

$$\begin{array}{rcl} x'w & = & \gamma + 1 \\ x'w & = & \gamma - 1, \end{array} \quad (3)$$

that bound most of the sets $A+$ and $A-$ respectively. The constant γ determines their location relative to the origin. When the two classes are strictly linearly separable, that is when the error variable $y = 0$ in (2) (which is not the case shown in Figure 1), the plane $x'w = \gamma + 1$ bounds all of the class $A+$ points, while the plane $x'w = \gamma - 1$ bounds all of the class $A-$ points as follows:

$$\begin{array}{rcll} A_i w & \geq & \gamma + 1, & \text{for} \quad D_{ii} = 1 \\ A_i w & \leq & \gamma - 1, & \text{for} \quad D_{ii} = -1. \end{array} \quad (4)$$

Consequently, the plane:

$$x'w = \gamma, \quad (5)$$

midway between the bounding planes (3), is a separating plane that separates $A+$ from $A-$ completely if $y = 0$, else only approximately as depicted in Figure 1. The quadratic term in (2), which is twice the reciprocal of the square of the 2-norm distance $\frac{2}{\|w\|}$ between the two bounding planes of (3) (see Figure 1), maximizes this distance, often called the "margin". Maximizing the margin enhances the generalization capability of a support vector machine [32, 6]. If the classes are linearly inseparable, which is the case shown in Figure 1, then the two planes bound the two classes with a "soft margin" (i.e. bound approximately with some error) determined by the nonnegative error variable y, that is:

$$\begin{array}{rcccll} A_i w & + & y_i & \geq & \gamma + 1, & \text{for } D_{ii} = 1 \\ A_i w & - & y_i & \leq & \gamma - 1, & \text{for } D_{ii} = -1. \end{array} \quad (6)$$

The 1-norm of the error variable y is minimized parametrically with weight ν in (2) resulting in an approximate separating plane (5) as depicted in Figure 1. This plane acts as a linear classifier as follows:

$$x'w - \gamma \begin{cases} > 0, & \text{then } x \in A+, \\ < 0, & \text{then } x \in A-, \\ = 0, & \text{then } x \in A+ \text{ or } x \in A-. \end{cases} \quad (7)$$

Our point of departure is similar to that of [23, 24], where the optimization problem (2) is replaced by the following problem:

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \nu\frac{1}{2}\|y\|^2 + \frac{1}{2}(w'w + \gamma^2) & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \end{array} \quad (8)$$

Note that no explicit nonnegativity constraint is needed on y, because if any component y_i is negative then the objective function can be decreased by setting that $y_i = 0$ while still satisfying the corresponding inequality constraint. Note further that the 2-norm of the error vector y is minimized instead of the 1-norm, and the margin between the bounding planes is maximized with respect to both orientation w and relative location to the origin γ. Extensive computational experience, as in [23, 24] and other publications, indicates that this formulation is just as good as the classical formulation (2) with some added advantages such as strong convexity of the objective function. Our key idea in this present paper is to make a very simple, but very fundamental change in the formulation (8), namely replace the inequality constraint by an equality as follows:

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \nu\frac{1}{2}\|y\|^2 + \frac{1}{2}(w'w + \gamma^2) & & \\ \text{s.t.} & D(Aw - e\gamma) + y & = & e \end{array} \quad (9)$$

This modification, even though very simple, changes the nature of optimization problem significantly. In fact it turns out that we can write an explicit exact solution to the problem in terms of the problem data as we show below, whereas it is impossible to do that in the previous formulations because of the combinatorial nature of these formulations. Geometrically the formulation (9) is depicted in Figure 2 which can be interpreted as follows. The planes $x'w - \gamma = \pm 1$ are not bounding planes anymore, but can be thought of as "proximal" planes, around which the points of each class are clustered and which are pushed apart as far apart as possible by the term $(w'w + \gamma^2)$ in the objective function which is nothing other than the reciprocal of the 2-norm distance squared between the two planes in the w, γ space.

We note that our formulation (9) can be also interpreted as a regularized least squares solution [31] of the system of linear *equations* $D(Aw - e\gamma) = e$, that is finding an approximate solution (w, γ) with least 2-norm. Similarly the standard SVM formulation (2) can be interpreted, by using linear programming perturbation theory [21], as a least 2-norm approximate solution to the system of linear *inequalities* $D(Aw - e\gamma) \geq e$. Neither of these interpretations, however, is based on maximizing the margin, the distance between the parallel planes (3), which is a key feature of support vector machines [33, 6, 20].

The Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions [19, p. 112] for our equality constrained problem (9) are obtained by setting equal to zero the gradients with respect to (w, γ, y, u) of the Lagrangian:

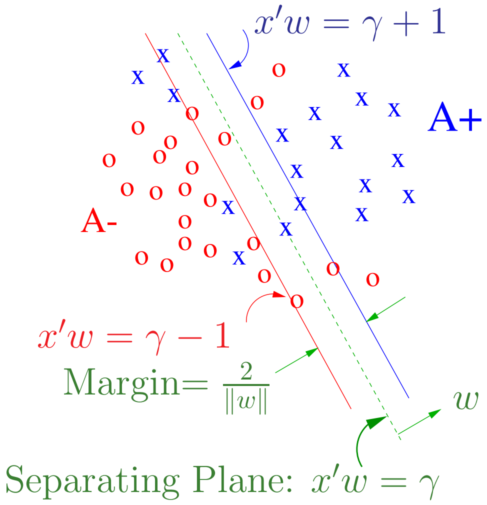

Figure 1: The Standard Support Vector Machine Classifier: The approximately bounding planes of equation (4) with a soft (i.e. with some error) margin $\frac{2}{\|w\|}$, and the plane of equation (5) approximately separating $A+$ from $A-$.

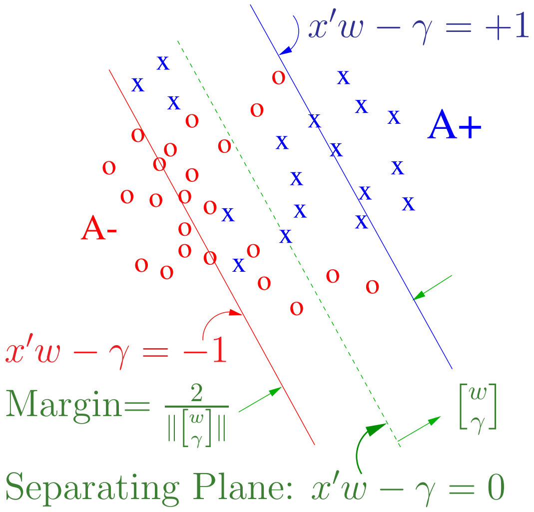

Figure 2: The Proximal Support Vector Machine Classifier: The planes $x'w - \gamma = \pm 1$ around which points of the sets $A+$ and $A-$ cluster and which are pushed apart by the optimization problem (9).

$$L(w, \gamma, y, u) = \frac{\nu}{2}\|y\|^2 + \frac{1}{2}\|\begin{bmatrix} w \\ \gamma \end{bmatrix}\|^2 - u'(D(Aw - e\gamma) + y - e). \tag{10}$$

Here, $u \in R^m$ is the Lagrange multiplier associated with the equality constraint of (9). Setting the gradients of L equal to zero gives the following KKT optimality conditions:

$$\begin{array}{lcl} w - A'Du & = & 0 \\ \gamma + e'Du & = & 0 \\ \nu y - u & = & 0 \\ D(Aw - e\gamma) + y - e & = & 0 \end{array} \tag{11}$$

The first three equations of (11) give the following following expressions for the original problem variables (w, γ, y) in terms of the Lagrange multiplier u:

$$w = A'Du, \quad \gamma = -e'Du, \quad y = \frac{u}{\nu}. \tag{12}$$

Substituting these expressions in the last equality of (11) allows us to obtain an explicit expression for u in terms of the problem data A and D as follows:

$$u = (\frac{I}{\nu} + D(AA' + ee')D)^{-1}e = (\frac{I}{\nu} + HH')^{-1}e, \tag{13}$$

where H is defined as:

$$H = D[A \quad -e]. \tag{14}$$

Having u from (13), the explicit solution (w, γ, y) to our problem (9) is given by (12). Because the solution (13) for u entails the inversion of a possibly massive $m \times m$ matrix, we make immediate use of the Sherman-Morrison-Woodbury formula [14, p. 51] for matrix inversion, as was done in [23, 10, 24], which results in:

$$u = \nu(I - H(\frac{I}{\nu} + H'H)^{-1}H')e. \tag{15}$$

This expression involves the inversion of a much smaller dimensional matrix of size $n \times n$ and completely solves the classification problem. For concreteness we explicitly state our very simple algorithm.

