Optimization Methods and Software
Vol. 00, No. 00, Month 200x, 1–12

Chunking for Massive Nonlinear Kernel Classification

O. L. Mangasarian and M. E. Thompson
(Received 00 Month 200x; In final form 00 Month 200x)

A chunking procedure [2] utilized in [18] for linear classifiers is proposed here for nonlinear kernel classification of massive datasets. A highly accurate algorithm based on nonlinear support vector machines that utilizes a linear programming formulation [15] is developed here as a completely unconstrained minimization problem [17]. This approach together with chunking leads to a simple and accurate method for generating nonlinear classifiers for a 250000-point dataset that typically exceeds machine capacity when standard linear programming methods such as CPLEX [12] are used. Because a 1-norm support vector machine underlies the proposed method, the approach together with a reduced support vector machine formulation [13] minimizes the number of kernel functions utilized to generate a simplified nonlinear classifier.

Keywords: *classification, nonlinear kernel, massive datasets, linear programming, dual penalty*

1 Introduction

One of the principal advantages of 1-norm support vector machines (SVMs) is that, unlike 2-norm SVMs, they are very effective in reducing the number of kernel functions used [8] especially when a reduced support vector machine (RSVM) is used [13]. We utilize here an exact completely unconstrained differentiable minimization formulation of 1-norm SVMs [17] coupled with a chunking procedure [2], which allows us to handle massive datasets.

We start with a linear programming formulation of the nonlinear kernel 1-norm SVM classifier [15]. We then show how such a linear program can be solved exactly by minimizing a classical exterior penalty function of its dual for a sufficiently small but *finite* value of the penalty parameter. We then apply a chunking procedure [2] to decompose the 1-norm linear program for massive datasets and solve the linear program for each chunk exactly by utilizing a

Data Mining Institute Technical Report 06-07, December 2006. Research supported by NSF Grants CR-0138308 and IIS-0511905.
olvi@cs.wisc.edu. Computer Sciences Department, University of Wisconsin, Madison, WI 53706, and Department of Mathematics, University of California at San Diego, La Jolla, CA 92093.
thompson@cs.wisc.edu. Computer Sciences Department, University of Wisconsin, Madison, WI 53706.

Newton method applied to the dual exterior penalty function of the linear program chunk.

The contents of the paper are as follows.

In Section 2 we describe our unconstrained minimization formulation for a 1-norm SVM and give a simple Newton method for its solution that merely solves a sequence of linear equations. In Section 3 we combine our unconstrained SVM formulation with a chunking method that enables it to handle massive datasets. In Section 4 we present our numerical results for massive datasets which show among other things an error reduction of more than 37% by our nonlinear classifier over a linear classifier and an error reduction of more than 26% over a nonlinear classifier generated by 1% of the data. Section 5 concludes the paper.

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. For a vector $x \in R^n$, x_* denotes the vector in R^n with components $(x_*)_i = 1$ if $x_i > 0$ and 0 otherwise (i.e. x_* is the result of applying the step function component-wise to x). For $x \in R^n$, $\|x\|_1$ and $\|x\|$ will denote the $1-$ and $2-$ norms of x. For simplicity we drop the 2 from $\|x\|_2$. 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, $A_{\cdot j}$ will denote the j-th column of A, and A_{ij} will denote the ij-th element of A. A vector of ones or zeroes in a real space of arbitrary dimension will be denoted by e or 0, respectively. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$, a *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, $K(x', B)$ is a row vector in R^k and $K(A, B)$ is an $m \times k$ matrix. We shall make no assumptions whatsoever on our kernels other than symmetry, that is $K(x', y)' = K(y', x)$, and in particular we shall not assume or make use of Mercer's positive definiteness condition [5,21,24]. The base of the natural logarithm will be denoted by ε. A frequently used kernel in nonlinear classification is the Gaussian kernel [3, 15, 24] whose ij-th element, $i = 1, \ldots, m,\ j = 1, \ldots, k$, is given by: $(K(A, B))_{ij} = \varepsilon^{-\mu\|A_{i\cdot}' - B_{\cdot j}\|^2}$, where $A \in R^{m\times n}$, $B \in R^{n\times k}$ and μ is a positive constant. 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 but is Lipschitz continuous with a Lipschitz constant of $\|A'\|\,\|A\| + \|P\|$. However, one can define its **generalized Hessian** [7, 10, 16] which is the $n \times n$ symmetric positive semidefinite matrix: $\partial^2 f(x) = A'diag(Ax - b)_* A + P$, 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 has many of the properties

