Journal of Machine Learning Research 1 (2001) 161-177 Submitted 8/00; Published 3/01

Lagrangian Support Vector Machines

O. L. Mangasarian OLVI@CS.WISC.EDU
Computer Sciences Department
University of Wisconsin
Madison, WI 53706, USA

David R. Musicant DMUSICAN@CARLETON.EDU
Department of Mathematics and Computer Science
Carleton College
Northfield, MN 55057, USA

Editor: Bernhard Schölkopf (for HTTP://WWW.KERNEL-MACHINES.ORG)

Abstract

An implicit Lagrangian for the dual of a simple reformulation of the standard quadratic program of a linear support vector machine is proposed. This leads to the minimization of an **unconstrained** differentiable convex function in a space of dimensionality equal to the number of classified points. This problem is solvable by an extremely simple linearly convergent Lagrangian support vector machine (LSVM) algorithm. LSVM requires the inversion at the outset of a single matrix of the order of the much smaller dimensionality of the original input space plus one. The full algorithm is given in this paper in 11 lines of MATLAB code without any special optimization tools such as linear or quadratic programming solvers. This LSVM code can be used "as is" to solve classification problems with millions of points. For example, 2 million points in 10 dimensional input space were classified by a linear surface in 82 minutes on a Pentium III 500 MHz notebook with 384 megabytes of memory (and additional swap space), and in 7 minutes on a 250 MHz UltraSPARC II processor with 2 gigabytes of memory. Other standard classification test problems were also solved. Nonlinear kernel classification can also be solved by LSVM. Although it does not scale up to very large problems, it can handle any positive semidefinite kernel and is guaranteed to converge. A short MATLAB code is also given for nonlinear kernels and tested on a number of problems.

1. Introduction

Support vector machines (SVMs) (Vapnik, 1995, Cherkassky and Mulier, 1998, Bradley and Mangasarian, 2000, Mangasarian, 2000, Lee and Mangasarian, 2000) are powerful tools for data classification. Classification is achieved by a linear or nonlinear separating surface in the input space of the dataset. In this work we propose a very fast simple algorithm, based on an an implicit Lagrangian formulation (Mangasarian and Solodov, 1993) of the dual of a simple reformulation of the standard quadratic program of a linear support vector machine. This leads to the minimization of an **unconstrained** differentiable convex function in an m-dimensional space where m is the number of points to be classified in a given n dimensional input space. The necessary optimality condition for this unconstrained minimization problem can be transformed into a very simple symmetric positive definite

©2001 O. L. Mangasarian and David R. Musicant.

complementarity problem (12). A linearly convergent iterative Lagrangian support vector machine (LSVM) Algorithm 1 is given for solving (12). LSVM requires the inversion at the outset of a single matrix of the order of the dimensionality of the original input space plus one: $(n+1)$. The algorithm can accurately solve problems with millions of points and requires only standard native MATLAB commands without any optimization tools such as linear or quadratic programming solvers.

As was the case in our recent active set support vector machine (ASVM) approach (Mangasarian and Musicant, 2000), the following two simple changes were made to the standard linear SVM: (i) The margin (distance) between the parallel bounding planes was maximized with respect to both orientation (w) as well as location relative to the origin (γ). Such a change was also carried out in our successive overrelaxation (SOR) approach (Mangasarian and Musicant, 1999) as well as in the smooth support vector machine (SSVM) approach of Lee and Mangasarian (2000). See equation (7) and Figure 1. (ii) The error in the soft margin (y) was minimized using the 2-norm squared instead of the conventional 1-norm. See equation (7). These straightforward, but important changes, lead to a considerably simpler positive definite dual problem with nonnegativity constraints only. See equation (8). Although our use of a quadratic penalty for the soft margin as indicated in (ii) above is nonstandard, it has been used effectively in previous work (Lee and Mangasarian, 2000, 2001, Mangasarian and Musicant, 2000). In theory, this change could lead to a less robust classifier in the presence of outliers. However, this modification to the SVM does not seem to have a detrimental effect, as can be seen in the experimental results of Section 5.

In Section 2 of the paper we begin with the standard SVM formulation and its dual and then give our formulation and its dual. We note that computational evidence given in our previous work (Mangasarian and Musicant, 2000) shows that this alternative formulation does not compromise on generalization ability. Section 3 gives our simple iterative Lagrangian support vector machine (LSVM) Algorithm 1 and establishes its global linear convergence. LSVM, stated in 11 lines of MATLAB Code 2 below, solves **once** at the outset a single system of $n+1$ equations in $n+1$ variables given by a symmetric positive definite matrix. It then uses a linearly convergent iterative method to solve the problem. In Section 4 LSVM is extended to positive semidefinite nonlinear kernels. Section 5 describes our numerical results which show that LSVM gives comparable or better testing results than those of other SVM algorithms, and in some cases is dramatically faster than a standard quadratic programming SVM solver.

