Computational Optimization and Aplications, 28, 185–202 (2004)

A Feature Selection Newton Method for Support Vector Machine Classification

GLENN M. FUNG gfung@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

O. L. MANGASARIAN olvi@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Received January 9, 2003; Revised July 31, 2003

Abstract. A fast Newton method, that suppresses input space features, is proposed for a linear programming formulation of support vector machine classifiers. The proposed stand-alone method can handle classification problems in very high dimensional spaces, such as 28,032 dimensions, and generates a classifier that depends on very few input features, such as 7 out of the original 28,032. The method can also handle problems with a large number of data points and requires no specialized linear programming packages but merely a linear equation solver. For nonlinear kernel classifiers, the method utilizes a minimal number of kernel functions in the classifier that it generates.

Keywords *classification, feature selection, linear programming, Newton method*

1. Introduction

By minimizing an exterior penalty function of the dual of a linear programming formulation of a 1-norm support vector machine (SVM) [18, 3], for a finite value of the penalty parameter, an exact least 2-norm solution to the SVM classifier is obtained. Our approach is based on a 1-norm SVM formulation that is known [3, 28] to generate very sparse solutions. When a linear classifier is used, solution sparsity implies that the separating hyperplane depends on very few input features. This fact, makes this algorithm a very effective tool for feature selection in classification problems. On the other hand, when a nonlinear classifier is used, a sparse solution implies that few kernel functions determine the classifier. This makes the nonlinear classifier easier to store and faster to evaluate. The proposed Newton method requires only a linear equation solver and can be given in a few lines of a MATLAB [22] code. We note that a fast Newton method (NSVM) was also proposed recently in [8] that is based on a quadratic programming formulation of support vector machines. NSVM however, does not generate sparse solutions and hence does not suppress features at all. This contrasts sharply with the strong feature suppression property of the new algorithm proposed here. Feature suppression is very important in the development of new techniques in bioinformatics that utilize gene microarrays [5, 23] for prognostic classification, drug discovery and other tasks. Such problems typically consist of thousands of gene expression measurements for each patient in a given study, and where the number of patients is frequently of the order of one hundred or even less. An excellent overview of the state-of-the-art methods for feature selection via support vector machines appears in [2].

We outline now the contents of the paper. In Section 2 we formulate the linear and nonlinear kernel classification problems as a linear programming problem. We show how an exact solution to the linear programming SVM can be obtained by minimizing an exterior penalty function of its dual for a finite value of the penalty parameter. In Section 3 we give our Newton algorithm for solving the exterior penalty problem and establish its global convergence. In Section 4 we give some computational results including those for a Multiple Myeloma gene expression problem with 28,032 variables, solved in seconds, as well as six other publicly available datasets.

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 $(x_+)_i = \max\,\{0, x_i\}$, $i = 1, \ldots, n$, while x_* denotes the subgradient of x_+ which is the step function defined as $(x_*)_i = 1$ if $x_i > 0$, $(x_*)_i = 0$ if $x_i < 0$, and $(x_*)_i \in [0, 1]$ if $x_i = 0$, $i = 1, \ldots, n$. Thus, $(x_*)_i$ is any value in the interval $[0, 1]$, when $x_i = 0$, and we typically take $(x_*)_i = 0.5$ in this case. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$, the 2-norm of x will be denoted by $\|x\|$ and $x \perp y$ denotes orthogonality, that is $x'y = 0$. The 1-norm and ∞-norm will be denoted by $\|\cdot\|_1$ and $\|\cdot\|_\infty$ respectively. 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 and $\|A\|$ is the 2-norm of A: $\max\limits_{\|x\|=1} \|Ax\|$. A column vector of ones of arbitrary dimension will be denoted by e and the identity matrix of arbitrary order will be denoted by I. For $A \in R^{m\times n}$ and $B \in R^{n\times l}$, the kernel $K(A, B)$ [27, 4, 18] is an arbitrary function which maps $R^{m\times n} \times R^{n\times l}$ into $R^{m\times l}$. In particular, if x and y are column vectors in R^n then, $K(x', y)$ is a real number, $K(x', A')$ is a row vector in R^m and $K(A, A')$ is an $m \times m$ matrix. If f is a real valued function defined on the n-dimensional real space R^n, the gradient of f at x is denoted by $\nabla f(x)$ which is a column vector in R^n and the $n \times n$ matrix of second partial derivatives of f at x is denoted by $\nabla^2 f(x)$. For a piecewise quadratic function such as, $f(x) = \frac{1}{2}||(Ax - b)_+||^2 + \frac{1}{2}x'Px$, where $A \in R^{m\times n}$, $P \in R^{n\times n}$, $P = P'$, P positive semidefinite and $b \in R^m$, the ordinary Hessian does not exist because its gradient, the $n \times 1$ vector $\nabla f(x) = A'(Ax - b)_+ + Px$, is not differentiable. However, one can define its **generalized Hessian** [10, 6, 19] which is the $n \times n$ symmetric positive semidefinite matrix:

$$\partial^2 f(x) = A' diag(Ax - b)_* A + P, \tag{1}$$

where $diag(Ax - b)_*$ denotes an $m \times m$ diagonal matrix with diagonal elements $(A_i x - b_i)_*$, $i = 1, \ldots, m$. The generalized Hessian (1) has many of the properties of the regular Hessian [10, 6, 19] in relation to $f(x)$. If the smallest eigenvalue of $\partial^2 f(x)$ is greater than some positive constant for all $x \in R^n$, then $f(x)$ is a strongly convex piecewise quadratic function on R^n. Throughout this work, the notation $:=$ will denote definition.

