Projection Support Vector Machines

Francisco J. González-Castaño
Universidad de Vigo,
Departamento de Tecnologías de las Comunicaciones,
ETSI Telecomunicación,
36200 Vigo, Spain.
javier@ait.uvigo.es
Robert R. Meyer
University of Wisconsin-Madison,
Computer Sciences Department
1210 West Dayton Street
Madison, WI 53706, USA.
rrm@cs.wisc.edu

Abstract

Large-scale classification is a very active research line in data mining. It can be applied to problems like credit card fraud detection or content-based document browsing. In recent years, several efficient algorithms for this area have been proposed by Mangasarian and Musicant. These approaches, based on quadratic problems, are: Successive OverRelaxation (SOR), Active Support Vector Machines (ASVM) and Lagrangian Support Vector Machines (LSVM). These algorithms have solved linear classification problems with millions of points. ASVM is perhaps the fastest and more scalable among them. This paper presents a projection-based SVM algorithm that outperforms ASVM on a 50,000 point data set generated by means of NDC (Normally Distributed Clusters), which has become a common tool in large-scale SVM research.

1 INTRODUCTION

A problem that appears frequently in many real applications is the Binary Classification Problem **BCP**. Given a sample of m points $x_i \in I\!R^n, i = 1, \ldots, m$, and a partition of this sample into disjoint sets S_1, S_2 . we need to construct a decision boundary; that is, a surface $\varphi(.) : I\!R^n \to I\!R$ that separates the point sets S_1, S_2. Ideally we wish to solve the problem:

$$\text{Find } \varphi(x) : I\!R^n \to I\!R, \theta \in I\!R : \left\langle \begin{array}{ll} \varphi(x) > \theta, & x_i \in S_1 \\ \varphi(x) < \theta, & x_i \in S_2. \end{array} \right. \quad (1)$$

It was shown by Makhoul et al [15] that a 2-layer neural network can reproduce a close approximation to any nonlinear decision boundary. However, the number of neurons necessary to achieve a given accuracy cannot be stated *a priori*. Moreover, the seemingly easy problem of deciding if two sets are separable by a piecewise linear function is NP-complete [20]. Research has focussed on *approximate* separating hyperplanes because of their relative simplicity. Essentially a hyperplane $w'x = \theta$ provides the surface $\varphi(x) = \theta$ in (1). It is well known (Bennett and Mangasarian [1]) that a hyperplane $w'x = \theta$ strictly separates S_1, S_2 if and only if

$$C \doteq \{w \in \mathbb{R}^n, \theta \in \mathbb{R} : \left\langle \begin{array}{ll} w'x_i - \theta \geq +1, & x_i \in S_1 \\ w'x_i - \theta \leq -1, & x_i \in S_2 \end{array} \right\} \neq \emptyset. \tag{2}$$

In the separable case, support vector training amounts to determining appropriate $\{w \in \mathbb{R}^n, \theta \in \mathbb{R}\} \in C$. Research presently centers on methods to efficiently compute an approximate separating hyperplane for the non-separable case, a contingency that frequently arises in data mining, machine learning, pattern recognition, neural network training, and surely other fields. An *ideal* hyperplane in this case should misclassify the fewest number of set elements, which is an NP-complete problem (Property attributed to Heath [12] by Mangasarian [16]). For a sufficiently small $\nu > 0$, an optimal hyperplane $w'x = \theta$ can be found by solving the following mixed integer linear programming problem:

$$\begin{array}{ll} \min\limits_{w,\theta,y} & \sum y_i \\ \text{such that} & w'x_i - \theta + \frac{1}{\nu} y_i \geq 1, x_i \in S_1 \\ & w'x_i - \theta - \frac{1}{\nu} y_i \leq -1, x_i \in S_2 \\ & y_i \in \{0, 1\}. \end{array} \tag{3}$$

However, problem (3) is intractable when the data to be classified consist of a massive number of points. Modern approaches are relaxations of (3). Essentially, the Boolean condition $y_i \in \{0,1\}$ is dropped and the model requires some desirable properties for w, θ (with no pretense of being exhaustive we refer the reader to [1, 4, 5, 6, 7, 22] and references therein).

Introducing $v = (w, \theta), e = (1, \ldots, 1)$, a suitable matrix A and dropping the Boolean constraints $y \in \{0, 1\}$, we replace (3) by:

$$\begin{array}{ll} \min\limits_{v,y} & \frac{1}{2}(v'v + y'y) \\ \text{such that} & Av + \frac{1}{\sqrt{\nu}} y \geq e \end{array} \tag{4}$$

(Note that an equivalent problem is obtained if the constraints $y \geq 0$ are added. In the projection algorithms to be discussed below, this constraint is used to "correct" the iterates).