We now describe our notation and give some background material. All vectors will be column vectors unless transposed to a row vector by a prime $'$. For a vector x in the n-dimensional real space R^n, x_+ denotes the vector in R^n with all of its negative components set to zero. This corresponds to projecting x onto the nonnegative orthant. The base of the natural logarithms will be denoted by ε , and for a vector $y \in R^m$, ε^{-y} will denote a vector in R^m with components ε^{-y_i}, $i = 1, \ldots, m$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix A' will denote the transpose of A, A_i will denote the i-th row of A and $A_{\cdot j}$ will denote the j-th column of A. A vector of ones or zeroes in a real space of arbitrary dimension will be denoted by e or 0, respectively. The identity matrix of arbitrary dimension will be denoted by I. For two vectors x and y in R^n, $x \perp y$ denotes orthogonality, that is $x'y = 0$. We use $:=$ to denote definition. The 2-norm of a vector x and a matrix Q will be denoted by $\|x\|$ and $\|Q\|$ respectively. A separating plane, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n, is a plane that attempts to separate R^n into two halfspaces such that each open halfspace contains points mostly of $\mathcal{A}$ or $\mathcal{B}$. A special case of the Sherman-Morrison-Woodbury (SMW) identity (Golub and Van Loan, 1996) will be utilized:

$$(\frac{I}{\nu} + HH')^{-1} = \nu(I - H(\frac{I}{\nu} + H'H)^{-1}H'), \tag{1}$$

where ν is a positive number and H is an arbitrary $m \times k$ matrix. This identity, easily verifiable by premultiplying both sides by $(\frac{I}{\nu} + HH')$, enables us to invert a large $m \times m$ matrix of the form in (1) by merely inverting a smaller $k \times k$ matrix. It was also used recently in our ASVM paper (Mangasarian and Musicant, 2000) as well as by Ferris and Munson (2000).

2. The Linear Support Vector Machine

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 class $\mathcal{A}+$ or $\mathcal{A}-$ 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 (Vapnik, 1995, Cherkassky and Mulier, 1998) is given by the following quadratic program with parameter $\nu > 0$:

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

Here w is the normal to the bounding planes:

$$x'w = \gamma \pm 1 \tag{3}$$

and γ determines their location relative to the origin. See Figure 1. The plane $x'w = \gamma + 1$ bounds the class $\mathcal{A}+$ points, possibly with some error, and the plane $x'w = \gamma - 1$ bounds the class $\mathcal{A}-$ points, also possibly with some error. The linear separating surface is the plane:

$$x'w = \gamma, \tag{4}$$

midway between the bounding planes (3). The quadratic term in (2) is twice the reciprocal of the square of the 2-norm distance $2/\|w\|$ between the two bounding planes of (3) (see Figure 1). This term enforces maximization of this distance, which is often called the "margin". If the classes are linearly inseparable, as depicted in Figure 1, then the two planes bound the two classes with a "soft margin". That is, they bound each set approximately with some error determined by the nonnegative error variable y:

$$\begin{array}{ccccl} A_iw & + & y_i & \geq & \gamma + 1, \ \text{for } D_{ii} = 1, \\ A_iw & - & y_i & \leq & \gamma - 1, \ \text{for } D_{ii} = -1. \end{array} \tag{5}$$

Traditionally the constant ν in (2) multiplies the the 1-norm of the error variable y and acts as a weighting factor. A nonzero y results in an approximate separation as depicted in Figure 1. The dual to the standard quadratic linear SVM (2) (Mangasarian, 1994, Schölkopf

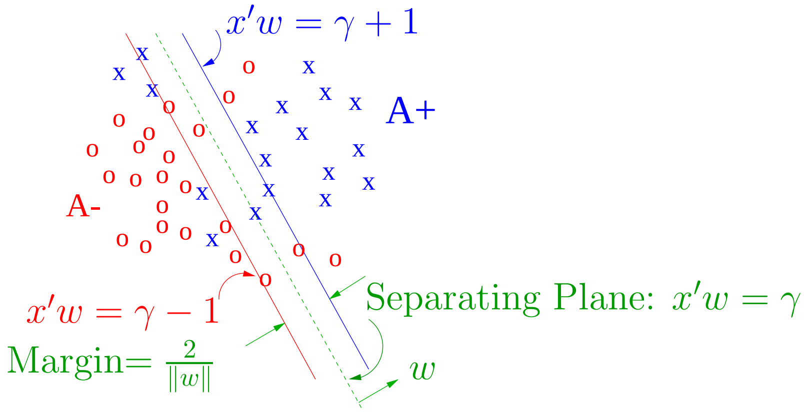

Figure 1: The bounding planes of a linear SVM with a soft margin (i.e. with some errors), and the separating plane approximately separating $\mathcal{A}+$ from $\mathcal{A}-$.

et al., 1999, Mangasarian, 2000, Cristianini and Shawe-Taylor, 2000) is the following:

$$\min_{u \in R^m} \frac{1}{2} u' DAA'Du - e'u \ \text{ s.t. } e'Du = 0, \ \ 0 \leq u \leq \nu e. \tag{6}$$

The variables (w, γ) of the primal problem which determine the separating surface (4) can be obtained from the solution of the dual problem above (Mangasarian and Musicant, 1999, Eqns. 5 and 7). We note immediately that the matrix $DAA'D$ appearing in the dual objective function (6) is not positive definite in general because typically $m >> n$. Also, there is an equality constraint present, in addition to bound constraints, which for large problems necessitates special computational procedures such as SMO (Platt, 1999) or SVM^{light} (Joachims, 1999). Furthermore, a one-dimensional optimization problem (Mangasarian and Musicant, 1999) must be solved in order to determine the locator γ of the separating surface (4). In order to overcome all these difficulties as well as that of dealing with the necessity of having to essentially invert a very large matrix of the order of $m \times m$, we propose the following simple but critical modifications to the standard SVM formulation (2). We change the 1-norm of y to a 2-norm squared which makes the constraint $y \geq 0$ redundant. We also append the term γ^2 to $w'w$. This in effect maximizes the margin between the parallel separating planes (3) by optimizing with respect to both w and γ (Mangasarian and Musicant, 1999), that is with respect to both orientation and location of the planes, rather that just with respect to w which merely determines the orientation of the plane. This leads to the following reformulation of the SVM:

$$\min_{(w,\gamma,y) \in R^{n+1+m}} \nu \frac{y'y}{2} + \frac{1}{2}(w'w + \gamma^2) \ \text{ s.t. } D(Aw - e\gamma) + y \geq e. \tag{7}$$

