Massive Data Classification via Unconstrained Support Vector Machines

Olvi L. Mangasarian and Michael E. Thompson
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
(olvi,thompson)@cs.wisc.edu

ABSTRACT

A highly accurate algorithm, based on support vector machines formulated as linear programs [13, 1], is proposed here as a completely unconstrained minimization problem [15]. Combined with a chunking procedure [2] this approach, which requires nothing more complex than a linear equation solver, leads to a simple and accurate method for classifying million-point datasets. Because a 1-norm support vector machine underlies the proposed approach, the method suppresses input space features as well. A state-of-the-art linear programming package, CPLEX [10], fails to solve problems handled by the proposed algorithm.

1. INTRODUCTION

One of the principal advantages of 1-norm support vector machines (SVMs) is that unlike conventional 2-norm SVMs [4, 22, 21], they are very effective in reducing input space features for linear kernels [1, 7]. We utilize here an exact completely unconstrained differentiable minimization formulation of 1-norm SVMs [15] coupled with a chunking procedure, which allows us to handle massive datasets.

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 us to handle massive datasets. In Section 4 we justify our use of the unconstrained approach and present our numerical results for massive datasets. Section 5 concludes the paper.

First we 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 $x \in R^n$, $\|x\|_1$, $\|x\|$ and $\|x\|_\infty$, will denote the $1-$, $2-$ and $\infty-$ 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 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 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 merely Lipschitz continuous with a Lipschitz constant of $\|A'\|\,\|A\| + \|P\|$. However, one can define its generalized Hessian [9, 6, 14] 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 $(A_ix-b_i)_*$, $i = 1,\ldots,m$. The generalized Hessian has many of the properties of the regular Hessian [9, 6, 14] 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. 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 $\mathcal{B}$.

2. 1-NORM SVM AS AN UNCONSTRAINED MINIMIZATION PROBLEM

We consider first the 1-norm linear SVM binary classification problem [13, 1, 7]:

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)} & \nu\|y\|_1 + \|w\|_1 & & \\ \text{s.t.} & D(Aw-e\gamma)+y & \geq & e \\ & y & \geq & 0, \end{array} \tag{1}$$

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

$$x'w = \gamma, \tag{2}$$

according to the class of each row of A as given by the $m\times m$ diagonal matrix D with elements $D_{ii} = \pm 1$. The objective term $\|y\|_1$ minimizes the classification error weighted with the positive parameter ν while the term $\|w\|_1$ maximizes the ∞-norm margin [12] between the bounding planes $x'w = \gamma \pm 1$ that approximately bound each of the two classes of points represented by A. It is well known [1, 7] that using $\|w\|_1$ in the objective function of (1) instead of the standard 2-norm squared term $\|w\|^2$ [4, 22, 21] results in input space feature selection by suppressing many components of w, whereas the the standard 2-norm SVM does not suppress any components of w in general. We convert (1) to an explicit linear program as in [7] by setting:

$$w = p - q, \;\; p \geq 0, \; q \geq 0, \tag{3}$$

which results in the linear program:

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

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 [15, Proposition 2] to reduce our linear program to a completely unconstrained minimization problem.

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

$$\begin{array}{l} \min\limits_{u \in R^m} \; -\epsilon e'u + \dfrac{1}{2}(\|(A'Du - e)_+\|^2 + \|(-A'Du - e)_+\|^2 + \\ \qquad\qquad (-e'Du)^2 + \|(u - \nu e)_+\|^2 + \|(-u)_+\|^2), \end{array} \tag{5}$$

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

$$\begin{array}{lcl} w = p - q = & = & \frac{1}{\epsilon}((A'Du - e)_+ - (-A'Du - e)_+), \\ \gamma & = & -\frac{1}{\epsilon}e'Du, \\ y & = & \frac{1}{\epsilon}(u - \nu e)_+. \end{array} \tag{6}$$

In addition this (w, γ, y) minimizes:

$$\|w\|^2 + \gamma^2 + \|y\|^2 + \|D(Aw - e\gamma) + y - e\|^2, \tag{7}$$

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.

$$\begin{array}{rcl} \nabla f(u) & = & -\epsilon e + DA(A'Du - e)_+ - DA(-A'Du - e)_+ \\ & & +Dee'Du + (u - \nu e)_+ - (-u)_+. \end{array} \tag{8}$$

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

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) as follows.

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) respectively. Set the parameter values ν, ϵ, δ, tolerance* tol, *and* imax.