ALGORITHM 2.1. **Linear Proximal SVM** *Given m data points in R^n represented by the $m \times n$ matrix A and a diagonal matrix D of ± 1 labels denoting the class of each row of A, we generate the linear classifier (7) as follows:*

- *(i) Define H by (14) where e is an $m \times 1$ vector of ones and compute u by (15) for some positive ν. (Typically ν is chosen by means of a tuning set.)*
- *(ii) Determine (w, γ) from (12).*
- *(iii) Classify a new x by using (7).*

For standard SVMs, support vectors consist of all data points which are the complement of the data points that can be dropped from the problem without changing the separating plane (5) [33, 20]. Thus, for the standard SVM formulation (2), support vectors correspond to data points for which the

Lagrange multipliers are nonzero because, solving (2) with these data points only will give the same answer as solving it with the entire dataset. In our proximal formulation (9) however, the Lagrange multipliers u are merely a multiple of the error vector y: $u = \nu y$ as given by (12). Consequently, because all components of y are typically nonzero since none of the data points usually lie on the proximal planes $x'w = \pm 1$, the concept of support vectors needs to be modified as follows. Because $(w, \gamma) \in R^{n+1}$ are given as linear functions of u by (11), it follows by the basis theorem for linear equations [13, Theorem 2.11][25, Lemma 2.1], that at most $n+1$ linearly independent data points are needed to determine a given specific $(w, \gamma) \in R^{n+1}$. Guided by this fact that only a small number of data points can characterize any specific (w, γ), we define the concept of *ϵ-support vectors* as those data points A_i for which error vector y_i is less than ϵ in absolute value. We typically pick ϵ small enough such that about 1% of the data are ϵ-support vectors. Re-solving our proximal SVM problem (9) with these data points and a ν adjusted (typically upwards) by a tuning set gives test set correctness that is essentially identical to that obtained by using the entire dataset.

We note that with explicit expressions (w, γ, y, u) in terms of problem data given by (12) and (15), we are able to get also an explicit expression for the leave-one-out-correctness *looc*, that is the fraction of correctly classified data points if each point in turn is left out of the PSVM formulation (9) and then is classified by the classifier (7). Omitting some algebra, we have the following leave-one-out-correctness:

$$looc = \frac{e'step(h)}{m}, \tag{16}$$

where the "step" function is defined in the Introduction, and for $i = 1, \dots, m$:

$$h_i = D_i H_i H^{i'}(I - H^i(\frac{I}{\nu} + H^{i'}H^i)^{-1}H^{i'})e. \tag{17}$$

Here, H is defined by (14), H_i denotes row i of H, while H^i denotes H with row H_i removed from H. Similarly, D_i denotes row i of D.

We extend now some of the above results to nonlinear proximal support vector machines.

3. NONLINEAR PROXIMAL SUPPORT VECTOR MACHINES

To obtain our nonlinear proximal classifier we modify our equality constrained optimization problem (9) as in [20, 18] by replacing the primal variables w by its dual equivalent $w = A'Du$ from (12) to obtain:

$$\begin{array}{rl} \displaystyle\min_{(u,\gamma,y)\in R^{n+1+m}} & \nu\frac{1}{2}\|y\|^2 + \frac{1}{2}(u'u + \gamma^2) \\ \text{s.t.} & D(AA'Du - e\gamma) + y \;=\; e, \end{array} \tag{18}$$

where the objective function has also been modified to minimize weighted 2-norm sums of the problem variables (u, γ, y). If we now replace the linear kernel AA' by a nonlinear kernel $K(A, A')$ as defined in the Introduction, we obtain:

$$\begin{array}{rl} \displaystyle\min_{(u,\gamma,y)\in R^{n+1+m}} & \nu\frac{1}{2}\|y\|^2 + \frac{1}{2}(u'u + \gamma^2) \\ \text{s.t.} & D(K(A, A')Du - e\gamma) + y \;=\; e. \end{array} \tag{19}$$

Using the shorthand notation:

$$K := K(A, A'), \tag{20}$$

the Lagrangian for (19) can be written similarly to (10) as:

$$L(u, \gamma, y, v) = \frac{\nu}{2}\|y\|^2 + \frac{1}{2}\|\begin{bmatrix} u \\ \gamma \end{bmatrix}\|^2 - v'(D(KDu - e\gamma) + y - e). \tag{21}$$

Here, $v \in R^m$ is the Lagrange multiplier associated with the equality constraint of (19). Setting the gradients of this Lagrangian with respect to $(u, \gamma, y, , v)$ equal to zero gives the following KKT optimality conditions:

$$\begin{array}{lcl} u - DK'Dv & = & 0 \\ \gamma + e'Dv & = & 0 \\ \nu y - v & = & 0 \\ D(KDu - e\gamma) + y & = & e. \end{array} \tag{22}$$