The dual of this problem is (Mangasarian, 1994):

$$\min_{0 \leq u \in R^m} \frac{1}{2} u' (\frac{I}{\nu} + D(AA' + ee')D)u - e'u. \tag{8}$$

The variables (w, γ) of the primal problem which determine the separating surface (4) are recovered directly from the solution of the dual (8) above by the relations:

$$w = A'Du, \ \ y = \frac{u}{\nu}, \ \ \gamma = -e'Du. \tag{9}$$

We immediately note that the matrix appearing in the dual objective function is positive definite and that there is no equality constraint and no upper bound on the dual variable u. The only constraint present is a nonnegativity one. These facts lead us to our simple iterative Lagrangian SVM Algorithm which requires the inversion of a positive definite $(n+1) \times (n+1)$ matrix, at the beginning of the algorithm followed by a straightforward linearly convergent iterative scheme that requires no optimization package.

3. LSVM (Lagrangian Support Vector Machine) Algorithm

Before stating our algorithm we define two matrices to simplify notation as follows:

$$H = D[A \quad -e], \qquad Q = \frac{I}{\nu} + HH'. \tag{10}$$

With these definitions the dual problem (8) becomes

$$\min_{0 \le u \in R^m} f(u) := \frac{1}{2}u'Qu - e'u. \tag{11}$$

It will be understood that within the LSVM Algorithm, the single time that Q^{-1} is computed at the outset of the algorithm, the SMW identity (1) will be used. Hence only an $(n+1) \times (n+1)$ matrix is inverted.

The LSVM Algorithm is based directly on the Karush-Kuhn-Tucker necessary and sufficient optimality conditions (Mangasarian, 1994, KTP 7.2.4, page 94) for the dual problem (11):

$$0 \le u \ \perp \ Qu - e \ge 0. \tag{12}$$

By using the easily established identity between any two real numbers (or vectors) a and b:

$$0 \le a \ \perp \ b \ge 0 \iff a = (a - \alpha b)_+, \ \alpha > 0, \tag{13}$$

the optimality condition (12) can be written in the following equivalent form for any positive α:

$$Qu - e = ((Qu - e) - \alpha u)_+. \tag{14}$$

These optimality conditions lead to the following very simple iterative scheme which constitutes our LSVM Algorithm:

$$u^{i+1} = Q^{-1}(e + ((Qu^i - e) - \alpha u^i)_+), \ i = 0, 1, \ldots, \tag{15}$$

for which we will establish global linear convergence from any starting point under the easily satisfiable condition:

$$0 < \alpha < \frac{2}{\nu}. \tag{16}$$

We impose this condition as $\alpha = 1.9/\nu$ in all our experiments, where ν is the parameter of our SVM formulation (7). It turns out, and this is the way that led us to this iterative scheme, that the optimality condition (14), is also the necessary and sufficient condition for the unconstrained minimum of the implicit Lagrangian (Mangasarian and Solodov, 1993) associated with the dual problem (11):

$$\min_{u \in R^m} L(u, \alpha) = \min_{u \in R^m} \frac{1}{2}u'Qu - e'u + \frac{1}{2\alpha}(\|(-\alpha u + Qu - e)_+\|^2 - \|Qu - e\|^2). \tag{17}$$

Setting the gradient with respect to u of this convex and differentiable Lagrangian to zero gives

$$(Qu - e) + \frac{1}{\alpha}(Q - \alpha I)((Q - \alpha I)u - e)_+ - \frac{1}{\alpha}Q(Qu - e) = 0, \tag{18}$$

or equivalently:

$$(\alpha I - Q)((Qu - e) - ((Q - \alpha I)u - e)_+) = 0, \tag{19}$$

which is equivalent to the optimality condition (14) under the assumption that α is positive and not an eigenvalue of Q.

We establish now the global linear convergence of the iteration (15) under condition (16).

Algorithm 1 LSVM Algorithm & Its Global Convergence *Let $Q \in R^{m\times m}$ be the symmetric positive definite matrix defined by (10) and let (16) hold. Starting with an arbitrary $u^0 \in R^m$, the iterates u^i of (15) converge to the unique solution $\bar{u}$ of (11) at the linear rate:*

$$\|Qu^{i+1} - Q\bar{u}\| \leq \|I - \alpha Q^{-1}\| \cdot \|Qu^i - Q\bar{u}\|. \tag{20}$$

Proof Because $\bar{u}$ is the solution of (11), it must satisfy the optimality condition (14) for any $\alpha > 0$. Subtracting that equation with $u = \bar{u}$ from the iteration (15) premultiplied by Q and taking norms gives:

$$\|Qu^{i+1} - Q\bar{u}\| = \|(Qu^i - e - \alpha u^i)_+ - (Q\bar{u} - e - \alpha\bar{u})_+\| \tag{21}$$

