Data Discrimination via Nonlinear Generalized Support Vector Machines

O. L. Mangasarian and David R. Musicant
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
olvi@cs.wisc.edu, musicant@cs.wisc.edu

Abstract

The main purpose of this paper is to show that new formulations of support vector machines can generate nonlinear separating surfaces which can discriminate between elements of a given set better than a linear surface. The principal approach used is that of generalized support vector machines (GSVMs) which employ possibly indefinite kernels [17]. The GSVM training procedure is carried out by either the simple successive overrelaxation (SOR) [18] iterative method or by linear programming. This novel combination of powerful support vector machines [24, 5] with the highly effective SOR computational algorithm [15, 16, 14] or with linear programming allows us to use a nonlinear surface to discriminate between elements of a dataset that belong to one of two categories. Numerical results on a number of datasets show improved testing set correctness, by as much as a factor of two, when comparing the nonlinear GSVM surface to a linear separating surface.

1 Introduction

A very simple convex quadratic program with *only* nonnegativity constraints was obtained in [18] by taking the dual of the quadratic program associ-

ated with a support vector machine [24, 5]. The margin (distance between bounding separating planes) was maximized with respect to *both* the normal to the separating planes as well as their location. This quadratic program was solved by successive overrelaxation (SOR) for datasets with as many as 10 million points in [18] to obtain a *linear* separating surface. In the present paper we show how to obtain a *nonlinear* separating surface for more complex datasets by using SOR as well as by linear programming. The nonlinear separating surface is induced by using a possibly indefinite kernel of a generalized support vector machine [17] and is computed by a linearly convergent successive overrelaxation algorithm [15, 14] applied to the associated convex quadratic dual problem with bound constraints only, or by the finite simplex algorithm [6] applied to a linear programming formulation of the nonlinear generalized support vector machine problem.

In Section 2 we first state our discrimination problem as a standard dual support vector machine problem (1) and introduce our variant of the problem (4): an SOR-solvable convex quadratic program with bound constraints only when the kernel is positive semidefinite. We then introduce another version of our generalized support vector machine [17] (5) in which the kernel is completely arbitrary but the problem is convex and hence also SOR-solvable. We note that in our previous work [18] only a *linear kernel* (polynomial kernel of degree one) was implemented, whereas various *nonlinear kernels* that may be indefinite will be used in this paper. In Section 3 we state our SOR algorithm for nonlinear kernels and establish its linear convergence. In Section 4 we give linear programming formulations of the generalized support vector machine problem, including a novel one that includes two or more arbitrary kernels. In Section 5 we give numerical results for various datasets all of which show the improvement obtained in using the generalized support vector machine with a nonlinear kernel over a linear one. Section 6 draws some conclusions and points out future directions such as parallel SOR implementations that may lead to the solution of very large discrimination problems with nonlinear separating surfaces.

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 $(x_+)_i = \max\{0, x_i\}$, $i = 1, \ldots, n$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For an $m \times n$ matrix A, A_i will denote the ith row of A and $A_{\cdot j}$ will denote the jth column of A. The identity matrix in a real space of arbitrary dimension will be

denoted by I, while a column vector of ones of arbitrary dimension will be denoted by e. We shall employ the MATLAB "dot" notation [19] to signify application of a function to all components of a matrix or a vector. For example if $A \in R^{m\times n}$, then $A^2_\bullet \in R^{m\times n}$ will denote the matrix obtained by squaring each element of A. 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 $R^{m\times \ell}$. In particular if x and y are column vectors in R^n then, $K(x', A')$ is a row vector in R^m, $K(x', y)$ is a real number and $K(A, A')$ is an $m \times m$ matrix. Note that for our purposes here $K(A, A')$ will be assumed to symmetric, that is $K(A, A') = K(A, A')'$. For a vector x in the n-dimensional real space R^n, the step function $x_{\bullet *}$ is defined as a vector in R^n of minus ones, zeros and plus ones, corresponding to negative, zero and positive components of x respectively. Similarly for an $m \times n$ matrix M, $M_{\bullet *}$ will denote an $m \times n$ matrix of minus ones, zeros and plus ones.