2. Linear and Nonlinear Kernel Classification

We describe in this section the fundamental classification problems that lead to a linear programming problem. We consider the problem of classifying m points in the n-dimensional real space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the classes +1 or -1 as specified by a given $m \times m$ diagonal matrix D with ones or minus ones along its diagonal. For this problem, the standard support vector machine with a linear kernel AA' [27, 4] is given by the following quadratic program for some $\nu > 0$:

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

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

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

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

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

midway between the bounding planes (3). If the classes are linearly inseparable then the two planes bound the two classes with a "soft margin" determined by a nonnegative slack variable y, that is:

$$\begin{array}{rcl} x'w - \gamma + y_i & \geq & +1, \\ & & \text{for } x' = A_i \text{ and } D_{ii} = +1. \\ x'w - \gamma - y_i & \leq & -1, \\ & & \text{for } x' = A_i \text{ and } D_{ii} = -1. \end{array} \tag{5}$$

The 1-norm of the slack variable y is minimized with weight ν in (2). 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) in the n-dimensional space of $w \in R^n$ for a *fixed* γ, maximizes that distance, often called the "margin". Figure 1 depicts the points represented by A, the bounding planes (3) with margin $\frac{2}{\|w\|}$, and the separating plane (4) which separates $A+$, the points represented by rows of A with $D_{ii} = +1$, from $A-$, the points represented by rows of A with $D_{ii} = -1$.

In a linear programming SVM formulation [18, 3] of the standard SVM (2) the term $\frac{1}{2}w'w$ in (2) is replaced by $\|w\|_1$ which is twice the reciprocal of the ∞-norm distance [17] between the bounding planes (3). Empirical evidence [3] indicates that the 1-norm formulation has the advantage of generating very sparse solutions. This results in the normal w to the separating plane $x'w = \gamma$ having many zero

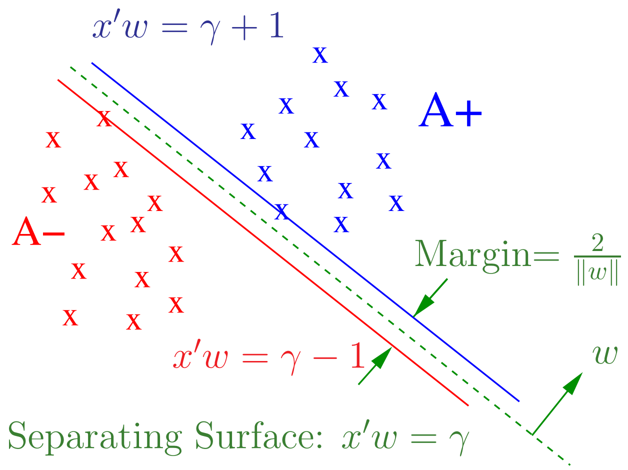

Figure 1. **The bounding planes (3) with margin $\frac{2}{\|w\|}$, and the plane (4) separating $A+$, the points represented by rows of A with $D_{ii} = +1$, from $A-$, the points represented by rows of A with $D_{ii} = -1$.**

components, which implies that many input space features do not play a role in determining the linear classifier. This makes this approach suitable for feature selection in classification problems. We note that in addition to the conventional interpretation of smaller ν as emphasizing a larger margin between the bounding planes (3), a smaller ν here also results in a sparse solution. This 1-norm formulation leads to the linear programming problem:

$$\begin{array}{rrcl} \displaystyle\min_{(p,q,\gamma,y)} & \nu e'y + e'(p+q) & & \\ \text{s.t.} & D(A(p-q) - e\gamma) + y & \geq & e \\ & p, q, y & \geq & 0, \end{array} \tag{6}$$

where the following substitution for w has been made:

$$w = p - q, \;\; p \geq 0, \; q \geq 0, \tag{7}$$

This is a different and a simpler linear program from previous linear programming SVM formulations [18, 3]. The dual of the linear program (6) is the following:

$$\begin{array}{rrcrcl} & \displaystyle\max_{u \in R^m} & & e'u & & \\ \text{s.t.} & -e & \leq & A'Du & \leq & e, \\ & & & -e'Du & = & 0, \\ & & & u & \leq & \nu e, \\ & & & u & \geq & 0. \end{array} \tag{8}$$

The asymptotic exterior penalty problem [7, 1] for this linear program is the following nonnegatively constrained minimization problem:

$$\begin{array}{rl} \min\limits_{u\geq 0} & -\epsilon e'u + \frac{1}{2}\|(A'Du-e)_+\|^2 + \\ \frac{1}{2}\|(-A'Du-e)_+\|^2 & +\frac{1}{2}\|e'Du\|^2 + \frac{1}{2}\|(u-\nu e)_+\|^2, \end{array} \tag{9}$$

where ϵ is a positive penalty parameter that needs to approach zero for standard penalty application for solving the dual linear program (8). *However*, in our approach we shall establish the fact that ϵ will remain finite and we still can obtain an exact solution to our linear programming SVM (6). To do that we first write the Karush-Kuhn-Tucker [15] necessary and sufficient optimality conditions for the penalty problem (9):

$$\begin{array}{rcl} 0 \leq u & \perp & (-\epsilon e + DA(A'Du-e)_+ \\ & & -DA(-A'Du-e)_+ \\ & & +Dee'Du + (u-\nu e)_+) \geq 0, \end{array} \tag{10}$$

where, as defined in the Introduction, $\perp$ denotes orthogonality. We will now show that these are also the necessary and sufficient conditions for finding an exact least 2-norm solution to the linear programming SVM (6) *without* ϵ approaching zero. To do that we first formulate the least 2-norm problem for (6) as follows:

$$\begin{array}{rl} \min\limits_{(p,q,\gamma,y)} & \nu e'y + e'(p+q) + \frac{\epsilon}{2}(\|p\|^2 + \|q\|^2 + \gamma^2 + \|y\|^2) \\ \text{s.t.} & D(A(p-q) - e\gamma) + y \geq e \\ & p, q, y \geq 0, \end{array} \tag{11}$$

with Karush-Kuhn-Tucker necessary and sufficient optimality conditions:

$$\begin{array}{rcl} 0 \leq \epsilon p & \perp & e + \epsilon p - A'Du \geq 0 \\ 0 \leq \epsilon q & \perp & e + \epsilon q + A'Du \geq 0 \\ & & \epsilon\gamma + e'Du = 0 \\ 0 \leq \epsilon y & \perp & \nu e + \epsilon y - u \geq 0 \\ 0 \leq u & \perp & DA(p-q) - De\gamma + y - e \geq 0. \end{array} \tag{12}$$

It follows, by [20, 14], that for any positive ϵ such that $\epsilon \in (0, \bar{\epsilon}]$ for some $\bar{\epsilon} > 0$, any (p, q, γ, y) satisfying the KKT conditions (12) for some $u \in R^m$, is the **exact** least 2-norm solution to the linear programming SVM (6). But, if we set in the KKT conditions (10) for the penalty problem (9):

$$\begin{array}{rcl} p & = & \frac{1}{\epsilon}(A'Du-e)_+, \\ q & = & \frac{1}{\epsilon}(-A'Du-e)_+, \\ \gamma & = & -\frac{1}{\epsilon}e'Du, \\ y & = & \frac{1}{\epsilon}(u-\nu e)_+, \end{array} \tag{13}$$

and make use of the simple equivalence:

$$a = b_+ \iff 0 \leq a \perp (a-b) \geq 0, \tag{14}$$

then (13) together with the KKT conditions (10) for the exterior penalty problem (9), become precisely the KKT necessary and sufficient conditions (12) for the least 2-norm linear programming SVM (11). We have thus proven the following result.

Proposition 1 (Equivalence of Least 2-norm LP SVM to Dual Exterior Penalty) *A solution u to the dual exterior penalty (DEP) problem (9) for $\epsilon \in (0, \bar{\epsilon}]$ for some $\bar{\epsilon} > 0$, provides an* **exact** *least 2-norm solution to the primal linear programming SVM (6) as follows:*

$$\begin{array}{lcl} w = p - q = & = & \frac{1}{\epsilon}((A'Du - e)_+ - (-A'Du - e)_+), \\ \gamma & = & -\frac{1}{\epsilon}e'Du, \\ y & = & \frac{1}{\epsilon}(u - \nu e)_+. \end{array} \tag{15}$$

We turn now to nonlinear kernel classifiers and use the notation of [18]. 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:

$$K(x', A')Dv = \gamma, \tag{16}$$

where v is the solution of the dual problem (8). For a linear kernel $K(A, A') = AA'$, we have that $w = A'Dv$ [18] and the primal linear programming SVM (2) becomes upon using $w = p - q = A'Dv$ and the 1-norm of v in the objective instead that of w:

$$\begin{array}{rrcl} \min\limits_{(v,\gamma,y)} & \nu e'y + \|v\|_1 & & \\ \text{s.t.} & D(AA'Dv - e\gamma) + y & \geq & e \\ & y & \geq & 0, \end{array} \tag{17}$$

Setting:

$$v = r - s, \;\; r \geq 0, \; s \geq 0, \tag{18}$$

the linear program (17) becomes:

$$\begin{array}{rrcl} \min\limits_{(r,s,\gamma,y)} & \nu e'y + e'(r + s) & & \\ \text{s.t.} & D(AA'D(r - s) - e\gamma) + y & \geq & e \\ & r, s, y & \geq & 0, \end{array} \tag{19}$$

which is the linear kernel SVM in terms of the dual variable $v = r - s$. If we replace the linear kernel AA' in (19) by the nonlinear kernel $K(A, A')$ we obtain the nonlinear kernel linear program:

$$\begin{array}{rrcl} \min\limits_{(r,s,\gamma,y)} & \nu e'y + e'(r + s) & & \\ \text{s.t.} & D(K(A, A')D(r - s) - e\gamma) + y & \geq & e \\ & r, s, y & \geq & 0. \end{array} \tag{20}$$

We immediately note that the linear program (20) is identical to the linear classifier SVM (6) if we let:

$$A \;\longrightarrow\; K(A, A')D, \tag{21}$$

in (6) and $n \longrightarrow m$. Hence the results outlined in Proposition 1 are applicable to a nonlinear kernel if we make the replacement (21) in (9) and (15) and let $p \longrightarrow r$, $q \longrightarrow s$, $w \longrightarrow v$ in (15) and use the nonlinear kernel classifier (16). As in the linear case, the 1-norm formulation (21) leads to a very sparse v. Every zero component v_i of v implies non-dependence of the nonlinear kernel classifier on the kernel function $K(x', A'_i)$. This is because:

$$\begin{array}{rcl} K(x', A')Dv & = & \sum_{i=1}^{m} D_{ii}v_iK(x', A'_i) \\ & = & \sum_{\{i|v_i\neq 0\}} D_{ii}v_iK(x', A'_i). \end{array} \tag{22}$$

We turn now to our algorithmic implementation of Proposition 1.