The first three equations of (22) give the following following expressions for (u, γ, y) in terms of the Lagrange multiplier v:

$$u = DK'Dv, \;\; \gamma = -e'Dv, \;\; y = \frac{v}{\nu}. \tag{23}$$

Substituting these expressions in the last equality of (22) gives an explicit expression for v in terms of the problem data A and D as follows:

$$v = (\frac{I}{\nu} + D(KK' + ee')D)^{-1}e = (\frac{I}{\nu} + GG'))^{-1}e, \tag{24}$$

where G is defined as:

$$G = D[K \quad -e]. \tag{25}$$

Note the similarity between G above and H as defined in (14). This similarity allows to obtain G from the expression (14) for H by replacing A by K in (14). This can be taken advantage of in the MATLAB code 4.1 of Algorithm 2.1 which is written for the linear classifier (7). To generate a nonlinear classifier by Algorithm 3.1 merely replace A by K in the algorithm.

Having the solution v from (24), the solution (u, γ, y) to our problem (19) is given by (23). Unlike the situation with linear kernels, the Sherman-Morrison-Woodbury formula is useless here because the kernel matrix $K = K(A, A')$ is a square $m \times m$ matrix, so the inversion must take place in a potentially high-dimensional R^m. However, the reduced kernel techniques of [17] can be utilized to reduce dimensionality. The effectiveness of these reduced kernels is demonstrated by means of a numerical test problem in the next section of the paper.

The nonlinear separating surface corresponding to the kernel $K(A, A')$ [20, Equation (8.1)] and can be deduced from the linear separating surface (5) and $w = A'Du$ from (12) as follows:

$$x'w - \gamma = x'A'Du - \gamma = 0. \tag{26}$$

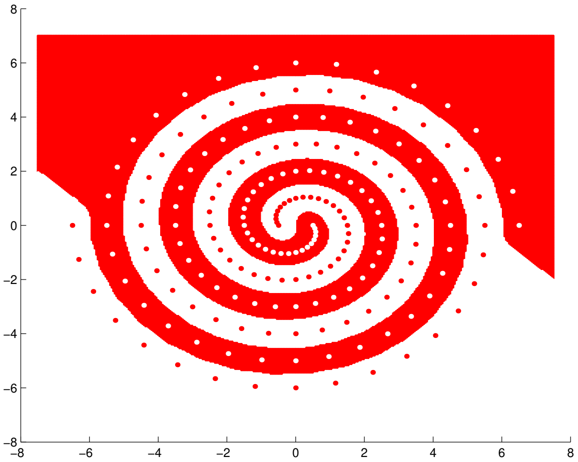

Figure 3: The spiral dataset consisting of 97 black points and 97 white points intertwined as two spirals in 2-dimensional space. PSVM with a Gaussian kernel generated a sharp nonlinear spiral-shaped separating surface.

If we replace $x'A'$ by the corresponding kernel expression $K(x', A')$, and substitute from (23) for u and γ: $u = DK'Dv$ and $\gamma = -e'Dv$ we obtain the nonlinear separating surface:

$$\begin{aligned} K(x', A')Du - \gamma &= K(x', A')DDK(A, A')'Dv + e'Dv \\ &= (K(x', A')K(A, A')' + e')Dv = 0. \end{aligned} \tag{27}$$

The corresponding *nonlinear* classifier to this nonlinear separating surface is then:

$$(K(x', A')K(A, A')' + e')Dv \begin{cases} > 0, & \text{then } x \in A+, \\ < 0, & \text{then } x \in A-, \\ = 0, & \text{then } x \in A+ \text{ or } x \in A-. \end{cases} \tag{28}$$

We now give an explicit statement of our nonlinear classifier algorithm.

ALGORITHM 3.1. **Nonlinear Proximal SVM** *Given m data points in R^n represented by the $m \times n$ matrix A and a diagonal matrix D of ± 1 labels denoting the class of each row of A, we generate the nonlinear classifier (28) as follows:*

(i) Choose a kernel function $K(A, A')$, typically the Gaussian kernel (1).

(ii) Define G by (25) where e is an $m \times 1$ vector of ones and compute v by (24) for some positive ν. (Typically ν is chosen by means of a tuning set.)