2 The Support Vector Machine and its Generalization: Quadratic Formulation

We consider the problem of discriminating between 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 positive semidefinite kernel $K(A, A')$ [24, 5] is given by the following dual convex quadratic program for some $\nu > 0$:

$$\begin{array}{rll} \min\limits_{u\in R^m} & \frac{1}{2}u'DK(A,A')Du - e'u & \\ \text{s.t.} & e'Du = 0 & \\ & 0 \le u \le \nu e. & \end{array} \tag{1}$$

A solution u of this quadratic program leads to the nonlinear separating surface

$$K(x', A')Du = \gamma, \tag{2}$$

where γ is defined [18, Equation (7)] by a solution of:

$$\min_{\gamma\in R} e'(e - D(K(A,A')Du - e\gamma))_+. \tag{3}$$

Two significant changes to this support vector machine formulation were made recently. In [18] the equality constraint of (1) was incorporated into the

objective function of an SVM problem with a *linear* kernel, as a consequence of maximizing the distance between the bounding separating planes with respect to *both* the normal *and* the location of the separating planes. By a similar approach we obtain the following simplified version of (1) with bound constraints only and a *nonlinear* kernel:

$$\begin{gathered}\min_u \frac{1}{2}u'D[K(A,A')+ee']Du-e'u,\ \text{s.t. } 0\le u\le \nu e\\ (\gamma=-e'Du),\end{gathered} \tag{4}$$

with an explicit expression for γ and the same separating surface (2). The formulation (4) allows a direct use of the SOR algorithm to solve very large problems.

The second significant change in (1), proposed in [17], allowed the use of possibly indefinite kernels thereby substantially widening the range of nonlinear separating surfaces that can be utilized. One particular formulation [17, Equation (10)] is the following one which, as in (4), incorporates the equality $e'Du=0$ into the objective function:

$$\begin{gathered}\min_u \frac{1}{2}u'D[K(A,A')^2+ee']Du-e'u,\ \text{s.t. } 0\le u\le \nu e\\ (\gamma=-e'Du),\end{gathered} \tag{5}$$

with a separating surface [17, Equations (1) & (10)] different from (2):

$$K(x',A')K(A,A')Du=\gamma. \tag{6}$$

Note that the kernel $K(A,A')$ in the formulation (5) is completely arbitrary and need not satisfy any condition in order for the objective function of (5) to be convex. This makes the separating surface (6) quite general.

3 Successive Overrelaxation for Nonlinear GSVM

We shall apply the SOR algorithm, which was used effectively in [18] for linear support vector machines, to the nonlinear SVMs (4) and (5). Both of

these problems can be stated as:

$$\min_u \frac{1}{2}u'Mu - e'u, \text{ s.t. } u \in S = \{u \mid 0 \leq u \leq \nu e\}, \tag{7}$$

with the symmetric matrix M defined as $D(K(A,A')+ee')D$ and $D(K(A,A')^2+ee')D$ respectively. Thus M will be positive semidefinite if we assume that $K(A,A')$ is positive semidefinite in the former case and under no assumptions in the latter case. If we decompose M as follows:

$$M = L + E + L', \tag{8}$$

where the nonzero elements of $L \in R^{m\times m}$ constitute the strictly lower triangular part of the symmetric matrix M, and the nonzero elements of $E \in R^{m\times m}$ constitute the diagonal of M, then a necessary and sufficient optimality condition for (7) for positive semidefinite M is the following gradient projection optimality condition [23, 14]:

$$u = (u - \omega E^{-1}(Mu - e))_{\#}, \ \omega > 0, \tag{9}$$

where $(\cdot)_{\#}$ denotes the 2-norm projection on the feasible region S of (8), that is:

$$((u)_{\#})_i = \left\{ \begin{array}{l} 0 \ \text{ if } u_i \leq 0 \\ u_i \ \text{ if } 0 < u_i < \nu \\ \nu \ \text{ if } u_i \geq \nu \end{array} \right\}, \ i = 1, \ldots, m. \tag{10}$$

Our SOR method, which is a matrix splitting method that converges linearly to a point $\bar{u}$ satisfying (9), consists of splitting the matrix M into the sum of two matrices as follows:

$$M = \omega^{-1}E(B + C), \text{ s.t. } B - C \text{ is positive definite.} \tag{11}$$

For our specific problem we take:

$$B = (I + \omega E^{-1}L), \ C = ((\omega - 1)I + \omega E^{-1}L'), \ 0 < \omega < 2. \tag{12}$$

This leads to the following linearly convergent [14, Equation (3.14)] matrix splitting algorithm:

$$u^{i+1} = (u^{i+1} - Bu^{i+1} - Cu^i + \omega E^{-1}e)_{\#}, \tag{13}$$

for which

$$B + C = \omega E^{-1}M, \ B - C = (2 - \omega)I + \omega E^{-1}(L - L'). \tag{14}$$

Note that for $0 < \omega < 2$, the matrix $B+C$ is positive semidefinite and matrix $B-C$ is positive definite. The matrix splitting algorithm (13) results in the following easily implementable SOR algorithm once the values of B and C given in (12) are substituted in (13).