3. Newton Method for Linear Programming SVM (NLPSVM)

We shall solve the dual exterior penalty (9) for a finite value of the penalty parameter ϵ by incorporating the nonnegativity constraint $u \geq 0$ into the objective function of (9) as a penalty term which results in the following convex penalty function:

$$\begin{array}{rl} \min\limits_{u} f(u) = & -\epsilon e'u + \frac{1}{2}\|(A'Du - e)_+\|^2 \\ & +\frac{1}{2}\|(-A'Du - e)_+\|^2 + \frac{1}{2}\|e'Du\|^2 \\ & +\frac{1}{2}\|(u - \nu e)_+\|^2 + \frac{\alpha}{2}\|(-u)_+\|^2. \end{array} \tag{23}$$

We note that the use of the asymptotic penalty function above introduces an approximation to our solution which accounts for a slightly different computational result from that of CPLEX on the test problem results given in Tables 1 and 2. Another possible cause for the differences is that CPLEX was run with its default settings, whereas the parameters of NLPSVM were adjusted by a tuning set.

The gradient of the function $f(u)$ is given by:

$$\begin{array}{rcl} \nabla f(u) & = & -\epsilon e + DA(A'Du - e)_+ - DA(-A'Du - e)_+ \\ & & +Dee'Du + (u - \nu e)_+ - \alpha(-u)_+, \end{array} \tag{24}$$

and its generalized Hessian as defined by (1) in the Introduction:

$$\begin{array}{rcl} \partial^2 f(u) & = & DA(diag((A'Du - e)_* + (-A'Du - e)_*)A'D \\ & & +Dee'D + diag((u - \nu e)_* + \alpha(-u)_*) \\ & = & DA(diag(|A'Du| - e)_*)A'D \\ & & +Dee'D + diag((u - \nu e)_* + \alpha(-u)_*), \end{array} \tag{25}$$

where the last equality follows from the equality:

$$(a - 1)_* + (-a - 1)_* = (|a| - 1)_*. \tag{26}$$

We are ready now to state our Newton algorithm.

Algorithm 1 Newton Algorithm for (9) *Let $f(u)$, $\nabla f(u)$ and $\partial^2 f(u)$ be defined by (23)-(25). Set the parameter values ν, ϵ, δ, tolerance* tol, *α and* imax *(typically: $\epsilon = 4 \times 10^{-4}$ for linear SVMs and 10^{-1} for nonlinear SVMs, $tol = 10^{-3}$, $\alpha = 10^3$, $imax = 50$, while ν and δ are set by a tuning procedure described in Section 4.1.2). Start with any $u^0 \in R^m$. For $i = 0, 1, \ldots$:*

(I) $u^{i+1} = u^i - \lambda_i(\partial^2 f(u^i) + \delta I)^{-1}\nabla f(u^i) = u^i + \lambda_i d^i$,
where the Armijo stepsize $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is such that:

$$f(u^i) - f(u^i + \lambda_i d^i) \geq -\frac{\lambda_i}{4}\nabla f(u^i)' d^i, \tag{27}$$

and d^i is the modified Newton direction:

$$d^i = -(\partial^2 f(u^i) + \delta I)^{-1}\nabla f(u^i). \tag{28}$$

(II) *Stop if $\|u^i - u^{i+1}\| \leq tol$ or $i = imax$. Else, set $i = i + 1$, $\alpha = 2\alpha$ and go to (I).*

(III) *Define the least 2-norm solution of the linear programming SVM (6) by (15) with $u = u^i$.*

We state a convergence result for this algorithm now.

THEOREM 1 *Let $tol = 0$, $imax = \infty$ and let $\epsilon > 0$ be sufficiently small. Each accumulation point $\bar{u}$ of the sequence $\{u^i\}$ generated by Algorithm 1 solves the exterior penalty problem (9). The corresponding $(\bar{w}, \bar{\gamma}, \bar{y})$ obtained by setting u to $\bar{u}$ in (15) is the exact least 2-norm solution to the primal linear program SVM (6).*

Proof That each accumulation point $\bar{u}$ of the sequence $\{u^i\}$ solves the minimization problem (9) follows from exterior penalty results [7, 1] and standard unconstrained descent methods such as [16, Theorem 2.1, Examples 2.1(i), 2.2(iv)] and the facts that the direction choice d^i of (19) satisfies, for some $c > 0$:

$$\begin{array}{rcl} -\nabla f(u^i)' d^i & = & \nabla f(u^i)'(\delta I + \partial^2 f(u^i))^{-1}\nabla f(u^i) \\ & \geq & c\|\nabla f(u^i)\|^2, \end{array} \tag{29}$$

and that we are using an Armijo stepsize (27). The last statement of the theorem follows from Proposition 1.□

Remark 2 Choice of ϵ *Determining the size of $\bar{\epsilon}$, such that the solution u of the quadratic program (11) for $\epsilon \in (0, \bar{\epsilon}]$, is the least 2-norm solution of the problem (6), is not an easy problem theoretically. However, computationally this does not seem to be critical and is effectively addressed as follows. By [13, Corollary 3.2], if for two successive values of ϵ: $\epsilon^1 > \epsilon^2$, the corresponding solutions of the ϵ-perturbed quadratic programs (11): u^1 and u^2 are equal, then under certain assumptions, $u = u^1 = u^2$ is the least 2-norm solution of the dual linear program (6). This result can be implemented computationally by using an ϵ, which when decreased by some factor yields the same solution to (6).*

We turn now to our computational results.