Using the Projection Theorem (Bertsekas, 1999, Proposition 2.1.3) which states that the distance between any two points in R^m is not less than the distance between their projections on any convex set (the nonnegative orthant here) in R^m, the above equation gives:

$$\begin{aligned} \|Qu^{i+1} - Q\bar{u}\| &\leq \|(Q - \alpha I)(u^i - \bar{u})\| \\ &\leq \|I - \alpha Q^{-1}\| \cdot \|Q(u^i - \bar{u})\|. \end{aligned} \tag{22}$$

All we need to show now is that $\|I - \alpha Q^{-1}\| < 1$. This follows from (16) as follows. Noting the definition (10) of Q and letting λ_i, $i = 1, \ldots, m$, denote the nonnegative eigenvalues of HH', all we need is:

$$-1 < 1 - \alpha(\frac{1}{\nu} + \lambda_i)^{-1} < 1, \tag{23}$$

or equivalently:

$$2 > \alpha(\frac{1}{\nu} + \lambda_i)^{-1} > 0, \tag{24}$$

which is satisfied under the assumption (16).□

We give now a complete MATLAB (The MathWorks, Inc., 1994-2001) code of LSVM which is capable of solving problems with millions of points using only native MATLAB commands. The input parameters, besides A, D and ν of (10), which define the problem, are: itmax, the maximum number of iterations and tol, the tolerated nonzero error in $\|u^{i+1} - u^i\|$ at termination. The quantity $\|u^{i+1} - u^i\|$ bounds from above:

$$\|Q\|^{-1} \cdot \|Qu^i - e - ((Qu^i - e) - \alpha u^i)_+\|, \tag{25}$$

which measures the violation of the optimality criterion (14). It follows (Mangasarian and Ren, 1994) that $\|u^{i+1} - u^i\|$ also bounds $\|u^i - \bar{u}\|$, and by (9) it also bounds $\|w^i - \bar{w}\|$ and $|\gamma^i - \bar{\gamma}|$, where $(\bar{w}, \bar{\gamma}, \bar{y})$ is the unique solution of the primal SVM (7).

Code 2 LSVM MATLAB M-File

```
function [it, opt, w, gamma] = svml(A,D,nu,itmax,tol)
% lsvm with SMW for min 1/2*u'*Q*u-e'*u s.t. u=>0,
% Q=I/nu+H*H', H=D[A -e]
% Input: A, D, nu, itmax, tol; Output: it, opt, w, gamma
% [it, opt, w, gamma] = svml(A,D,nu,itmax,tol);
  [m,n]=size(A);alpha=1.9/nu;e=ones(m,1);H=D*[A -e];it=0;
  S=H*inv((speye(n+1)/nu+H'*H));
  u=nu*(1-S*(H'*e));oldu=u+1;
  while it<itmax & norm(oldu-u)>tol
    z=(1+pl(((u/nu+H*(H'*u))-alpha*u)-1));
    oldu=u;
    u=nu*(z-S*(H'*z));
    it=it+1;
  end;
  opt=norm(u-oldu);w=A'*D*u;gamma=-e'*D*u;

function pl = pl(x); pl = (abs(x)+x)/2;
```

4. LSVM for Nonlinear Kernels

In this section of the paper we show how LSVM can be used to solve classification problems with positive semidefinite nonlinear kernels. Algorithm 1 and its convergence can be extended for such nonlinear kernels as we show below. The only price paid for this extension is that problems with large datasets can be handled using the Sherman-Morrison-Woodbury (SMW) identity (1) only if the inner product terms of the kernel (Mangasarian, 2000, Equation (3)) are explicitly known, which in general they are not. Nevertheless LSVM may be a useful tool for classification with nonlinear kernels because of its extreme simplicity as we demonstrate below with the simple MATLAB code for which it does not make use of the Sherman-Morrison-Woodbury identity nor any optimization package.

We shall use the notation of Mangasarian (2000). 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 $R^{m \times \ell}$. A typical kernel is the Gaussian kernel $\varepsilon^{-\mu\|A'_i - B_{\cdot j}\|^2}$, $i, j = 1, \ldots, m$, $\ell = m$, where ε is the base of natural logarithms, while a linear kernel is $K(A, B) = AB$. For a column vector x in R^n, $K(x', A')$ is a row vector in R^m, and the linear separating surface (4) is replaced by the nonlinear surface

$$K([x' \quad -1], \begin{bmatrix} A' \\ -e' \end{bmatrix})Du = 0, \tag{26}$$

where u is the solution of the dual problem (11) with Q re-defined for a general nonlinear kernel as follows:

$$G = [A \quad -e], \qquad Q = \frac{I}{\nu} + DK(G, G')D. \tag{27}$$

Note that the nonlinear separating surface (26) degenerates to the linear one (4) if we let $K(G, G') = GG'$ and make use of (9).

To justify the nonlinear kernel formulation (27) we refer the reader to (Mangasarian, 2000, Equation (8.9)) for a similar result and give a brief derivation here. If we rewrite our dual problem for a linear kernel (8) in the equivalent form:

$$\min_{0 \leq u \in R^m} \frac{1}{2} u' (\frac{I}{\nu} + DGG'D) u - e'u, \tag{28}$$

and replace the linear kernel GG' by a general nonlinear positive semidefinite symmetric kernel $K(G, G')$ we obtain:

$$\min_{0 \leq u \in R^m} \frac{1}{2} u' (\frac{I}{\nu} + DK(G, G')D) u - e'u. \tag{29}$$

This is the formulation given above in (27). We note that the Karush-Kuhn-Tucker necessary and sufficient optimality conditions for this problem are:

$$0 \leq u \quad \perp \quad (\frac{I}{\nu} + DK([A \quad -e], \begin{bmatrix} A' \\ -e' \end{bmatrix})D)u - e \geq 0, \tag{30}$$

which underly LSVM for a nonlinear positive semidefinite kernel $K(G, G')$. The positive semidefiniteness of the nonlinear kernel $K(G, G')$ is needed in order to ensure the existence of a solution to both (29) and (30).