of the regular Hessian [7, 10, 16] 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. The abbreviation "s.t." stands for "subject to".

2 Nonlinear 1-Norm SVMs as Unconstrained Minimization Problems

We consider first the 1-norm nonlinear SVM binary classification problem [8, 15]:

$$\begin{array}{rl} \min\limits_{(v,\gamma,y)} & \nu\|y\|_1 + \|v\|_1 \\ \text{s.t.} & D(K(A,B')v - e\gamma) + y \geq e, \\ & y \geq 0, \end{array} \tag{1}$$

where the $m \times n$ matrix A represents m points in R^n to be separated to the best extent possible by the nonlinear separating surface:

$$K(x', B')v = \gamma, \tag{2}$$

according to the class of each row of A as given by the $m \times m$ diagonal matrix D with elements $D_{ii} = \pm 1$. Here, $v \in R^k$ and $\gamma \in R$ are parameters to be determined by a linear program, $B \in R^{k \times n}$ and $K(x', B') : R^{1 \times n} \times R^{n \times k} \longrightarrow R^{1 \times k}$ is an arbitrary kernel function. In general the matrix B is set equal to A [15]. However, in reduced support vector machines [11, 13] $B = \bar{A}$, where $\bar{A}$ is a submatrix of A whose rows are a small subset of the rows of A. In fact B can be an arbitrary matrix in $R^{k \times n}$. The objective term $\|y\|_1$ minimizes the empirical classification error weighted with the positive parameter ν, while the term $\|v\|_1$ minimizes the number of kernel functions utilized and leads to classifier simplicity. We convert (1) to an explicit linear program as in [8] by setting:

$$v = r - s, \;\; r \geq 0, \; s \geq 0, \tag{3}$$

which results in the linear program:

$$\begin{array}{rl} \min\limits_{(r,s,\gamma,y)} & \nu e'y + e'(r+s) \\ \text{s.t.} & D(K(A,B')(r-s) - e\gamma) + y \geq e, \\ & r, s, y \geq 0. \end{array} \tag{4}$$

We note immediately that this linear program is solvable because it is feasible and its objective function is bounded below by zero. We shall utilize the following proposition [17, Proposition 1] to reduce our linear program to a completely unconstrained minimization problem. Briefly stated, Proposition 2.1 below states that by using a classical exterior penalty function on the dual of the linear program (4) with a sufficiently small penalty parameter $\epsilon \in (0, \bar{\epsilon}]$ we can get an *exact* solution to the primal problem (4).

Proposition 2.1 **Exact 1-Norm SVM Solution via Unconstrained Minimization** The unconstrained dual exterior penalty problem for the 1-norm SVM (4):

$$\begin{array}{cl} \min\limits_{u \in R^m} & -\epsilon e'u + \frac{1}{2}(\|(K(B,A')Du - e)_+\|^2 + \|(-K(B,A')Du - e)_+\|^2 \\ & +(-e'Du)^2 + \|(u - \nu e)_+\|^2 + \|(-u)_+\|^2), \end{array} \tag{5}$$

is solvable for all $\epsilon > 0$. For any $\epsilon \in (0, \bar{\epsilon}]$ and for some $\bar{\epsilon} > 0$, any solution u of (5) generates an exact solution of the 1-norm SVM classification problem (1) as follows:

$$\begin{array}{lll} v = r - s = & = \frac{1}{\epsilon}((K(B,A')Du - e)_+ - (-K(B,A')Du - e)_+), \\ \gamma & = -\frac{1}{\epsilon}e'Du, \\ y & = \frac{1}{\epsilon}(u - \nu e)_+. \end{array} \tag{6}$$