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:*

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

and d^i is the modified Newton direction:

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

(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 the key step (12) in the above algorithm requires merely a simple linear equation solver such as the one available in MATLAB [16].

We state a convergence result for this algorithm now [15, 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 $(\bar{w}, \bar{\gamma}, \bar{y})$ 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

$$\min_x \{c'x \,|\, Hx \geq b\}, \tag{13}$$

where $c \in R^n$, $H \in R^{m \times n}$ and $b \in R^m$. We state now our chunking algorithm and give its finite termination for the linear program (13), where m could 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 [8], Dantzig-Wolfe [5], [3, pp 198-200,428-429] and others [19, pp 243-248], but it differs from active set methods [11, 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:*

$$\left[\begin{array}{cc} H & b \end{array}\right] = \left[\begin{array}{cc} H^1 & b^1 \\ \vdots & \vdots \\ H^\ell & b^\ell \end{array}\right].$$

Assume that (13) and all subproblems (14) below, have vertex solutions. At iteration $j = 1, \dots$ compute x^j as a vertex solution of the following linear program:

$$min\left\{ c'x \;\middle|\; \begin{array}{ccc} H^{(j\ modulo\ \ell)}x & \geq & b^{(j\ modulo\ \ell)} \\ \bar{H}^{(j\ modulo\ \ell)-1}x & \geq & \bar{b}^{(j\ modulo\ \ell)-1} \end{array} \right\}, \tag{14}$$

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 $c'x^j = c'x^{j+\mu}$ for some input integer μ.

This algorithm terminates at an optimal solution as follows.

Theorem 3.2. **Finite Termination of LPC Algorithm** *The sequence $\{x^j\}$ generated by the LPC Algorithm 3.1 has the following properties:*

(i) The sequence $\{c'x^j\}$ of objective function values is non-decreasing and is bounded above by the global minimum of $\min\limits_{x} \{c'x \,|Hx \geq b\}$.

(ii) The sequence of objective function values $\{c'x^j\}$ becomes constant, that is: $c'x^{j+1} = c'x^j$ 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 $j \geq \tilde{\bar{j}}$, for some $\tilde{\bar{j}} \geq \bar{j}$, 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 2.2. We note that Algorithm 2.2 may not necessarily find vertex solutions as required by the finite termination Theorem 3.2, but this does not impede the finite termination of our Algorithm 3.1 as shown by our computational results.

We turn to our computational results now.

4. COMPUTATIONAL METHODS AND 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 [15], 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 [10]. In addition, the average number of features used by Algorithm 2.2 for the six datasets was 53.4% of the original number of features compared to 79.0% of features used by CPLEX 9.0. 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 $D_{jj}(A_j(p-q)-\gamma)+y_j \geq 1$ and $y_j \geq 0$ are always included together in the chunk of constraints. Also, we always include the constraints $p \geq 0$ and $q \geq 0$. We found that defining the chunks in this manner leads to the chunking algorithm terminating using 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. First, the million-point dataset in R^{32} used in [2], which was generated using the Synthetic Classification Data Set Generator (SCDS) [17]. Of the 32 attributes, 4 are relevant, and there is a 10% ratio of misclassified points. We also used three datasets in R^{32} generated using the Normally Distributed Clusters (NDC) data generator [20]. NDC generates a series of random centers for multivariate normal distributions and a random fraction of data points from each center. It then chooses the class for each center based on a random separating plane. NDC also provides an estimate of separability for the dataset it generates. The datasets we used each had an estimated separability of around 80%. Also, in order to better compare our NDC datasets with the SCDS dataset, we modified the NDC generator so it generated datasets with 4 relevant attributes by generating a dataset with 4 attributes and then adding 28 attributes consisting of uniformly distributed random points over the same range as the original 4.

We ran Algorithm 3.1 on each dataset using tenfold cross validation. In solving the 1-norm linear SVM via unconstrained minimization, we set the parameter values to be $\nu = 2^{-12}$ (approximately $2.44 * 10^{-4}$), $\delta = 10^{-3}$, $\epsilon = 10^{-4}$, $tol = 10^{-6}$, and $imax = 1000$. 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 $c'x^j = c'x^{j+\mu}$, 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. Our results are given in Table 1.

Since we are using tenfold cross validation, a problem with 90% of the given points is being solved for each fold. So, for our million-point datasets, we are training on 900000 points, which, when using the formulation of (4), results in a problem with 900065 variables and 900000 constraints, excluding bounds. Figure 1 depicts a typical run of Algorithm 3.1 on one fold of the million-point SCDS 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 2 shows the correctness on one fold of the 500000-point NDC dataset using Algorithm 3.1. This demonstrates how the correctness increases as the algorithm progresses and then stabilizes before the algorithm terminates.