All the results of the previous section remain valid, with Q re-defined as above for any positive semidefinite kernel K. This includes the iterative scheme (15) and the convergence result given under the Algorithm 1. However, because we do not make use of the Sherman-Morrison-Woodbury identity for a nonlinear kernel, the LSVM MATLAB Code 3 is somewhat different and is as follows:

Code 3 LSVM MATLAB M-File for Nonlinear Kernel $K(\cdot,\cdot)$

```
function [it, opt,u] = svmlk(nu,itmax,tol,D,KM)
% lsvm with nonlinear kernel for min 1/2*u'*Q*u-e'*u s.t. u=>0
% Q=I/nu+DK(G,G')D, G=[A -e]
% Input: nu, itmax, tol, D,  KM=K(G,G')
% [it, opt,u] = svmlk(nu,itmax,tol,D,KM);
  m=size(KM,1);alpha=1.9/nu;e=ones(m,1);I=speye(m);it=0;
  Q=I/nu+D*KM*D;P=inv(Q);
  u=P*e;oldu=u+1;
  while it<itmax & norm(oldu-u)>tol
    oldu=u;
    u=P*(1+pl(Q*u-1-alpha*u));
    it=it+1;
  end;
  opt=norm(u-oldu);[it opt]

function pl = pl(x); pl = (abs(x)+x)/2;
```

Since we cannot use the Sherman-Morrison-Woodbury identity in the general nonlinear case, we note that this code for a nonlinear kernel is effective for moderately sized problems. The sizes of the matrices P and Q in Code 3 scale quadratically with the number of data points.

5. Numerical Implementation and Comparisons

The implementation of LSVM is straightforward, as shown in the previous sections. Our first "dry run" experiments were on randomly generated problems just to test the speed and effectiveness of LSVM on large problems. We first used a Pentium III 500 MHz notebook with 384 megabytes of memory (and additional swap space) on 2 million randomly generated points in R^{10} with $\nu = \frac{1}{m}$ and $\alpha = \frac{1.9}{\nu}$. LSVM solved the problem in 6 iterations in 81.52 minutes to an optimality criterion of $9.398e-5$ on a 2-norm violation of (14). The same problem was solved in the same number of iterations and to the same accuracy in 6.74 minutes on a 250 MHz UltraSPARC II processor with 2 gigabytes of memory. After these preliminary encouraging tests we proceeded to more systematic numerical tests as follows.

Most of the rest of our experiments were run on the Carleton College workstation "gray", which utilizes a 700 MHz Pentium III Xeon processor and a maximum of 2 Gigabytes of memory available for each process. This computer runs Redhat Linux 6.2, with MATLAB 6.0. The one set of experiments showing the checkerboard were run on the UW-Madison Data Mining Institute "locop2" machine, which utilizes a 400 MHz Pentium II Xeon and a maximum of 2 Gigabytes of memory available for each process. This computer runs Windows NT Server 4.0, with MATLAB 5.3.1. Both gray and locop2 are multiprocessor machines. However, only one processor was used for all the experiments shown here as MATLAB is a single threaded application and does not distribute any effort across processors (The MathWorks, Inc., 2000). We had exclusive access to these machines, so there were no issues with other users inconsistently slowing the computers down.

The first results we present are designed to show that our reformulation of the SVM (7) and its associated algorithm LSVM yield similar performance to the standard SVM (2), referred to here as SVM-QP. Results are also shown for our active set SVM (ASVM) algorithm (Mangasarian and Musicant, 2000). For six datasets available from the UCI Machine Learning Repository (Murphy and Aha, 1992), we performed tenfold cross validation in order to compare test set accuracies between the methodologies. Furthermore, we utilized a tuning set for each algorithm to find the optimal value of the parameter ν. For both LSVM and ASVM, we used an optimality tolerance of 0.001 to determine when to terminate. SVM-QP was implemented using the high-performing CPLEX barrier quadratic programming solver (ILOG, Inc., 1999) with its default stopping criterion. Altering the CPLEX default stopping criterion to match that of LSVM did not result in significant change in timing relative to LSVM, but did reduce test set correctness for SVM-QP.

We also tested SVM-QP with the well-known optimized algorithm SVM^{light} (Joachims, 1998). Joachims, the author of SVM^{light}, graciously provided us with the newest version of the software (Version 3.10b) and advice on setting the parameters. All features for all experiments were normalized to the range $[-1, +1]$ as recommended in the SVM^{light} documentation. Such normalization was used in all experiments that we present, and is recommended for the LSVM algorithm due to the penalty on the bias parameter γ which we have added to our objective (7). We chose to use the default termination error criterion in SVM^{light} of 0.001. We also present an estimate of how much memory each algorithm used, as well as the number of support vectors. The number of support vectors was determined here as the number of components of the dual vector u that were nonzero. Moreover, we also present the number of components of u larger than 0.001 (referred to in the table as "SVs with tolerance").