(iii) The nonlinear surface (27) with the computed v constitutes the nonlinear classifier (28) for classifying a new point x.

The nonlinear classifier (28), which is a direct generalization of the linear classifier (7), works quite effectively as indicated by the numerical examples presented in the next section.

4. NUMERICAL IMPLEMENTATION AND COMPARISONS

All our computations were performed on the University of Wisconsin Data Mining Institute "locop1" machine, which utilizes a 400 Mhz Pentium II and allows a maximum of 2 Gigabytes of memory for each process. This computer runs on Windows NT server 4.0, with MATLAB 6 installed. Even though "locop1" is a multiprocessor machine, only one processor was used for all the experiments since MATLAB is a single threaded application and does not distribute any load across processors [26]. Our algorithms require the solution of a single square system of linear equations of the size of the number of input attributes n in the linear case, and of the size of the number of data points m in the nonlinear case. When using a rectangular kernel [18], the size of the problem can be reduced from m to k with $k << m$ for the nonlinear case. Because of the simplicity of our algorithm, we give below the actual MATLAB implementation that was used in our experiments and which consists of 6 lines of native MATLAB code:

CODE 4.1. **PSVM MATLAB Code**

```
function [w,gamma] = psvm(A,D,nu)
 % PSVM:linear and nonlinear classification
 % INPUT: A, D, nu. OUTPUT: w, gamma
 % [w, gamma] = psvm(A,D,nu);
  [m,n]=size(A);e=ones(m,1);H=D*[A -e];
  r=sum(H)'; %r=H'*e;
  r=(speye(n+1)/nu+H'*H)\r; %solve (I/nu+H'*H)r=H'*e
  u=nu*(1-(H*r)); s=D*u;
  w=(s'*A)';                   %w=A'*D*u
  gamma=-sum(s);               %gamma=-e'*D*u
```

Note that the command line in the MATLAB code above:

```
 r=(speye(n+1)/nu+H'*H);
```

computes directly the factor $(\frac{I}{\nu}+H'H)^{-1}H'e$ of (15). This is a much more economical and stable than computing the inverse $(\frac{I}{\nu}+H'H)^{-1}$ explicitly then multiplying it by $H'e$. The calculations $H'e$ and $A's$ involve the transpose of typically large matrices which can be time consuming. Instead, we calculate `r=sum(H)'` and `w=(s'*A)'` respectively, the transposes of these vectors.

We further note that the MATLAB code above not only works for a linear classifier but also for a nonlinear classifier as well. In the nonlinear case, the matrix $K(A, A')$ is used as input instead of A and the pair (u, γ) is returned instead of (w, γ). The nonlinear separating surface is then given by (27) as:

$$K(x, A')u - \gamma = 0.$$

Rectangular kernels [17] can also be handled by this code. The the input then is the rectangular matrix $K(A, \bar{A}')$, where $\bar{A} \in R^{m \times k}$, $k << m$ and the given output is the pair (u, γ) with $u \in R^k$.

A final note regarding a further simplification of PSVM. If we substitute the expression (15) for u in (12), we obtain after some algebra the following simple expression for w and γ in terms of the problem data:

$$\begin{bmatrix} w \\ \gamma \end{bmatrix} = (\frac{I}{\nu} + E'E)^{-1}E'De, \tag{29}$$

where $E = DH$ and hence $H = DE$. Thus:

$$E = DH = [A \quad -e], \text{ and } H = DE = D[A \quad -e]. \tag{30}$$

This direct explicit solution of our PSVM problem (9) can be written as the following single line of MATLAB code, which also does not perform the explicit matrix inversion $(\frac{I}{\nu}+E'E)^{-1}$, and is slightly faster than the above MATLAB code:

```
[w  gamma]'=(I/nu+E'*E)\(diag(D)'*E)',
```

where, according to MATLAB commands, $diag(D)$ is an $m \times 1$ vector generated from the diagonal of the matrix D. Computational testing using this MATLAB code are slightly better than the ones reported in the tables below and are not given here. We comment further that the solution (29) can also be obtained directly from (9) by using the equality constraint to eliminate y from the problem and solving the resulting unconstrained minimization problem in the variables w and γ by setting to zero the gradients with respect to w and γ.