In addition this (v, γ, y) minimizes:

$$\|v\|^2 + \gamma^2 + \|y\|^2 + \|D(K(A,B')v - e\gamma) + y - e\|^2, \tag{7}$$

over the solution set of the 1-norm SVM classification problem (1).

We will now give a generalized Newton method for solving (5). To do that we let $f(u)$ denote the exterior penalty function (5). Then the gradient and generalized Hessian as defined in the Introduction are given as follows.

$$\begin{array}{rl} \nabla f(u) = & -\epsilon e + DK(A,B')(K(B,A')Du - e)_+ \\ & -DK(A,B')(-K(B,A')Du - e)_+ \\ & +Dee'Du + (u - \nu e)_+ - (-u)_+. \end{array} \tag{8}$$

$$\begin{array}{rl} \partial^2 f(u) = & DK(A,B')diag(K(B,A')Du - e)_* K(B,A')D \\ & +DK(A,B')diag(-K(B,A')Du - e)_* K(B,A')D \\ & +Dee'D + diag((u - \nu e)_* + (-u)_*) \\ = & DK(B,A')diag(|K(B,A')Du| - e)_*)K(B,A')D \\ & +Dee'D + diag((u - \nu e)_* + (-u)_*), \end{array} \tag{9}$$

where the last equality follows from the equality:

$$(a-1)_* + (-a-1)_* = (|a|-1)_*. \tag{10}$$

We state now our generalized Newton algorithm for solving the unconstrained minimization problem (5) for which we have introduced a regularization [23] parameter δ for purposes of numerical stability.

Algorithm 2.2 **Generalized Newton Algorithm for (5)** Let $f(u)$, $\nabla f(u)$ and $\partial^2 f(u)$ be defined by (5),(8) and (9). Set the parameter values ν, ϵ, δ, tolerance *tol*, and *imax* (typically: $\epsilon = 4 \times 10^{-4}$ for linear SVMs and 1 for nonlinear SVMs, $tol = 10^{-6}$, $imax = 1000$, while ν and δ are set by a tuning procedure). 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 [1] stepsize $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is such that:

$$f(u^i) - f(u^i + \lambda_i d^i) \geq -\frac{\lambda_i}{4}\nabla f(u^i)' d^i, \tag{11}$$

and d^i is the modified Newton direction:

$$d^i = -(\partial^2 f(u^i) + \delta I)^{-1}\nabla f(u^i). \tag{12}$$

In other words, start with stepsize $\lambda_i = 1$ and keep multiplying λ_i by $\frac{1}{2}$ until (11) is satisfied.

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

(iii) Define the solution of the 1-norm SVM (1) with least quadratic perturbation (7) by (6) with $u = u^i$.

Note that in practice instead of using the explicit inverse $(\partial^2 f(u^i) + \delta I)^{-1}$ in Step (i) of Algorithm 2.2 we can use the *backslash* notation "\" of MATLAB [19] $(\partial^2 f(u^i) + \delta I)\backslash\nabla f(u^i)$ which does not usually require computation of the inverse.

We state a convergence result for this algorithm now [17, Propositon 4].

Proposition 2.3 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 2.2 solves the exterior penalty problem (5). The corresponding $(\bar{v}, \bar{\gamma}, \bar{y})$ obtained by setting u to $\bar{u}$ in (6) is an exact solution to the primal 1-norm SVM (1) which in addition minimizes the quadratic perturbation (7) over the solution set of (1).

We turn now to our chunking method for handling massive datasets.