The results demonstrate that LSVM performs comparably to SVM-QP with respect to generalizability, and shows running times comparable to or better than SVM^{light}. LSVM and ASVM show nearly identical generalization performance, as they solve the same optimization problem. Any differences in generalization observed between the two are only due to different approximate solutions being found at termination. LSVM and ASVM usually run in roughly the same amount of time, though there are some cases where ASVM is noticeably faster than LSVM. Nevertheless, this is impressive performance on the part of LSVM, which is a dramatically simpler algorithm than ASVM, CPLEX, and SVM^{light}. LSVM utilizes significantly less memory than the SVM-QP algorithms as well, though it does require more support vectors than SVM-QP. We address this point in the conclusion.

We next compare LSVM with SVM^{light} on the Adult dataset (Murphy and Aha, 1992), which is commonly used to compare SVM algorithms (Platt, 1999, Mangasarian and Musicant, 1999). These results, shown in Table 2, demonstrate that for the largest training sets LSVM performs faster than SVM^{light} with similar test set accuracies. Note that the "iterations" column takes on very different meanings for the two algorithms. SVM^{light} defines an iteration as solving an optimization problem over a a small number, or "chunk," of constraints. LSVM, on the other hand, defines an iteration as a matrix calculation which updates all the dual variables simultaneously. These two numbers are not directly comparable, and are included here only for purposes of monitoring scalability. In this experiment, LSVM uses significantly more memory than SVM^{light}.

Dataset m x n	Algorithm	Training Correctness	Testing Correctness	Time (CPU sec)	Memory (Kilobytes)	# of SVs	# of SVs (w/ tolerance)
Liver Disorders	CPLEX	70.76%	68.41%	3.806	4,427	310.5	268.7
	SVMlight	70.76%	68.41%	0.252	1,451	225.7	225.7
	ASVM	70.14%	68.68%	0.007	2	299.4	299.4
345 x 6	LSVM	70.14%	68.68%	*0.053	5	305.3	299.4
Cleveland Heart	CPLEX	87.73%	85.91%	2.453	3,406	267.3	96.3
	SVMlight	87.73%	85.91%	0.057	1,376	96.1	96.1
	ASVM	87.36%	85.89%	0.011	0	169.7	169.5
297 x 13	LSVM	87.36%	85.89%	*0.048	0	220.5	169.5
Pima Diabetes	CPLEX	77.36%	76.95%	90.360	20,978	687.8	454.8
	SVMlight	77.36%	76.95%	0.121	1,364	454.8	454.5
	ASVM	78.04%	78.12%	0.023	41	610.0	610.0
768 x 8	LSVM	*78.04%	78.12%	0.126	207	651.1	610.0
Ionosphere	CPLEX	92.81%	88.60%	4.335	13,230	301.7	98.8
	SVMlight	92.81%	88.60%	0.161	1,340	98.4	98.4
	ASVM	93.29%	87.75%	0.070	193	167.0	166.8
351 x 34	LSVM	93.29%	87.75%	*0.089	183	216.6	166.8
Tic Tac Toe	CPLEX	65.34%	65.34%	178.844	32,681	862.2	862.2
	SVMlight	65.34%	65.34%	0.208	1,344	608.3	608.2
	ASVM	70.27%	69.72%	0.015	136	862.2	862.2
958 x 9	LSVM	*70.27%	69.72%	*0.004	140	862.2	862.2
Votes	CPLEX	96.02%	95.85%	9.929	6,936	391.5	57.7
	SVMlight	96.02%	95.85%	0.038	1,344	57.8	57.4
	ASVM	96.73%	96.07%	0.025	69	128.8	123.1
435 x 16	LSVM	*96.73%	96.07%	0.047	245	245.4	123.2

Table 1: LSVM compared with conventional SVM-QP (CPLEX and SVM^{light}) and ASVM on six UCI datasets. LSVM test correctness is comparable to SVM-QP, with timing much faster than CPLEX and faster than or comparable to SVM^{light}. An asterisk in the first three columns indicates that the LSVM results are significantly different from SVM^{light} at significance level $\alpha = 0.05$. The column "w/ tolerance" indicates the number of support vectors with dual variable $u > 0.001$.

Training Set Size	CPU Sec	Iterations	Test Set Accuracy	Memory (Kilobytes)	# of SVs	# of SVs (w/ tolerance)
	SVM^{light}	SVM^{light}	SVM^{light}	SVM^{light}	SVM^{light}	SVM^{light}
	LSVM	LSVM	LSVM	LSVM	LSVM	LSVM
1605	0.3	256	84.05%	1340	708	706
	1.4	38	84.27%	2816	1364	1069
2265	0.5	389	84.37%	1340	1012	1011
	2.1	40	84.66%	4672	1924	1540
3185	0.8	495	84.22%	2444	1316	1312
	3.2	43	84.55%	6148	2662	2040
4781	1.5	724	84.33%	3388	1906	1904
	5.3	46	84.55%	9044	3933	3026
6414	2.7	966	84.47%	3684	2490	2488
	7.6	47	84.68%	12584	5220	3985
11221	11.6	1588	84.58%	5536	4222	4220
	15.3	50	84.84%	22368	9008	6879
16101	28.2	2285	84.81%	7508	6003	5999
	23.8	52	85.01%	32544	12935	9803
22697	55.9	3389	85.16%	10212	8385	8381
	36.5	54	85.35%	44032	18048	13721
32562	112.7	4850	85.03%	15500	11782	11777
	56.1	55	85.05%	70416	25769	19448

Table 2: Comparison of LSVM with SVM^{light} on the UCI adult dataset. LSVM test correctness is comparable to that of SVM^{light}, but is faster on large datasets. The column "w/ tolerance" indicates the number of support vectors with dual variable $u > 0.001$. ($\nu = 0.03$)

# of Points	# of Attributes	Iterations	Training Correctness	Testing Correctness	Time (CPU min)	Memory (Kilobytes)
2 million	10	81	69.80%	69.44%	22.4	918,256

Table 3: Performance of LSVM on NDC generated dataset ($\nu = 0.1$). SVM^{light} failed on this dataset.