The datasets used for our numerical tests were the following:

- Seven publicly available datasets from the UCI Machine Learning Repository [27]: WPBC, Ionosphere, Cleveland Heart, Pima Indians, BUPA Liver, Mushroom, tic-tac-toe.
- The Census dataset is a version of the US Census Bureau "Adult" dataset, which is publicly available from Silicon Graphics' website [4].
- The Galaxy Dim dataset used in galaxy discrimination with neural networks from [29]
- Two large datasets (2 million points and 10 attributes) created using the NDC Data Generator [28].
- The Spiral dataset proposed by Alexis Wieland of the MITRE Corporation and available from the CMU Artificial Intelligence Repository [34].

We outline our computational results now in five groups as follows.

1. **Table 1: Comparison of seven different methods on the Adult dataset** In this experiment we compared the performance of seven different methods for linear classification on different sized versions of the Adult dataset. Reported results on the SOR [22], SMO [30] and SVM^{light} [16] are from [22]. Results for LSVM [24] results were computed here using "locop1", whereas SSVM [18] and RLP [2] are from [18]. The SMO experiments were run on a 266 MHz Pentium II processor under Windows NT 4 using Microsoft's Visual C++ 5.0 compiler. The SOR experiments were run on a 200 MHz Pentium Pro with 64 megabytes of RAM, also under Windows NT 4 and using Visual C++ 5.0. The SVM^{light} experiments were run on the same hardware as that for SOR, but under the Solaris 5.6 operating system. Bold type indicates the best result and a dash (-) indicates that the results were not available from [22]. Although the timing comparisons are approximate because of the different machines used, they do indicate that PSVM has a distinct edge in speed, e.g. solving the largest problem in 7.4 seconds, which is much faster than any other method. Times, training and testing ten-fold correctness are shown in Table 1.

2. **Table 2: Comparative performances of LSVM [24] and PSVM on a large dataset**

   Two large datasets consisting of 2 million points and 10 attributes were created using the NDC Data Generator [28]. One of them is called NDC-easy because it is highly linearly separable (around 90%). The other one is called NDC-hard since it has linear separability of around 70%. As is shown in Table 2 the linear classifiers obtained using both methods performed almost identically. Despite the 2 million size of the datasets, PSVM solved the problems in about 20 seconds each compared to LSVM's times of over 650 seconds. In contrast SVM^{light} [16, 24] failed on this problem.

3. **Table 3: Comparison of LSVM,SSVM and PSVM using a Linear Classifier**

   In this experiment we compared three methods: LSVM, SSVM and PSVM on six publicly available datasets from UCI Machine Learning Repository [27]. As shown in Table 3, the correctness of the three methods were very similar but the execution time including ten-fold cross validation for PSVM was smaller by as much as one order of magnitude or more than the other two methods tested. Since these three methods are all based on similar formulations of the classification problem the same value of ν was used for all of them.

4. **Figure 3: PSVM on the spiral dataset**

   We used a Gaussian kernel in order to classify the spiral dataset. This dataset consisting of 194 black and white points intertwined in the shape of a spiral is a synthetic dataset [34]. However, it apparently is a difficult test case for data mining algorithms and is known to give neural networks severe problems [15]. In contrast, a sharp separation was obtained using PSVM as can be seen in Figure 3.

5. **Table 4: Nonlinear Classifier Comparison using LSVM, SSVM and PSVM**

   For this experiment we chose four datasets from the UCI Machine Learning Repository for which it is known that a nonlinear classifier performs significantly better that a linear classifier. We used LSVM, SSVM and PSVM in order to find a Gaussian kernel based nonlinear classifier to classify the data. In all datasets tested in these experiments, the three methods performed similarly as far as ten-fold cross validation is concerned. However, execution time of PSVM was much smaller than that of other two methods. Note that for the mushroom dataset that consists of $m = 8124$ points with $n = 22$ attributes each, the square 8124×8124 kernel matrix does not fit into memory. In order to address this problem, we used a rectangular kernel with $\bar{A} \in R^{215 \times 8124}$ instead, as described in [17]. In general, our algorithm performed particularly well with a rectangular kernel since the system solved is of size $k \times k$, with $k << m$ and where k is the much smaller number of rows of $\bar{A}$. In contrast with a full square kernel the system to solve is of the size $m \times m$.