This model has been proposed by Mangasarian and Lee [13] (previously, in [17], Mangasarian and Musicant used a linear term for y in the objective function). A salient feature of (4) is that its dual is a quadratic optimization problem subject to bounds on the variables, and several efficient algorithms have been proposed for its solution [2][8][17][18][19]. For $B = [A \frac{I}{\sqrt{\nu}}]$ and $Q = BB'$, the dual problem is:

$$\min_{u \geq 0} \quad \frac{1}{2} u'Qu - e'u, \tag{5}$$

and its KKT conditions are:

$$\begin{array}{l} Qu - e \geq 0 \\ u \geq 0 \\ u'(Qu - e) = 0 \end{array} \qquad (6)$$

Alternatively, we replace (3) by:

$$\begin{array}{rl} \min\limits_{v,y} & \frac{1}{2}(v'v + y'y + z'z) \\ \text{such that} & Av + \frac{1}{\sqrt{\nu}}y - \frac{1}{\sqrt{\mu}}z = e \\ & y, z \geq 0 \end{array} \qquad (7)$$

This model is quite different from the preceding models in the sense that there is actually a penalty for points that are well classified. The z variables are simply used so that the RHS value is matched. This allows us to work with equalities, and for a small value of μ it is expected that these types of corrections will have little effect on the objective.

The next section describes projection algorithms based on models (4) and (7). Section 3 reports remarkable numerical results on data generated via the NDC Matlab code [21]. Conclusions and directions for further research, particularly with respect to parallel implementations, are given in the last section of this paper.

We briefly comment on notation: Capital Latin letters are sets or matrices depending upon the context. Lowercase Latin letters denote vectors in $I\!R^n$, except for the range $i, \ldots, q$ that denotes integers. Lower case Greek letters are real scalars. Subindices are different components, i.e., x_i is the i-th component of the n-component vector x and $a'b$ is the inner product $\sum_{i=1}^{n} a_i b_i$. For any vector, a K subindex denotes a subvector whose components have indices belonging to the set K. $B = [A \ \frac{I}{\sqrt{\nu}} \ -\frac{I}{\sqrt{\mu}}]$ and B_i is the i-th row of matrix B. B_K is the matrix composed of all B_i such that $i \in K$. A_K is defined accordingly. Finally, we define $Q_K = B_K B_K'$, and $r_i = A_i v + \frac{1}{\sqrt{\nu}} y_i - \frac{1}{\sqrt{\mu}} z_i - 1$.

2 Projection SVM

The projection SVM (PSVM) algorithm can be formulated in terms of the determination of the least 2-norm correction of the constraints of (7) that are violated by the current iterate:

Initialization: $v = 0$, $y = 0$, $z = 0$, $s = 0$, $t = 0$.

PSVM iteration:

Step 1: Define the subindex set $K = \{i \in 1..m \mid r_i \neq 0\}$.
Step 2: $d_K = -Q_K^{-1} r_K$.

Step 3: $v \leftarrow v + A'_K d_K$, $y_K \leftarrow y_K + \frac{d_K}{\sqrt{\nu}}$, $z_K \leftarrow z_K - \frac{d_K}{\sqrt{\mu}}$.
Step 4: $\forall i \in K$, if $s_i + y_i < 0$ then $s_i \leftarrow s_i + y_i$, $y_i \leftarrow 0$. Else, $y_i \leftarrow s_i + y_i$, $s_i \leftarrow 0$.
Step 5: $\forall i \in K$, if $t_i + z_i < 0$ then $t_i \leftarrow t_i + z_i$, $z_i \leftarrow 0$. Else, $z_i \leftarrow t_i + z_i$, $t_i \leftarrow 0$.

Convergence of the PSVM algorithm: The PSVM algorithm is a special case of the Boyle-Dijkstra algorithm [3] applied to problem (7). The Boyle-Dijkstra algorithm is a 'corrected' sequential projection algorithm.

- Step 2 is the projection of the previous iterate onto $A_K v_K + \frac{1}{\sqrt{\nu}} y_K = e_K$. That is, the violated subset of the equation part in (7) (there is no need of correction in this case, since these constraints are linear equalities).
- Steps 4 and 5 introduce corrections for constraints $z > 0$ and $y > 0$, by considering increments s and t.

The algorithm converges to the projection of $v, y, z = 0$ onto the feasible region of problem (7) , which by definition minimizes the 2-norm to the feasible region.

Remark 1: In a fully primal implementation, any iterative projection algorithm could be used to solve step 2 [10, Lemma 1] (for example, a parallel projection algorithm like [11]).

Remark 2: In cases such that $m >> n$, the matrix inverse in step 2 can be efficiently computed by using the Sherman-Morrison-Woodbury identity, as proposed in [18].