Table 3 shows results from running LSVM on a massively sized dataset. This dataset was created using our own NDC Data Generator (Musicant, 1998) as suggested by Usama Fayyad. The results show that LSVM can be used to solve massive problems quickly, which is particularly intriguing given the simplicity of the LSVM Algorithm. Note that for these experiments, all the data was brought into memory. As such, the running time reported consists of the time used to actually solve the problem to termination excluding I/O time. This is consistent with the measurement techniques used by other popular approaches (Joachims, 1999, Platt, 1999). Putting all the data in memory is simpler to code and results in faster running times. However, it is not a fundamental requirement of our algorithm — block matrix multiplications, incremental evaluations of Q^{-1} using another application of the Sherman-Morrison-Woodbury identity, and indices on the dataset can be used to create an efficient disk based version of LSVM. We note that we tried to run SVM^{light} on this dataset, but after running for more than six hours it did not terminate.

We now demonstrate the effectiveness of LSVM in solving nonlinear classification problems through the use of kernel functions. One highly nonlinearly separable but simple example is the "tried and true" checkerboard dataset (Ho and Kleinberg, 1996), which has often been used to show the effectiveness of nonlinear kernel methods on a dataset for which a linear separation clearly fails. The checkerboard dataset contains 1000 points randomly sampled from a checkerboard. These points are used as a training set in LSVM to try to reproduce an accurate rendering of a checkerboard. For this problem, we used the following Gaussian kernel:

$$K(G, G') = \varepsilon^{-2\cdot 10^{-4}\|G_i - G_j\|^2}, i, j = 1, \ldots, m \tag{31}$$

Our results, shown in Figure 2, demonstrate that LSVM shows similar or better generalization capability on this dataset when compared to other methods (Kaufman, 1999, Mangasarian and Musicant, 2001, Lee and Mangasarian, 2000). Total time for the checkerboard problem using LSVM with a Gaussian kernel was 2.85 hours on the Locop2 Pentium II Xeon machine on the 1000-point training set in R^2 after 100,000 iterations. Test set accuracy was 97.0% on a 39,000-point test set. However, within 58 seconds and after 100 iterations, a quite reasonable checkerboard was obtained (`http://www.cs.wisc.edu/~olvi/lsvm/check100iter.eps`) with a 95.9% accuracy on the same test set.

The final set of experiments, shown in Table 4, demonstrates the effectiveness of nonlinear kernel functions with LSVM on UCI datasets (Murphy and Aha, 1992) specialized as follows for purposes of our tests:

- 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 eight 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

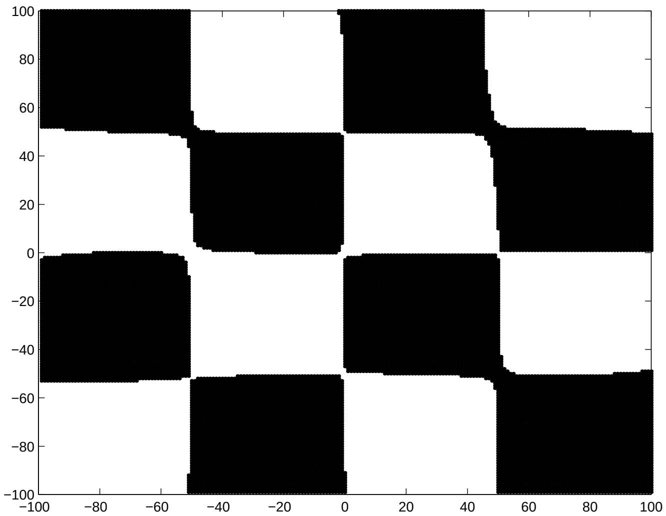

Figure 2: Gaussian kernel LSVM performance on checkerboard training dataset. ($\nu = 10^5$)

called "cap surface," which can take one one of four categories, namely "fibrous," "grooves," "scaly," or "smooth." We represented this as four binary attributes. A 1 is assigned to the attribute that corresponds to the actual category, and a 0 to the rest. Thus the categorical value "fibrous" would be represented by [1 0 0 0], while the value "smooth" would be represented by [0 0 0 1]. 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 chose which kernels to use and appropriate values of ν via tuning sets and ten-fold cross-validation techniques. The results show that nonlinear kernels produce test set accuracies that improve on those obtained with linear kernels.