5. CONCLUSION AND FUTURE WORK

We have proposed an extremely simple procedure for generating linear and nonlinear classifiers based on proximity to one of two parallel planes that are pushed as far apart as possible. This procedure, a proximal support vector machine (PSVM), requires nothing more sophisticated than solving a simple system of linear equations, for either a linear or nonlinear classifier. In contrast, standard support vector machine classifiers require the more costly solution of a linear or quadratic program. For a linear classifier, all that is needed by PSVM is the inversion of a small matrix of the order of the input space dimension, typically of the order of 100 or less, even if there are millions of data points to classify. For a nonlinear classifier, a linear system of equations of the order of the number data points needs to be solved currently. This allows us to easily classify datasets with as many as a few thousand of points. For larger datasets, data selection and reduction methods such as [11, 17, 12] can be utilized and will be the subject of future work. Our computational results demonstrate that PSVM classifiers obtain test set correctness comparable to that of standard of SVM classifiers at a fraction of the time, sometimes an order of magnitude less.

Another avenue for future is that of incremental classification for large datasets. This appears particularly promising in view of the very simple explicit solutions (15) and (24) for the linear and nonlinear PSVM classifiers that can be updated incrementally as new data points come streaming in.

To sum up, the principal contribution of this work, is a very efficient classifier that requires no specialized software. PSVM can be easily incorporated into all sorts of data mining applications that require a fast, simple and effective classifier.

Acknowledgements
The research described in this Data Mining Institute Report 00-02, February 2001, was supported by National Science Foundation Grants CCR-9729842 and CDA-9623632, by Air Force Office of Scientific Research Grant F49620-00-1-0085 and by the Microsoft Corporation.

6. REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, Pennsylvania, third edition, 1999. http://www.netlib.org/lapack/.

[2] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[3] 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.

[4] US Census Bureau. Adult dataset. Publicly available from: www.sgi.com/Technology/mlc/db/.

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

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

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

[8] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[9] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–203, Cambridge, MA, 2000. MIT Press.

[10] 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.

[11] G. Fung and O. L. Mangasarian. Data selection for support vector machine classification. In R. Ramakrishnan and S. Stolfo, editors, *Proceedings KDD2000: 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.

[12] G. Fung, O. L. Mangasarian, and A. Smola. Minimal kernel classifiers. Technical Report 00-08, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, November 2000. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-08.ps.

[13] D. Gale. *The Theory of Linear Economic Models*. McGraw-Hill Book Company, New York, 1960.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.

Table 1: Testing set correctness and running times on the larger Adult dataset obtained by seven different methods using a linear classifier. Timing comparisons are approximate because of the different machines used, but they do indicate that PSVM has a distinct edge, e.g. solving the largest problem in 7.4 seconds, much faster than any other method. Best results are shown in bold.

Dataset size	Testing Correctness % Running Time *Sec.*						
(Training, Testing)	Method						
$n =$ no. of attributes	PSVM	LSVM	SSVM	SOR	SMO	SVM^{light}	RLP
$(1605, 30957)$	84.00	**84.27**	**84.27**	84.06	-	84.25	78.68
$n = 123$	**0.3**	3.3	1.9	**0.3**	0.4	5.4	9.9
$(2265, 30297)$	84.13	**84.66**	84.57	84.24	-	84.43	77.19
$n = 123$	**0.5**	5.0	2.8	1.2	0.9	10.8	19.12
$(3185, 29377)$	84.25	84.55	**84.63**	84.23	-	84.40	77.83
$n = 123$	**0.7**	8.1	3.9	1.4	1.8	21.0	80.1
$(4781, 27781)$	84.35	**84.55**	**84.55**	84.28	-	84.47	79.15
$n = 123$	**1.2**	8.1	6.0	1.6	3.6	43.2	88.6
$(6414, 26148)$	84.49	**84.68**	84.60	84.30	-	84.43	71.85
$n = 123$	**1.6**	18.8	8.1	4.1	5.5	87.6	218.8
$(11221, 21341)$	84.48	**84.84**	84.79	84.37	-	84.68	60.00
$n = 123$	**2.5**	38.9	14.1	18.8	17.0	306.6	449.2
$(16101, 16461)$	84.78	**85.01**	84.96	84.62	-	84.83	72.52
$n = 123$	**3.7**	60.5	21.5	24.8	35.3	667.2	632.6
$(22697, 9865)$	85.16	**85.35**	**85.35**	85.06	-	85.17	77.43
$n = 123$	**5.2**	92.0	29.0	31.3	85.7	1425.6	991.9
$(32562, 16282)$	84.56	**85.05**	85.02	84.96	-	**85.05**	83.25
$n = 123$	**7.4**	140.9	44.5	83.9	163.6	2184.0	1561.1

Table 2: LSVM and PSVM performance on two, 2 million point NDC datasets with 10-attributes. A linear classifier with parameter $\nu = 0.1$ was used in both methods on the same locop1 machine.