3 Chunking Method

We shall apply the chunking approach proposed in [2] to our unconstrained minimization approach of Section 2. We consider a general linear program

$$\min_x \left\{ c^T x \,|Hx \geq b \right\}, \tag{13}$$

where $c \in R^n$, $H \in R^{m \times n}$ and $b \in R^m$. We state now our chunking algorithm and establish its finite termination for the linear program (13), where m may be orders of magnitude larger than n. In its dual form our algorithm can be interpreted as a block-column generation method related to column generation methods of Gilmore-Gomory [9], Dantzig-Wolfe [6], [4, pp 198-200,428-429] and others [20, pp 243-248], but it differs from active set methods [14, pp 326-330] in that it does not require the satisfaction of a working set of constraints as equalities.

Algorithm 3.1 **LPC: Linear Programming Chunking Algorithm for (13)** Let $[H \quad b]$ be partitioned into ℓ blocks, possibly of different sizes, as follows:

$$\begin{bmatrix} H & b \end{bmatrix} = \begin{bmatrix} H^1 & b^1 \\ \vdots & \vdots \\ H^\ell & b^\ell \end{bmatrix}.$$

Assume that (13) and all subproblems (14) below, have vertex solutions. At iteration $j = 1, \ldots$ compute x^j by solving the following linear program:

$$x^j \in \arg \text{ vertex } \min \left\{ c^T x \,\middle|\, \begin{matrix} H^{(j \bmod \ell)} \, x & \geq b^{(j \bmod \ell)} \\ \bar{H}^{(j \bmod \ell)-1} \, x & \geq \bar{b}^{(j \bmod \ell)-1} \end{matrix} \right\}, \tag{14}$$

where $[\bar{H}^0 \quad \bar{b}^0]$ is empty and $[\bar{H}^j \quad \bar{b}^j]$ is the set of active constraints (that is all inequalities of (14) satisfied as equalities by x^j) with positive optimal Lagrange multipliers at iteration j. Stop when $c^T x^j = c^T x^{j+\sigma}$ for some input integer σ. Typically $\sigma = 4$.

Theorem 3.1 **Finite Termination of LPC Algorithm** The sequence $\{x^j\}$ generated by the LPC Algorithm 3.1 has the following properties:

(i) The sequence $\{c^T x^j\}$ of objective function values is nondecreasing and is bounded above by the global minimum of $\min_x \left\{ c^T x \,|Hx \geq b \right\}$.

(ii) The sequence of objective function values $\{c^T x^j\}$ becomes constant, that is: $c^T x^{j+1} = c^T x^j$ for all $j \geq \bar{j}$ for some $\bar{j} \geq 1$.

(iii) For $j \geq \bar{j}$, active constraints of (14) at x^j with positive multipliers remain active for iteration $j+1$.
(iv) For all $j \geq \tilde{j}$, for some $\tilde{j} \geq \bar{j}$, x^j is a solution of the linear program (13) provided all active constraints at x^j have positive multipliers for $j \geq \bar{j}$.

The proof of this theorem is given in [2] and is directly applicable to solving our linear program (4) by solving successive small linear programs obtained from chunks of the constraints of (4) using the formulation (5) and Algorithm 3.1.

Before describing our computational experience we outline how the chunking procedure will be applied to the linear programming formulation (4) of the 1-norm SVM. First we break the linear program (4) into chunks as outlined in the LPC Algorithm 3.1. Then for each linear programming chunk that needs to be solved by the LPC Algorithm 3.1 we utilize the Generalized Newton Algorithm 2.2 to solve it exactly by minimizing its unconstrained exterior dual penalty function which is of the form of the minimization problem (5).

We turn to our computational results now.