4. Numerical Experience

In order to show that our algorithm can achieve very significant feature suppression, our numerical tests and comparisons were carried out on a dataset with the high dimensional input space and a moderate number of data points. On the other hand, in order to show that our proposed algorithm has a computational speed comparable to that of other fast methods, we also performed experiments on more conventional datasets where the dimensionality of the input space is considerably smaller than the number of data points. 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 [22].

Because of the simplicity of our algorithm, we give below a simple MATLAB implementation of the algorithm without the Armijo stepsize, which does not seem to be needed in most applications. Although this is merely an empirical observation in the present case, it considerably simplifies our MATLAB Code 1. However, it has also been shown [19, Theorem 3.6] that under a well conditioned assumption, not generally satisfied here, the proposed Newton method indeed terminates in a finite number of steps without an Armijo stepsize. Note that this version of the algorithm is intended for cases where the number of data point m is smaller that the number of features n, i.e. when $m \ll n$ since the speed of the algorithm depends on m in a cubic fashion.

Code 1 NLPSVM Code

```
function [w,gamma]=nlpsvm(A,d,nu,delta)
%NLPSV: linear and nonlinear classification
%       without Armijo
%INPUT: A, D, nu, delta. OUTPUT=w, gamma.
%[w,gamma]=nlpsvm(A,d,nu,delta)
epsi=10^(-1);alpha=10^3;tol=10^(-3);imax=50;
[m,n]=size(A);en=ones(n,1);em=ones(m,1);
u=ones(m,1);%initial point
iter=0;g=1;
epsi=epsi*em;nu=nu*em;
DA=spdiags(d,0,m,m)*A;
while (norm(g)>tol) & (iter<imax)
  iter=iter+1;
  du=d.*u;Adu=A'*du;
  pp=max(Adu-en,0);np=max(-Adu-en,0);
  dd=sum(du)*d;unu=max(u-nu,0);uu=max(-u,0);
  g=-epsi+(d.*(A*pp))-(d.*(A*np))+dd+unu-alpha*uu;
  E=spdiags(sqrt(sign(np)+sign(pp)),0,n,n);
  H=[DA*E d];
```

```
  F=delta+sign(unu)+alpha*sign(uu);
  F=spdiags(F,0,m,m);
  di=-((H*H'+F)\g);
  u=u+di;
end
du=d.*u;Adu=A'*du;
pp=max(Adu-en,0);np=max(-Adu-en,0);
w=1/epsi(1)*(pp-np);
gamma=-(1/epsi(1))*sum(du);
return
```

We further note that the MATLAB code above not only works for a linear classifier, but also for a nonlinear classifier as well [18, Equations (1), (10)]. In the nonlinear case, the matrix $K(A, A')D$ is used as input instead of A, and the pair (v, γ), is returned instead of (w, γ). The nonlinear separating surface is then given by (11) as:

$$K(x, A')Dv - \gamma = 0. \tag{30}$$

Our first numerical testing and comparisons were carried out on the high dimensional Multiple Myeloma dataset available at: http://lambertlab.uams.edu/publicdata.htm, and processed by by David Page and his colleagues [26]. The structure of this dataset with very large n and $(m \ll n)$ results from the DNA microarray dataset used. Hence, feature selection is very desirable in such high dimensional problems. Other tests and comparisons were also carried out on six moderately dimensioned, publicly available datasets [24, 25] and are described Section 4.2.

4.1. Multiple Myeloma dataset

Multiple Myeloma is cancer of the plasma cell. The plasma cell normally produces antibodies that destroy foreign bodies such as bacteria. As a product of the Myeloma disease the plasma cells get out of control and produce a tumor. These tumors can grow in several sites, usually in the soft middle part of bone, the bone marrow. When these tumors appear in multiples sites they are called Multiple Myeloma. A detailed description of the process used to obtain the data can be found in [26].

4.1.1. **Description of the dataset** The data consists of 105 data points, 74 of the points representing newly-diagnosed multiple Myeloma patients while 31 points represent 31 healthy donors. Each data point represents measurements taken from 7008 genes using plasma cells samples from the patients. For each one of the 7008 genes there are two measurements. One measurement is called Absolute Call (AC) and takes on one of three nominal values: A (Absent), M (Marginal) or P (Present). The other measurement, the average difference (AD), is a floating point number that can be either positive or negative. Since each one of the 7008 AC features takes on nominal values from the set $\{A, M, P\}$, a real valued representation is needed to utilize our classifier which requires an input of real numbers. Thus, each nominal value is mapped into a three dimensional binary vector depending on the value that is being represented. This simple and widely used "1 of N" mapping scheme for converting nominal attributes into real-valued attributes is illustrated in Figure 2. Once this simple conversion is applied to the dataset, the AC feature space is transformed from a 7008-dimensional space with nominal values A, M, P into a $7008 \times 3 = 21024$ real-valued dimensional space. Adding the numerical AD feature for each of the 7008 genes results in each data point being transformed to a point in R^{28032}, with 21024 coming from the AC transformation mentioned above and 7008 from the AD values. This makes this dataset very interesting for our method, since a main objective of this paper is to show that our proposed algorithm does a remarkable job of suppressing features especially for datasets in a very high dimensional input space.