Algorithm 3.1 SOR Algorithm *Choose $\omega \in (0,2)$. Start with any $u^0 \in R^m$. Having u^i compute u^{i+1} as follows:*

$$u^{i+1} = (u^i - \omega E^{-1}(Mu^i - e + L(u^{i+1} - u^i)))_\#, \tag{15}$$

until $\|u^{i+1} - u^i\|$ is less than some prescribed tolerance.

Remark 3.2 *The components of u^{i+1} are computed in order of increasing component index. Thus the SOR iteration (15) consists of computing u_j^{i+1} using $(u_1^{i+1}, \ldots, u_{j-1}^{i+1}, u_j^i, \ldots, u_m^i)$. That is, the latest computed components of u are used in the computation of u_j^{i+1}. The strictly lower triangular matrix L in (15) can be thought of as a substitution operator, substituting $(u_1^{i+1}, \ldots, u_{j-1}^{i+1})$ for $(u_1^i, \ldots, u_{j-1}^i)$.*

We have immediately from [14, Proposition 3.5] the following linear convergence result.

Theorem 3.3 SOR Linear Convergence *The iterates $\{u^i\}$ of the SOR Algorithm 3.1 converge R-linearly to a solution of $\bar{u}$ of the dual problem (6), and the objective function values $\{f(u^i)\}$ of (6) converge Q-linearly to $f(\bar{u})$. That is for $i \geq \bar{i}$ for some $\bar{i}$:*

$$\begin{array}{rcl} \|u^i - \bar{u}\| & \leq & \mu\delta^i, \text{ for some } \mu > 0, \ \delta \in (0,1), \\ f(u^{i+1}) - f(\bar{u}) & \leq & \tau(f(u^i) - f(\bar{u})), \text{ for some } \tau \in (0,1). \end{array} \tag{16}$$

We turn our attention to linear programming formulation of the nonlinear GSVM.

4 The Nonlinear GSVM as a Linear Program

We briefly describe now how the nonlinear separating surface (1) can also be generated by a linear program [17]. The linear programming formulation,

which allows completely arbitrary kernels, surrogates support vector suppression for maximizing the distance between separating bounding planes. One such formulation is given by the linear program [17, Equation (11)]:

$$\begin{array}{rll} \min\limits_{u,\gamma,y,s} & \nu e'y + e's & \\ \text{s.t.} & D(K(A,A')Du - e\gamma) + y \geq e & \\ & s \geq u \geq -s & \\ & y \geq 0. & \end{array} \tag{17}$$

A solution (u, γ, y, s) to this linear program for a completely arbitrary kernel $K(A, A')$ will provide a separating surface given by (2). We note that if the linear kernel (i.e. polynomial kernel of degree 1) $K(A, A') = AA'$ is used in the linear program above and w is introduced and set equal to $A'Du$ we obtain the high-performing 1-norm linear SVM proposed in [4] and utilized successfully in [3, 1, 2].

Since the linear programming formulation does not impose restrictions on the kernel employed, we are free to use more than one kernel in a given formulation. For example we can use **both** a linear kernel and a discontinuous neural network kernel in a single linear programming formulation. More generally if we let $K^1(A, A')$ and $K^2(A, A')$ be two arbitrary kernels, then the corresponding linear programming formulation becomes:

$$\begin{array}{rll} \min\limits_{u^1,u^2,\gamma,y,s^1,s^2} & \nu e'y + e's^1 + e's^2 & \\ \text{s.t.} & D(K^1(A,A')Du^1 + K^2(A,A')Du^2 - e\gamma) + y \geq e & \\ & s^1 \geq u^1 \geq -s^1 & \\ & s^2 \geq u^2 \geq -s^2 & \\ & y \geq 0. & \end{array} \tag{18}$$

The corresponding separating surface to a solution $(u^1, u^2, \gamma, y, s^1, s^2)$ of this linear program is given by:

$$K^1(x', A')Du^1 + K^2(x', A')Du^2 = \gamma \tag{19}$$

For example we can use a linear and a discontinuous neural network kernel for $K^1(A, A')$ and $K^2(A, A')$ as follows:

$$K^1(A', A') = (AA' - ee');\ K^2(A', A') = (AA' - ee')_{\bullet *}, \tag{20}$$

where, as defined in the Introduction, the subscript $\bullet *$ denotes replacement of each component of the matrix AA' by 1, 0, or -1, depending on whether

it is positive, zero or negative, respectively. Note that in general this neural network kernel is not positive semidefinite and hence standard support vector techniques cannot make use of it. We also note that the linear programming formulation (18) can be extended to more than two kernels in an obvious manner.