4 Computational Results

Before giving implementation and computational results for our Chunking Algorithm 3.1 for solving (4)-(5) we would like to describe some computational aspects of the underlying Generalized Newton Algorithm 2.2 for smaller classification problems. As reported in [17], a 1-norm linear SVM classifier was obtained by Algorithm 2.2 on six public datasets, with points numbering between 297 and 4192, in time that was an average 9.6 times faster than that of the state-of-the-art linear programming package CPLEX 9.0 [12]. This favorable comparison with a classical simplex or interior point method used in CPLEX 9.0 motivated the idea of utilizing the Generalized Newton Algorithm 2.2, which requires only a simple linear equation solver, as the best candidate for the proposed chunking approach.

We turn now to implementation details and computational results for our chunking Algorithm 3.1.

In implementing Algorithm 3.1, we note that the constraints of (4) can be grouped such that, for each constraint defined by A_j and D_{jj}, there is a corresponding y_j. Therefore, each chunk is selected such that both the constraints $D_{jj}(K(A_j, B')D(r-s)-\gamma)+y_j \geq 1$ and $y_j \geq 0$ are always included together in the chunk of constraints. Also, we always include the constraints $r \geq 0$ and $s \geq 0$. We found that defining the chunks in this manner leads to the chunking algorithm terminating in fewer iterations than with an entirely random constraint selection method. Furthermore, the problem maintains the form

of (1), allowing us to use Proposition 2.1 and Algorithm 2.2 to perform the minimization of each chunk.

We tested our algorithm on multiple massive datasets based on normally distributed clusters on cubes (NDCC) data generator [22]. NDCC generates datasets using three 1-norm cubes of increasing size centered around the origin. It then generates multivariate normal distributions around each vertex where the distributions that are centered around opposing vertices of each cube belong to opposite classes. In addition, instead of distributing the number of points equally to each center, some centers are chosen to generate a relatively small number of points, while the center on the opposite vertex of the cube generates a relatively large number of points. An example of a dataset generated using NDCC can be seen in Figure 1.

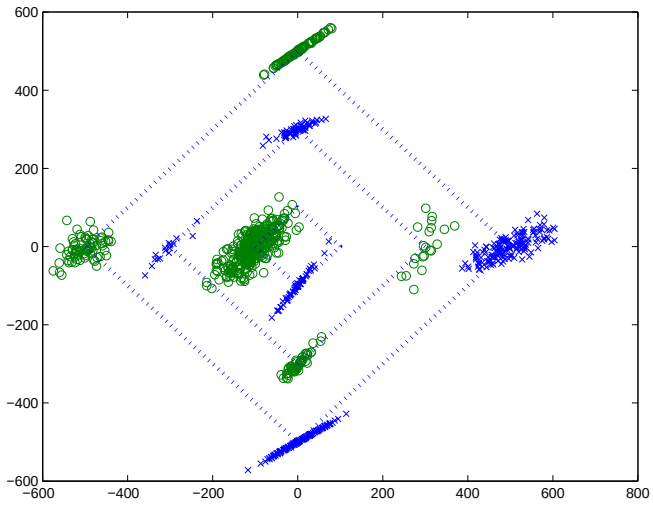

Figure 1. **Example of a NDCC dataset generated in R^2. Clusters of normally distributed points are centered around the vertices of each of the three 1-norm cubes (squares in R^2) shown in dotted lines.**

We ran Algorithm 3.1 on each dataset using tenfold cross validation and give our results in Table 1. In solving the 1-norm nonlinear SVM via unconstrained minimization, we set the parameter values to be $\nu = 2^5$, $\mu = 2^{-20}$, $\delta = 10^{-6}$, $\epsilon = 1$, $tol = 10^{-6}$, and $imax = 1000$. In addition, the number of rows of the matrix B in the reduced kernel $K(A, B')$ was fixed at 90 regardless of the number of points in the dataset. In implementing the chunking, we used a 10% chunk size, which means it takes 10 chunking iterations to go through all the points in the dataset once. As in [2], we found that it was