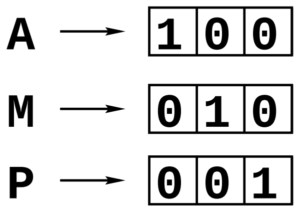

Figure 2. **Real-valued representation of the AC features set** $\{A, M, P\}$ **.**

4.1.2. **Numerical comparisons** Performance of our Newton Linear Programming SVM (**NLPSVM**) algorithm on the Myeloma dataset, in terms of feature selection and generalization ability, is first compared with two publicly available SVM solvers: LSVM [21] and NSVM [8]. Reported times for LSVM here differ from the ones reported in [21] because the calculation time for the matrix H of (9) is considered as input time in [21], whereas here it is counted as part of the computational time. The other algorithm included in our comparisons, consists of solving the linear programing formulation (20) employing the widely used commercial solver CPLEX 6.5 [11]. We call this approach CPLEX SVM. Termination criteria for all methods, with the exception of CPLEX SVM, was set to $tol = 0.001$, which is the default for LSVM. For CPLEX SVM the termination criterion used was the default supplied in the commercial package. We outline some of the results of our comparative testing.

- All three methods tested: NSVM, NLPSVM and CPLEX SVM obtained 100% *leave-one-out correctness (looc).* The following tuning procedure was employed for each of the 105 folds:

  - A random tuning set of the the size of 10% of the training data was chosen and separated from the training set.
  - Several SVMs were trained on the remaining 90% of the training data using values of ν equal to 2^i with $i = -12, \ldots, 0, \ldots, 12$. Values of the parameter δ tried were 10^j with $j = -3, \ldots, 0, \ldots, 3$. This made the search space for the pair (ν, δ) a grid of dimension 25×7.
  - Values of ν and δ that gave the best SVM correctness on the tuning set were chosen.
  - A final SVM was trained using the chosen values of ν, δ and all the training data. The resulting SVM was tested on the testing data.

- The remaining parameters were set to the following values: $\epsilon = 4 \times 10^{-4}, \alpha = 10^3, tol = 10^{-3}, imax = 50$

- Our NLPSVM algorithm outperformed all others in the feature selection task, and it obtained 100% looc using only 7 features out of 28032 original features. The closest contender was CPLEX SVM which required more than twice as many features.

- The average cpu time required by our algorithm for the leave-one-out correctness (*looc*) computations was 75.16 seconds per fold and total time for 105 folds was 7891.80 seconds. This outperformed CPLEX SVM both in cpu time and number of features used. CPLEX SVM had a cpu time average of 108.28 seconds per fold, a total time of 11369.40 seconds and used an average 16 features per fold. However, NLPSVM was considerably slower than the NSVM which had a cpu time of 4.20 seconds average per fold and total *looc* time of 441.00 seconds. Also, the NSVM classifier required an average 6554 features per fold, more than any classifier obtained by all other methods.

- LSVM reported an out of memory error. However, as pointed out by one of the referees, this can be avoided if LSVM is recoded without the Sherman-Morrison-Woodbury which is part of its code now. Hence the comparisons reported here are with LSVM as presently coded.

- Only NLPSVM and CPLEX SVM achieved a rather dramatic feature reduction from 28032 to 7 and 16 respectively as average number of features used per fold of the 105 folds of testing. Also to show the variance in the specific features used in each fold, we kept track of the overall features used in all the folds by NLPSVM and CPLEX SVM. These are respectively 7 and 16 again, with an overlap of 4 features between the 7 and 16 features.

These results are summarized in Table 1 below.

4.2. Tests on six other datasets

In this section we exhibit the effectiveness of NLPSVM in performing feature selection while maintaining accuracy and cpu time comparable to those of other methods

Table 1. **Myeloma Dataset Results for NSVM [8], CPLEX SVM [11], LSVM [21] & NLPSVM : *leave-one-out correctness (looc)*, total running times, average number of features per fold used by a LINEAR classifier, and the overall number of different features used in the 105 folds of testing. Best results are in bold. "oom", which stands for "out of memory", can be avoided for LSVM by recoding it without the Sherman-Morrison-Woodbury formula. However, as indicated by the results in Table 2, LSVM will not reduce features as does NLPSVM.**

Data Set $m \times n$ ($points \times dimensions$)	NSVM[8] looc Time (Sec.) Average Features	CPLEX SVM[11] looc Time (Sec.) Average Features Overall Features	LSVM[21] looc Time (Sec.) Average Features	NLPSVM looc Time (Sec.) Average Features Overall Features
Myeloma 105×28032	**100.0%** **441.00** 6554	**100.0%** 11369.40 16 16	oom oom oom	**100%** 7891.80 **7** **7**

that do not perform feature selection. We tested our algorithm on six publicly available datasets. Five from the UCI Machine Learning Repository [24]: Ionosphere, Cleveland Heart, Pima Indians, BUPA Liver and Housing. The sixth dataset is the Galaxy Dim dataset available at [25]. The dimensionality and size of each dataset is given in Table 2.

4.2.1. **Numerical comparisons using a linear classifier** In this set of experiments we used a linear classifier to compare our method NLPSVM with LSVM, NSVM and CPLEX SVM on the six datasets mentioned above. Because $m \gg n$ for these datasets, it was preferable to use the Sherman-Morrison-Woodbury identity [9] to calculate the direction d_i in the Newton iteration (28) and solve an $(n+1) \times (n+1)$ linear system of equations instead of an $m \times m$ linear system of equations. For this purpose define:

$$\begin{array}{l} E^2 := diag(|A'Du| - e)_*, \\ H := D[AE \quad e]. \\ \text{and } F := diag((u - \nu e)_* + \alpha(-u)_*) + \delta I. \end{array} \tag{31}$$

Then, it follows from (25) that:

$$\partial^2 f(u) + \delta I = HH' + F,$$

which is the matrix whose inverse is needed in the Newton iteration (28).