5 Numerical Testing

We tested nonlinear kernels versus linear ones by using two different formulations of the generalized SVM, namely the quadratic programming formulation as given in (4), and the linear programming formulation as given in (17). Most of our code development was in the MATLAB environment [19], though the actual solutions to the SVMs were handled externally. We coded the SOR algorithm in C, and accessed it via the MATLAB "mex" interface which allows one to execute external C code as a native MATLAB function [20]. The linear programs were solved with the MINOS solver [22]. All experiments were run on the University of Wisconsin Computer Sciences Department Ironsides cluster. This cluster of four Sun Enterprise E6000 machines consists of 16 UltraSPARC II 250 MHz processors and 2 gigabytes of RAM on each node.

These SVMs were tested on the liver-disorders dataset from the University of California at Irvine (UCI) repository [21], the checkerboard dataset [13, 11, 12], and also on a synthetic dataset of Gaussian distribution. The Gaussian synthetic dataset, suggested by Usama Fayyad of Microsoft Research, was created by constructing 20 random centers, then generating "clouds" of points around these centers in a Gaussian fashion based on random covariance matrices. An arbitrary hyperplane was used to separate the centers into two classes. Each center was assigned to the class corresponding to the side of the hyperplane it fell on; furthermore, each randomly generated point was assigned to the same class as the center from which it came. We then used a scaling parameter to size the clouds so that the data was not linearly separable.

Table 1 shows results on the Gaussian dataset, using the SOR Algorithm 3.1 with both linear and quadratic kernels. The quadratic kernel SOR approach yielded over 14% improvement in tenfold testing set correctness when compared to a linear kernel.

Dataset	[Class 1],[Class -1] n	Tenfold Training Correctness Tenfold Testing Correctness	
		Linear Kernel	Quadratic Kernel
Gaussian	373, 627	83.20%	97.68%
	32	81.70%	93.41%

Table 1: **SOR training and testing set correctness for linear and quadratic kernels**

Table 2 shows training and testing set correctness, using the linear programming formulation (17) with various kernels, under tenfold cross validation for the above mentioned datasets. We implemented a linear kernel, a quadratic kernel, a symmetric sixth degree polynomial, and a symmetric sinusoidal kernel based on the following formulations which in general are indefinite kernels:

Kernel 5.1 Polynomial $((\frac{A}{\lambda} - \rho)(\frac{A}{\lambda} - \rho)' - \mu)_\bullet^d$

Kernel 5.2 Sinusoidal $(sin(\frac{A}{\lambda} - \rho)sin(\frac{A}{\lambda} - \rho)' - \mu)_\bullet^d$

. The variables λ, ρ, μ are scalar parameters and d is a positive integer. When a parameter is added or subtracted from a matrix or vector, it means componentwise addition or subtraction of the parameter in conformity with MATLAB notation. Specific values of these parameters were used in our experiments. The nonlinear kernels gave improved testing set results over linear kernels. The most striking results are those for the checkerboard dataset of Figure 1. Tenfold testing set correctness was over 97.70% for both nonlinear kernels compared with 48.60% for a linear kernel. This is reflected in the very realistic rendering in Figures 2 and 3 depicting the resulting separation of the original dataset. Our experiments were inspired by those of Kaufman in [13] where eighth degree polynomial and radial basis kernels were used in a standard support vector machine formulation to separate the checkerboard dataset. The separations depicted in Figures 2 and 3 are sharper than those of [13].

Dataset	[Class1],[Class-1] n	Tenfold Training Correctness Tenfold Testing Correctness	
		Linear Kernel	Nonlinear Kernel (Type)
Liver Disorders	145, 200 6	71.21% 68.70%	78.33% 73.37% (Quadratic)
Checkerboard	486, 514 2	51.12% 48.60%	99.97% 98.50% (6th Degree Polynomial)
Checkerboard	486, 514 2	51.12% 48.60%	98.43% 97.70% (Sinusoidal)

Table 2: **Linear programming training and testing set correctness for linear and nonlinear support vector machines**

6 Conclusion

We have implemented a simple iterative method, successive overrelaxation, and a linear programming approach for the solution of discrimination problems using generalized support vector machines with nonlinear kernels. The methods scale up and can be parallelized by using techniques already implemented [8, 9, 7, 10]. Future work includes multicategory nonlinear discrimination via kernel methods using successive overrelaxation and linear programming, as well as parallel implementation on multiprocessor machines.