Comparing Algorithm 3.1 to other methods, all of our datasets caused CPLEX 9.0 [10], a state-of-the-art linear programming code, to fail with an out-of-memory error. In comparing our results to those of [2], we can now solve the SCDS problem in approximately 2.21 hours and 14 iterations on a 3.0 Ghz Pentium IV machine with 2GB RAM running CentOS 4 Linux and MATLAB 7.1. This compares favorably to the previous results [2] of 231.32 hours and 63 iterations, performed on a cluster of 4 Sun Enterprise E6000 machines, with a total of 64 UltraSPARC II processors and 8 GB of RAM using a simplex based MINOS linear programming package [18].

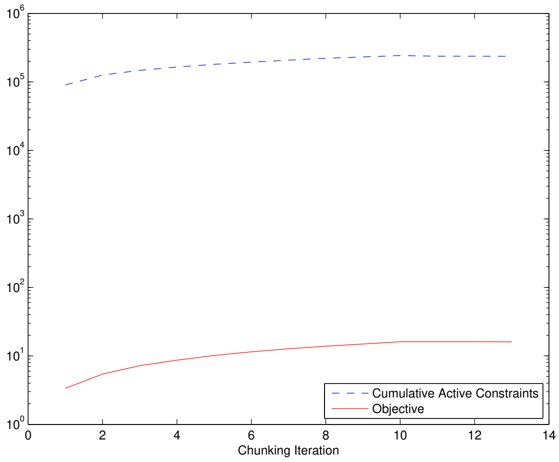

Figure 1: Number of active constraints and objective function values versus number of iterations for the million-point SCDS dataset. The objective function becomes stable after only ten chunking iterations, that is, going through the entire dataset once. Note the logarthmic scale.

Dataset	SCDS	NDC		
# of Points	1000000	1000000	750000	500000
Time (hours)	2.21	4.72	3.49	2.14
Iterations	14	14	14	14
Training corr.	98.205%	91.228%	91.217%	91.214%
Testing corr.	98.204%	91.228%	91.210%	91.213%
Features used	8.0	29.4	28.5	27.9

Table 1: Results using Algorithm 3.1 on synthetic test problems in R^{32}, averaged over ten folds.

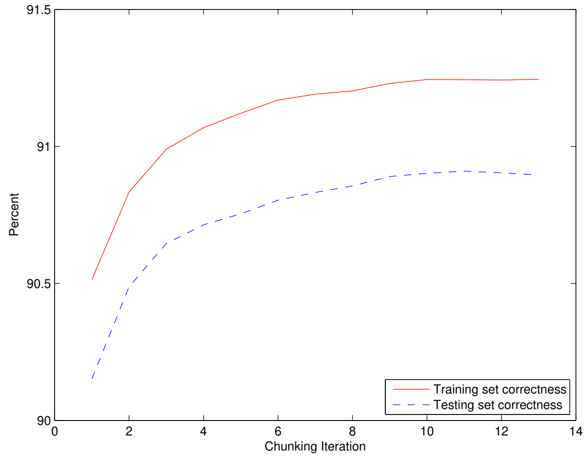

Figure 2: Training and testing set correctness versus number of iterations for the 500000-point NDC dataset. The correctness increases as the algorithm progresses and quickly stabilizes.

5. CONCLUSION AND OUTLOOK

We have proposed an approach that effectively classifies massive datasets of up to one million points using feature-reducing 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. Future extensions of these ideas include applying these methods to SVMs with nonlinear kernels and improving the chunking method to allow for even larger datasets to be classified.

Acknowledgments

The research described in this Data Mining Institute Report 06-01, March 2006, was supported by National Science Foundation Grants CCR-0138308 and IIS-0511905, the Microsoft Corporation and ExxonMobil.

6. REFERENCES

[1] 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.

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

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

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

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

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.psa. Journal of Machine Learning Research 7, 2006, 1517-1530.

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

[17] G. Melli. Synthetic classification data sets (SCDS). Technical report, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada, 1997. http://fas.sfu.ca/cs/people/GradStudents/melli/SCDS/, ftp://ftp.cs.wisc.edu/pub/dmi/data/SCDS/.

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

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

[20] D. R. Musicant. NDC: Normally distributed clustered datasets, 1998. http://www.cs.wisc.edu/dmi/svm/ndc/.

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

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