only necessary to incorporate into $[\bar{H}^j \quad \bar{b}^j]$ the active constraints without determining whether the optimal Lagrange multipliers were positive. Also, instead of stopping when $c'x^j = c'x^{j+\sigma}$, we stopped when $\frac{|c'x^j - c'x^{j+1}|}{|c'x^j|} \leq 0.01$ for three consecutive iterations. This led to faster termination and did not affect the accuracy of our solutions. In our tests, we found that the execution time of Algorithm 3.1 grew at a rate less than the square of the increase of the number of points in the dataset. We also compared our results with those of a linear classifier and results using Algorithm 3.1 on a random subset of 1% of the original data points and testing the classifier, obtained on the 1% subset, on 10% of the original data. These results are given in Table 2. Note that by using Algorithm 3.1, error is reduced by more than 37% compared to the linear SVM and more than 26% compared to running the nonlinear SVM on a random subset of the points.

Dataset	NDCC		
# of Points	250000	100000	50000
Time (hours)	56.2	11.9	3.5
Iterations	14	14	14
Training error	13.82%	13.64%	14.28%
Testing error	13.81%	13.67%	14.31%
# of Kernel Functions used	89.7	89.1	88.7

Table 1. **Results using Algorithm 3.1 on test problems using the NDCC dataset generator in R^{10}, averaged over ten folds. Note that time grows less than the square of the increase of the number of points in the dataset.**

Dataset	NDCC		
# of Points	250000	100000	50000
Algorithm 3.1	13.81%	13.67%	14.31%
Algorithm 3.1 using 1% sample of points	18.91%	19.38%	20.77%
1-norm linear SVM	22.43%	22.61%	22.74%

Table 2. **Comparison of testing set error results on each dataset using various algorithms. Algorithm 3.1 reduces the error by more than 26% compared to using a small sample of the given points and by more than 37% compared to a linear SVM.**

Since we are using tenfold cross validation, a problem with 90% of the given points is being solved for each fold. So, for our 100000-point dataset, we are training on 90000 points, which, when using the formulation of (4), results in a problem with 90181 variables and 90000 constraints, excluding bounds, with $B \in R^{90\times 10}$. Figure 2 depicts a typical run of Algorithm 3.1 on one fold of the 100000-point NDCC dataset and demonstrates how the objective value of the linear program increases as active constraints are added and others are dropped in each iteration. It also shows the quick leveling off of the objective

function value as well as the number of active constraints despite the fact that the algorithm continuously drops and adds constraints. Both of these are important factors in the practical viability of the algorithm. Figure 3 shows the error on one fold of the 100000-point NDC dataset using Algorithm 3.1. This demonstrates how the error decreases as the algorithm progresses and then stabilizes before the algorithm terminates.

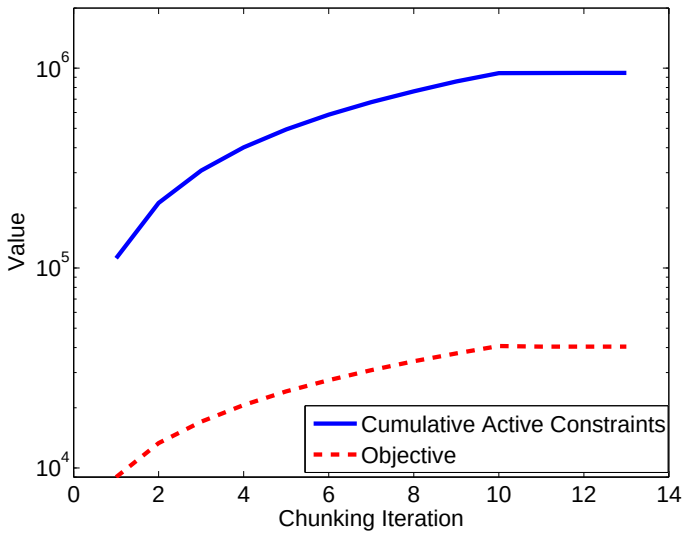

Figure 2. **Number of active constraints ond objective function values versus number of iterations for the 100000-point NDCC dataset. The value of the objective function steadily increases and reaches its optimal value after only ten chunking iterations. Note the logarithmic scale.**

Comparing Algorithm 3.1 to other methods, all of our datasets caused CPLEX 9.0 [12], a state-of-the-art linear programming code, to fail with an out-of-memory error.