Acknowledgements

We are indebted to our colleagues Usama Fayyad, Microsoft Research, for suggesting the Gaussian dataset, to Linda Kaufman, Bell Labs, for supplying us with the checkerboard dataset and to Tin Kam Ho, Bell Labs, for allowing us to make the checkerboard dataset public. This research is supported by National Science Foundation Grants CCR-9729842 and CDA-9623632, and by Air Force Office of Scientific Research Grant F49620-97-1-0326 as Mathematical Programming Technical Report 99-03, March 1999.

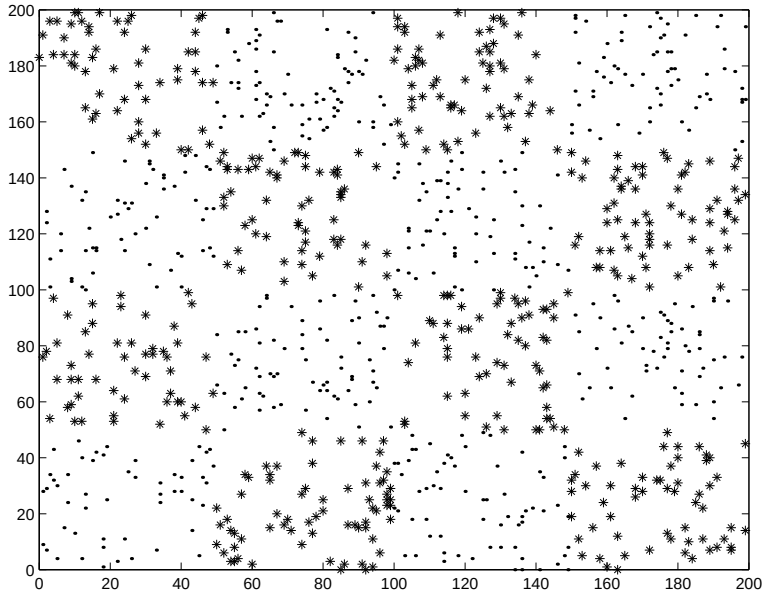

Figure 1: **Checkerboard training dataset**

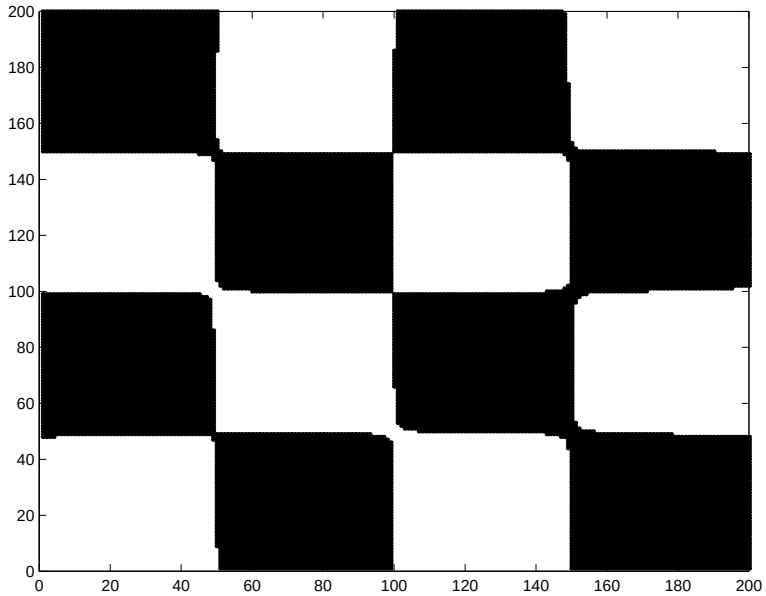

Figure 2: **Indefinite sixth degree polynomial kernel separation of the checkerboard dataset** ($\nu = 10,000,\ \lambda = 100,\ \rho = 1,\ d = 6,\ \mu = 0.5$)