6. Conclusion

A fast and extremely simple algorithm, LSVM, considerably easier to code than SVM^{light} (Joachims, 1999), SMO (Platt, 1999), SOR (Mangasarian and Musicant, 1999) and ASVM (Mangasarian and Musicant, 2000), capable of classifying datasets with millions of points has been proposed and implemented in a few lines of MATLAB code. For a linear kernel,

Dataset m x n	Algorithm	Kernel Type	Training Correctness	Testing Correctness	Time (secs)	Memory (Kilobytes)	# of SVs	# of SVs (w/ tolerance)
Liver Disorders	SVM^{light}	Linear	70.76%	68.41%	0.239	1448	225.7	225.7
		Quadratic	76.30%	72.79%	0.197	42413	218.6	218.6
	LSVM	Linear	70.15%	68.68%	*0.051	4	305.3	299.4
345 x 6		Quadratic	*75.17%	72.78%	*0.345	2418	304.4	298.5
Tic-Tac-Toe	SVM^{light}	Linear	66.67%	66.67%	0.036	1375	182.8	0.0
		Quadratic	98.30%	95.33%	17.575	42424	73.5	73.4
	LSVM	Linear	66.67%	66.67%	*0.001	23	270.0	0.0
300 x 9		Quadratic	*99.52%	95.00%	*0.377	1852	122.4	115.3
Letter Recognition	SVM^{light}	Linear	85.48%	85.00%	0.124	1234	240.3	240.3
		Cubic	89.31%	88.67%	0.374	1374	217.0	216.9
	LSVM	Linear	*81.69%	81.50%	*0.029	76	537.7	535.3
600 x 6		Cubic	88.56%	87.83%	*1.639	6976	472.2	400.7
Mushroom	SVM^{light}	Linear	77.24%	76.20%	0.536	1340	241.9	240.3
		Cubic	90.91%	88.40%	0.261	1340	174.5	163.6
	LSVM	Linear	77.96%	76.80%	*0.053	124	438.7	428.1
500 x 22		Cubic	90.91%	88.20%	*0.570	4724	366.7	250.5

Table 4: LSVM performance with linear, quadratic, and cubic kernels (polynomial kernels of degree 1, 2, and 3 respectively). An asterisk in the first three columns indicates that the LSVM results are significantly different from SVM^{light} at significance level $\alpha = 0.05$. The column "w/ tolerance" indicates the number of support vectors with dual variable $u > 0.001$.

LSVM is an iterative method which requires nothing more complex than the inversion of a single matrix of the order of the input space plus one, and thus has the ability to handle massive problems. For a positive semidefinite nonlinear kernel, a single matrix inversion is required in the space of dimension equal to the number of points classified. Hence, for such nonlinear classifiers LSVM can handle only intermediate size problems.

There is room for future work in reducing the number of support vectors in the solutions yielded by LSVM. One method would be to augment the quadratic term in y in the objective function of (7) by a 1-norm in y. This should decrease the number of support vectors as in work by Bradley and Mangasarian (1998) where the 1-norm was used to effectively suppress features. Additionally, the iterative algorithm itself could be modified so that dual variables with values smaller than a tolerance would be automatically set to zero.

Further future work includes extensions to parallel processing of the data and handling very large datasets directly from disk as well as extending nonlinear kernel classification to very large datasets.

Acknowledgements

Research described in this Data Mining Institute Report 00-06, June 2000, 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 Microsoft. We wish to thank our colleague Thorsten Joachims for supplying us with his latest code for SVM^{light} and for advice on how to run it efficiently.

© 2001 Olvi L. Mangasarian and David R. Musicant.
The LSVM Algorithm 1 and the LSVM Codes 2 and 3 are copyrighted and may not be used for any commercial purpose without written authorization from the authors.

References

D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, second edition, 1999.

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`.

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

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

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

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

ILOG, Inc. *ILOG CPLEX 6.5 Reference Manual.* ILOG CPLEX Division, Incline Village, Nevada, 1999.

T. Joachims. SVM^{light}, 1998. http://ais.gmd.de/˜thorsten/svm_light.

T. Joachims. Making large-scale support vector machine learning practical. In Schölkopf et al. (1999), pages 169–184.

L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In Schölkopf et al. (1999), pages 147–167.

Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining*, 2001. To appear. `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*, 2000. To appear. `ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-03.ps`.

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

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 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.

O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. *Advances in Neural Information Processing Systems (NIPS 2000)*, 2000. To appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

O. L. Mangasarian and D. R. Musicant. Data discrimination via nonlinear generalized support vector machines. In M. C. Ferris, O. L. Mangasarian, and J.-S. Pang, editors, *Complementarity: Applications, Algorithms and Extensions*, pages 233–251, Dordrecht, January 2001. Kluwer Academic Publishers. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/99-03.ps.

O. L. Mangasarian and J. Ren. New improved error bounds for the linear complementarity problem. *Mathematical Programming*, 66:241–255, 1994.

O. L. Mangasarian and M. V. Solodov. Nonlinear complementarity as unconstrained and constrained minimization. *Mathematical Programming, Series B*, 62:277–297, 1993.

P. M. Murphy and D. W. Aha. UCI repository of machine learning databases, 1992. http://www.ics.uci.edu/˜mlearn/MLRepository.html.

D. R. Musicant. NDC: Normally Distributed Clustered Datasets, 1998. http://www.cs.wisc.edu/˜musicant/data/ndc.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In Schölkopf et al. (1999), pages 185–208. http://www.research.microsoft.com/˜jplatt/smo.html.

B. Schölkopf, C. Burges, and A. Smola. *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1999.

The MathWorks, Inc. *MATLAB User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001.

The MathWorks, Inc. MATLAB Solution Number 1913: Can one session of MATLAB take advantage of multiple processors on the same PC or UNIX machine?, 2000. http://www.mathworks.com/support/solutions/data/1913.shtml.

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