Method	Dataset	Training Correctness %	Testing Correctness %	Time (CPU) sec.
LSVM	NDC-easy	90.86	91.23	658.5
PSVM	NDC-easy	90.80	91.13	20.8
LSVM	NDC-hard	69.80	69.44	655.6
PSVM	NDC-hard	69.84	69.52	20.6

Table 3: PSVM, SSVM and LSVM training and ten-fold testing correctness and running times using a linear classifier. Execution times include training and ten-fold testing. Same value of ν was used in all the methods. Best results are in bold.

Data Set $m \times n$	PSVM Train Test Time (Sec.)	SSVM Train Test Time (Sec.)	LSVM Train Test Time (Sec.)
WPBC (60 mo.)	**70.8%**	**70.8%**	**70.8%**
110 × 32	**68.5%**	**68.5%**	**68.5%**
	0.01	0.17	0.53
Ionosphere	90.6%	**92.8 %**	**92.8 %**
351 × 34	87.3%	**88.3 %**	**88.3 %**
	0.17	0.96	1.76
Cleveland Heart	87.00%	**87.3%**	**87.3%**
297 × 13	85.90%	**86.2%**	**86.2%**
	0.02	6.53	1.06
Pima Indians	**77.7%**	**77.7%**	**77.7%**
768 × 8	**77.3%**	**77.3%**	**77.3%**
	0.02	0.64	2.36
BUPA Liver	70.2%	**70.5%**	**70.5%**
345 × 6	69.1%	**70.1%**	**70.1%**
	0.02	0.53	1.265
Galaxy Dim	93.8%	**95.0%**	**95.0%**
4192 × 14	93.7%	**94.8%**	**94.8%**
	0.52	5.25	22.26

Table 4: PSVM, SSVM and LSVM training and ten-fold testing correctness and running times using a nonlinear classifier. Execution times include training and ten-fold testing. Same value of ν was used in all the methods. Best results are in bold.

Data Set $m \times n$	PSVM Train Test Time (Sec.)	SSVM Train Test Time (Sec.)	LSVM Train Test Time (Sec.)
Ionosphere	98.0%	**98.8 %**	**99.0 %**
351 × 34	94.9%	**95.5 %**	92.6%
	3.26	21.25	14.58
BUPA Liver	76.2%	76.5%	**79.0%**
345 × 6	73.4%	**74.3%**	71.4
	3.38	19.70	14.15
Tic-tac-toe	98.0%	98.0%	**98.2%**
958 × 9	**98.4%**	**98.4%**	94.7%
	60.11	316.13	137.35
Mushroom [*]	88.3%	**89.0%**	[**]
8124 × 22	88.3%	**88.9%**	-
	34.80	389.34	-

[*] A Rectangular kernel [18] of the size 8124 × 215 was used here instead of the square 8124 × 8124 kernel which does not fit into memory.

[**] Not applicable here since the LSVM algorithm does not support rectangular kernels.

[15] J. Gracke, M. Griebel, and M. Thess. Data mining with sparse grids. Technical report, Institut für Angrwandte Mathematik, Universitaät Bonn, Bonn, Germany, 2000. http://wissrech.iam.uni-bonn.de/research/projects/garcke/sparsemining.html.

[16] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.

[17] 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, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

[18] Yuh-Jye Lee and O. L. Mangasarian. SSVM: A smooth support vector machine. Technical Report 99-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, September 1999. *Computational Optimization and Applications* 20(1), October 2001, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-03.ps.

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

[20] 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.

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

[22] O. L. Mangasarian and D. 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] O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. Technical Report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, April 2000. *Machine Learning*, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

[24] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. Technical Report 00-06, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, June 2000. *Journal of Machine Learning Research*, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.

[25] O. L. Mangasarian and T.-H. Shiau. Lipschitz continuity of solutions of linear inequalities, programs and complementarity problems. *SIAM Journal on Control and Optimization*, 25(3):583–595, May 1987.

[26] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.s

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

[28] D. R. Musicant. NDC: normally distributed clustered datasets, 1998. www.cs.wisc.edu/∼musicant/data/ndc/.

[29] S. Odewahn, E. Stockwell, R. Pennington, R. Hummphreys, and W. Zumach. Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103(1):318–331, 1992.

[30] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999. http://www.research.microsoft.com/∼jplatt/smo.html.

[31] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill–Posed Problems*. John Wiley & Sons, New York, 1977.

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

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

[34] Alexis Wieland. Twin spiral dataset. http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/neural/bench/cmu/0.html.