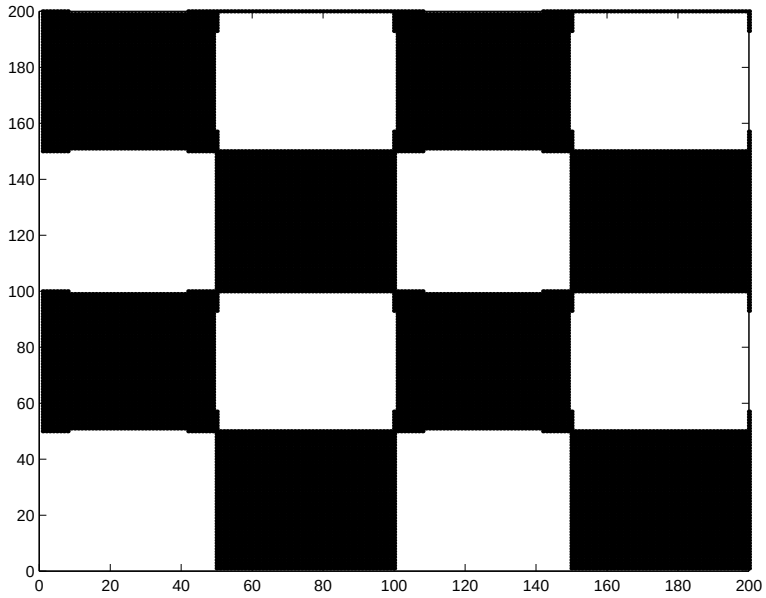

Figure 3: **Indefinite sinusoidal kernel separation of the checkerboard dataset** ($\nu = 10,000,\ \lambda = \frac{50}{\pi},\ \rho = 2\pi,\ d = 2,\ \mu = 1$)

References

[1] K. P. Bennett, D. Hui, and L. Auslender. On support vector decision trees for database marketing. Department of Mathematical Sciences Math Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY 12180, March 1998. http://www.math.rpi.edu/~bennek/.

[2] 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.Z.

[3] E. J. Bredensteiner. *Optimization Methods in Data Mining and Machine Learning*. PhD thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, 1997.

[4] E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. *Computational Optimizations and Applications*, 10:111–126, 1998.

[5] V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods.* wiley, New York, 1998.

[6] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[7] R. De Leone and O. L. Mangasarian. Serial and parallel solution of large scale linear programs by augmented Lagrangian successive overrelaxation. In A. Kurzhanski, K. Neumann, and D. Pallaschke, editors, *Optimization, Parallel Processing and Applications*, pages 103–124, Berlin, 1988. Springer–Verlag. Lecture Notes in Economics and Mathematical Systems 304.

[8] R. De Leone, O. L. Mangasarian, and T.-H. Shiau. Multi-sweep asynchronous parallel successive overrelaxation for the nonsymmetric linear complementarity problem. *Annals of Operations Research*, 22:43–54, 1990.

[9] R. De Leone and M. A. Tork Roth. Massively parallel solution of quadratic programs via successive overrelaxation. *Concurrency: Practice and Experience*, 5:623–634, 1993.

[10] M. C. Ferris and O. L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, 1(4):487–500, 1991.

[11] Tin Kam Ho and Eugene M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 880–885, Vienna, Austria, 1996. http://cm.bell-labs.com/who/tkh/pubs.html.

[12] Tin Kam Ho and Eugene M. Kleinberg. Checkerboard dataset, 1996. http://www.cs.wisc.edu/math-prog/mpml.html.

[13] L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In Bernhard Schlkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

[14] Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, 46:157–178, 1993.

[15] O. L. Mangasarian. Solution of symmetric linear complementarity problems by iterative methods. *Journal of Optimization Theory and Applications*, 22(4):465–485, August 1977.

[16] O. L. Mangasarian. On the convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem. *SIAM Journal on Optimization*, 1:114–122, 1991.

[17] O. L. Mangasarian. Generalized support vector machines. Technical Report 98-14, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, October 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.Z. Proceedings of NIPS*98 Workshop on Large Margin Classifiers, Breckenridge, Colorado, December 4-5, 1998, submitted.

[18] O. L. Mangasarian and David R. Musicant. Successive overrelaxation for support vector machines. Technical Report 98-18, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, October 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-18.ps.Z. IEEE Transactions on Neural Networks, submitted.

[19] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1992.

[20] MATLAB. *Application Program Interface Guide*. The MathWorks, Inc., Natick, MA 01760, 1997.

[21] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. Technical report, Department of Information and Computer Science, University of California, Irvine, 1992. www.ics.uci.edu/~mlearn/MLRepository.html.

[22] 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.

[23] B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.

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