5 Conclusion and Outlook

We have proposed an approach that effectively classifies massive datasets of up to 250000 points using nonlinear kernels with 1-norm support vector machines. Our method solves these problems by breaking a huge constraint set into chunks and solving the resulting linear programs by a simple linear equation solver. Our approach solves problems that cause conventional state-of-the-art methods to fail and produces less error than competing methods. Possible

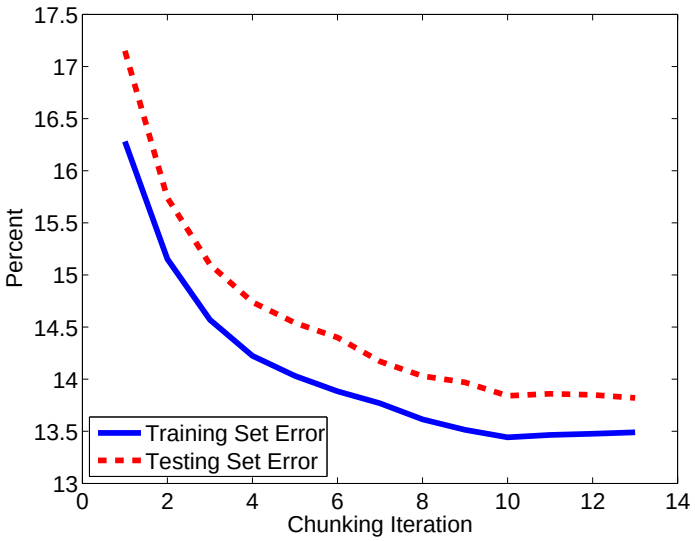

Figure 3. **Training and testing set error versus number of iterations for the 100000-point NDCC dataset. The error steadily decreases as the algorithm progresses and then stabilizes before the algorithm terminates.**

future work includes modifying the constraint inclusion criterion to include only a fraction of the active constraints based on the Lagrange multipliers, allowing even larger datasets to be classified.

References

[1] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[2] 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/98-05.ps.

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

[4] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

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

[6] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[7] F. Facchinei. Minimization of SC^1 functions and the Maratos effect. *Operations Research Letters*, 17:131–137, 1995.

[8] G. Fung and O. L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, July 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-03.ps.

[9] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[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] S.Y. Huang and Y.-J. Lee. Theoretical study on reduced support vector machines. Technical

report, National Taiwan University of Science and Technology, Taipei, Taiwan, 2004. yuh-jye@mail.ntust.edu.tw.

[12] ILOG, Incline Village, Nevada. *ILOG CPLEX 9.0 User's Manual*, 2003. http://www.ilog.com/products/cplex/.

[13] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM*, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

[14] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison–Wesley, second edition, 1984.

[15] 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.

[16] 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.

[17] O. L. Mangasarian. Exact 1-Norm support vector machines via unconstrained convex differentiable minimization. Technical Report 05-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, August 2005. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/05-03.ps. Journal of Machine Learning Research 7, 2006, 1517-1530.

[18] O. L. Mangasarian and M. E. Thompson. Massive data classification via unconstrained support vector machines. *Journal of Optimization Theory and Applications*, 131:315–325, 2006. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/06-01.pdf.

[19] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2006. http://www.mathworks.com.

[20] K. G. Murty. *Linear Programming*. John Wiley & Sons, New York, 1983.

[21] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[22] M. E. Thompson. NDCC: Normally distributed clustered datasets on cubes, 2006. www.cs.wisc.edu/dmi/svm/ndcc/.

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

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