Although the solution of (7) converges to the solution of (4) as μ goes to 0, it is better to avoid this scaling issue by handling the inequalities in (4) directly, even though there is the disadvantage that convergence to an optimal solution of (4) cannot be guaranteed.

For this reason, we present a simplified form of PSVM (SPSVM), which addresses the limiting case of $\mu = 0$, corresponding to the inequality constraints of (4) with the added constraints $y \geq 0$. In this case, we redefine B as $B = [A \ \frac{I}{\sqrt{\nu}}]$ and r_i as $r_i = A_i v + \frac{1}{\sqrt{\nu}} y_i - 1$.

Initialization: $v = 0$, $y = 0$.

SPSVM iteration:

Step 1: Define the subindex set $K = \{i \in 1..m \mid r_i < 0\}$.
Step 2: $d_K = -Q_K^{-1} r_K$.
Step 3: $v \leftarrow v + A'_K d_K$, $y_K \leftarrow y_K + \frac{d_K}{\sqrt{\nu}}$.

Step 4: $y_K \leftarrow (y_K)_+$.

Convergence of the SPSVM algorithm: SPSVM is a sequential block-projection algorithm [9], which alternates projections onto blocks $A_K v_K + \frac{1}{\sqrt{\nu}} y_K = e_K$ (step 2) and $y_K \geq 0$ (step 4). These blocks contain the intersection Z of $Av + \frac{1}{\sqrt{\nu}} y - e \geq 0$ and $y \geq 0$. Since the primal and dual variables satisfy $(v, y) = B'u$ at optimality, Z matches the feasible set defined by (6), with the possible exception of the complementarity constraints. The algorithm generates a Fejér sequence that converges to a point in Z. Note that the particular choice of sets used in this projection algorithm makes it quite similar to PSVM. Although the final point does not necessarily correspond to the optimal solution of problem (4), our numerical tests indicate that the result is a very good approximation to the solution of (4). Therefore, SPSVM is a heuristic, but one that is capable of producing excellent results for large-scale problems, particularly when the block-dynamic implementation discussed below is employed. From the point of view of implementation, remarks 1 and 2 above can be applied to step 2 in SPSVM.

3 Numerical results

We performed numerical tests on a Linux Pentium III machine, running at 600 MHz, with 128 MB of main memory, 256 KB L2 cache and 12 GB of secondary memory. We used the NDC Matlab code [21] to generate 50,000 points, with 32 features each ($m = 50000$, $n = 32$). The NDC expansion factor was set to 10.

ASVM (Active Support Vector Machine) and SPSVM algorithms were applied to model (4). ASVM is a high performance SVM generator for large scale problems, that yields the solution of (4) [18]. Algorithm settings were:

- $\nu = 0.1$, as in [19].
- Since ASVM stagnates if the dual variables are 0 at the beginning, we took the primal point associated to $u_0 = e$ as a starting point.
- Since $m >> n$, the Sherman-Morrison-Woodbury identity was used in ASVM and SPSVM (step 2). All auxiliary matrix inverses were calculated by means of Choleski factorizations.

We compared ASVM with a block-dynamic implementation of SPSVM. A random subset of K of size b is allowed at each iteration. The resulting relaxed SPSVM is also a block-projection algorithm, which projects each iterate onto a convex superset of Z. Therefore, the algorithm converges. However, it will stop at a point that only satisfies those constraints in (6) taken into account by the method. Numerical tests indicate that, even for small b sizes, a quite representative subset of (6) has been considered at algorithm termination.

Table 1: Performance of parametrized algorithms on the data set

type	block parameter (b)	time (s)	Best testing accuracy (a_t)
SPSVM	5000	16.84	86.63%
	2500	10.21	86.65%
	1000	6.67	86.71%
	500	4.41	86.64%
	100	1.6	86.44%
ASVM	45000	48.99	86.47%
	20000	22.19	86.42%
	10000	11.26	86.19%
	7500	8.25	86.14%
	5000	4.9	85.86%
	2500	1.67	85.86%
	1000	0.41	84.81%
	500	0.19	83.08%
	100	0.02	69.31%

Also, a block-static implementation of ASVM was tested. For a given training set, a random block of b points was selected at the beginning, and all remaining training points were discarded. It has been shown [14] that this simple strategy produces a good testing accuracy in large sets. At each iteration, ASVM works with all constraints whose dual variables are nonzero. A "brute force" block-dynamic ASVM implementation would simply pick a random sample of b constraints among them. Unfortunately, our tests suggest that such an implementation does not converge, and its development lies beyond the purpose of this paper.