Applying the Sherman-Morrison-Woodbury identity we have:

$$(HH' + F)^{-1} = F^{-1} - F^{-1}H(I + H'F^{-1}H)^{-1}H'F^{-1}$$

Note that the inverse F^{-1} of F is trivial to calculate since F is a diagonal matrix. This simple but effective algebraic manipulation makes our algorithm very fast even when $m \gg n$ but n is relatively small.

The values for the parameters ν and δ were again calculated using the same tuning procedure given in Section 4.1.2. The values of the remaining parameters were the same as those used in Section 4.1.2. As shown in Table 2, the correctness of the four methods was very similar, the execution time including ten-fold cross validation for NSVM was less for all the datasets tested. However, all solutions obtained by NSVM depended on **all** the original input features. In contrast, NLPSVM performed comparably to LSVM, was always faster than CPLEX SVM but used the least number of features on all the datasets compared to all other methods tested.

Table 2. **NSVM [8], CPLEX SVM [11], LSVM [21] & NLPSVM: Training correctness, ten-fold testing correctness, ten-fold training times and number of features needed using a LINEAR classifier. All parameters ν, δ chosen by tuning. For each algorithm a reduced kernel version was also tested. Best results are in bold. Training and testing correctness and number of features are all averages over ten folds, while time is the total time over ten folds.**

Data Set $m \times n$ (*points* $\times$ *dimensions*)	NSVM Train Test Time (Sec.) Features	CPLEX SVM Train Test Time (Sec.) Features	LSVM Train Test Time (Sec.) Features	NLPSVM Train Test Time (Sec.) Features
Ionosphere	**92.9%**	90.9 %	**92.9%**	90.7%
351×34	**88.9%**	88.3 %	**88.9%**	88.0%
	0.91	3.2	1.49	2.4
	34	17.7	34	**11.2**
BUPA Liver	70.3%	**71.2%**	70.3%	70.6%
345×6	**70.2%**	69.9%	**70.2%**	68.8%
	0.24	5.17	0.92	1.13
	6	6	6	**4.8**
Pima Indians	**77.7%**	76.8%	**77.7%**	76.8%
768×8	**77.2%**	77.0%	**77.2%**	77.1%
	0.55	3.94	2.30	1.07
	8	5	8	**4.9**
Cleveland Heart	**87.2%**	85.9%	**87.2%**	86.5%
297×13	**86.6%**	85.5%	**86.6%**	85.9%
	0.14	1.08	0.31	0.55
	13	7.5	13	**7.1**
Housing	**87.7%**	**87.7%**	**87.7%**	87.0%
506×13	**86.8%**	85.0%	**86.8%**	85.2%
	0.69	2.54	1.53	1.91
	13	10.9	13	**6.5**
Galaxy Dim	94.0%	**94.7%**	94.0%	94.4%
4192×14	94.2%	**94.7%**	94.2%	94.6%
	6.67	29.82	71.56	8.90
	14	5	14	**3.4**

4.2.2. **Numerical comparisons using a nonlinear classifier** In order to show that our algorithm can also be used to find nonlinear classifiers, we chose three datasets from the UCI Machine Learning Repository for which it is known that a nonlinear classifier performs better that a linear classifier. We used NSVM, LSVM , CPLEX SVM and our proposed algorithm NLPSVM in order to find a nonlinear classifier based on the Gaussian kernel:

$$\begin{array}{l}(K(A,B))_{ij} = \varepsilon^{-\mu\|A_i{}'-B_{\cdot j}\|^2},\\ i=1,\ldots,m,\ j=1,\ldots,k.\end{array} \tag{32}$$

where $A \in R^{m\times n}$, $B \in R^{n\times k}$ and μ is a positive constant. The value of μ in the Gaussian kernel and the value of ν in all the algorithms were chosen by tuning on the values 2^i with i an integer ranging from -12 to 12 following the same procedure described in Section 4.1.2. The value of δ in NLPSVM was obtained also by tuning on the values 10^j with $j=-3,\ldots,0,\ldots,3$. The value of the parameter ϵ in this case was set to 10^{-1}. The values of the remaining parameters were the same as in Section 4.1.2. Because the nonlinear kernel matrix is square and since NLPSVM, NSVM and LSVM perform better on rectangular matrices, we also used a rectangular kernel formulation as described in the Reduced SVM (RSVM) [12]. This resulted in as good or better correctness and much faster running times. The size of the random sample used to calculate the rectangular kernel was 10% of the size of the original dataset in all cases. We refer to these variations of NSVM,LSVM, CPLEX SVM and NLPSVM as Reduced NSVM, Reduced LSVM, Reduced CPLEX SVM and Reduced NLPSVM respectively. The results are summarized in Table 3 for these nonlinear classifiers.

Again, as in the linear case the correctness of the four methods was similar on all the datasets, the execution time including ten-fold cross validation for NSVM was less for all the datasets tested, but with non-sparse solutions. NLPSVM performance was fast when a reduced rectangular kernel was used and it obtained very sparse solutions that resulted in nonlinear kernel classifiers that are easier to store and to evaluate.

5. Conclusion

We have presented a fast and finitely terminating Newton method for solving a fundamental classification problem of data mining with a pronounced feature selection property for linear classifiers. When nonlinear kernels are used, the algorithm performs feature selection in a high dimensional space of the dual variable, which results in a nonlinear kernel classifier that depends on a small number of kernel functions. This makes the method a very good choice for classification when feature selection or a fast nonlinear kernel classifier is required, as in the case of online decision making such as fraud or intrusion detection.

The NLPSVM algorithm requires only a linear equation solver, which makes it simple, fast and easily accessible. In addition, NLPSVM can be applied very effectively to classification problems in very large dimensional input spaces, which is often the case in the analysis of gene expression microarray data. NLPSVM can

Table 3. **NSVM [8], LSVM [21], NLPSVM, CPLEX SVM [11] and Reduced [12] NSVM, LSVM, NLPSVM, CPLEX SVM: Training correctness, ten-fold testing correctness, ten-fold training times and cardinality of v ($Card(v)$) using a NON-LINEAR classifier. Best results are in bold. Training and testing correctness and and cardinality of v are all averages over ten folds, while time is the total time over ten folds.**

Algorithm	Data Set $m \times n$ ($points \times dimensions$)	Ionosphere 351×34	BUPA Liver 345×6	Cleveland Heart 297×13
NSVM	Train	96.1	75.7	87.6
	Test	95.0	73.1	86.8
	Time (Sec.)	23.27	25.54	17.51
	$Card(v)$	351	345	297
Reduced	Train	96.1	**76.4**	86.8
NSVM	Test	94.5	73.9	**87.1**
	Time (Sec.)	**0.88**	**0.67**	**0.53**
	$Card(v)$	35	35	30
LSVM	Train	96.1	75.7	87.6
	Test	95.0	73.1	86.8
	Time (Sec.)	23.76	27.01	12.98
	$Card(v)$	351	345	297
Reduced	Train	96.1	75.1	87.1
LSVM	Test	94.5	73.1	86.2
	Time (Sec.)	2.09	1.81	1.09
	$Card(v)$	35	35	30
NLPSVM	Train	94.4	75.4	86.9
	Test	93.5	**73.9**	86.2
	Time (Sec.)	195.31	187.91	70.47
	$Card(v)$	22.3	32.7	50.1
Reduced	Train	94.4	74.5	85.9
NLPSVM	Test	95.1	**73.9**	86.5
	Time (Sec.)	2.65	6.82	5.17
	$Card(v)$	**14.6**	**16.4**	**12.3**
CPLEX SVM	Train	**99.2**	**76.4**	**87.8**
	Test	**96.1**	73.6	86.2
	Time (Sec.)	34.8	34.48	18.37
	$Card(v)$	36.1	26.2	14.1
Reduced	Train	98.7	**76.4**	87.0
CPLEX SVM	Test	95.5	**73.9**	85.6
	Time (Sec.)	3.08	4.42	2.47
	$Card(v)$	26.9	18.7	12.6

also be used effectively for classifying large datasets in smaller dimensional input space. As such, NLPSVM is a versatile stand-alone algorithm for classification which hopefully is a valuable addition to the tools of data mining and machine learning.

Acknowledgments

The research described in this Data Mining Institute Report 02-03, September 2002, was supported by National Science Foundation Grants CCR-0138308 and CDA-9623632, and by Air Force Office of Scientific Research Grant F49620-00-1-0085. We thank David Page for making available to us the Multiple Myeloma dataset as well as a preprint of his joint paper [26].

References

1. D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, second edition, 1999.
2. J. Bi, K. P. Bennett, M. Embrechts, C. M. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, March 2003.
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. V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods.* John Wiley & Sons, New York, 1998.
5. M.J. van de Vijver *et al.* A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347:1999–2009, 2002.
6. F. Facchinei. Minimization of SC^1 functions and the Maratos effect. *Operations Research Letters*, 17:131–137, 1995.
7. A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley & Sons, New York, NY, 1968.
8. G. Fung and O. L. Mangasarian. Finite Newton method for Lagrangian support vector machine classification. Technical Report 02-01, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, February 2002. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-01.ps. Neurocomputing, to appear.
9. G. H. Golub and C. F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.
10. J.-B. Hiriart-Urruty, J. J. Strodiot, and V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with C^{L1} data. *Applied Mathematics and Optimization*, 11:43–56, 1984.
11. ILOG CPLEX Division, 889 Alder Avenue, Incline Village, Nevada. *CPLEX Optimizer.* `http://www.cplex.com/`.
12. Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. Technical Report 00-07, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, July 2000. Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM Proceedings. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.
13. S. Lucidi. A new result in the theory and computation of the least-norm solution of a linear program. *Journal of Optimization Theory and Applications*, 55:103–117, 1987.
14. O. L. Mangasarian. Normal solutions of linear programs. *Mathematical Programming Study*, 22:206–216, 1984.

15. O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.
16. O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995. ftp://ftp.cs.wisc.edu/tech-reports/reports/1993/tr1145.ps.
17. O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07r.ps.
18. 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.
19. O. L. Mangasarian. A finite Newton method for classification problems. Technical Report 01-11, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, December 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-11.ps. *Optimization Methods and Software* 17, 2002, 913-929.
20. O. L. Mangasarian and R. R. Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, November 1979.
21. O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.
22. MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com.
23. M. Molla, M. Waddell, D. Page, and J. Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine, Special Issue on Bioinformatics*, 2003. To appear. ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/molla.aimag03.pdf.
24. P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.
25. S. Odewahn, E. Stockwell, R. Pennington, R. Humphreys, and W. Zumach. Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103(1):318–331, 1992.
26. D. Page, F. Zhan, J. Cussens, M. Waddell, J. Hardin, B. Barlogie, and J. Shaughnessy, Jr. Comparative data mining for microarrays: A case study based on multiple myeloma. Technical Report 1453, Computer Sciences Department, University of Wisconsin, November 2002.
27. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.
28. J. Zhy, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. Working paper, Department of Statistics, Stanford University, Stanford CA 94305, 2003. http://www-stat.stanford.edu/∼hastie/Papers/.