For each algorithm and block size, ten runs of 100 iterations were made. Execution was stopped if there was no improvement in testing accuracy for 6 iterations. In all runs, the problem set was randomly divided in two parts, containing 90% and 10% of the points, corresponding to training and testing subsets, respectively. Table 1 shows the average of the best testing accuracies a_t reached, and the highest elapsed time among them.

Note that SPSVM reaches $a_t > 86\%$ in 1.6 seconds with a small block size, which is much faster than ASVM. Also, note that, for our particular problem, a_t increases with b for ASVM, while SPSVM achieves optimal performance with $b = 1000$.

Figures 1 and 2 show the average evolution of testing set accuracy across ten runs for the best instances of ASVM and SPSVM.

4 Final remarks

In this paper, we have proposed two new algorithms for SVM generation, PSVM and SPSVM. A block-dynamic implementation of SPSVM out-performs ASVM

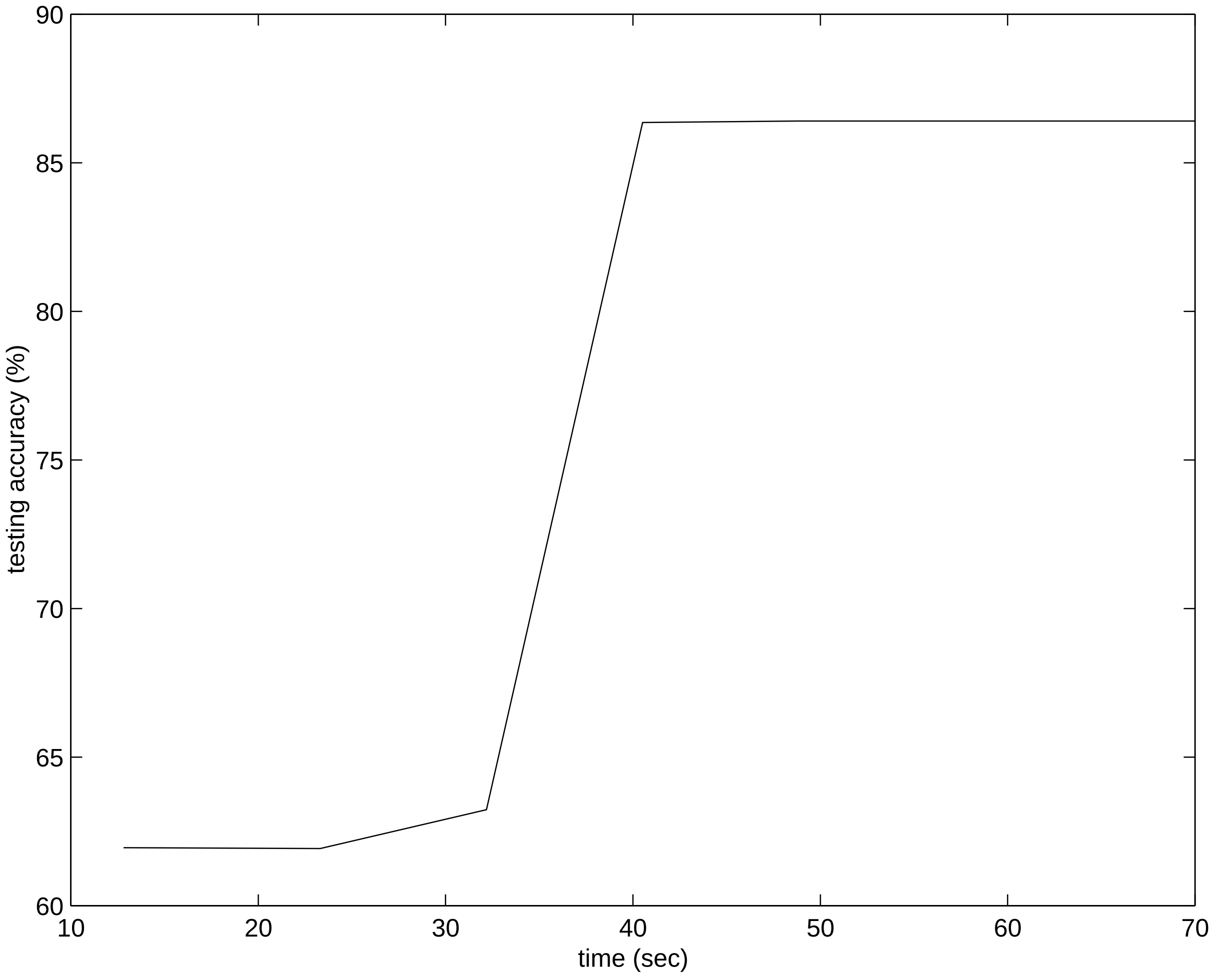

Figure 1: ASVM, block size 45000

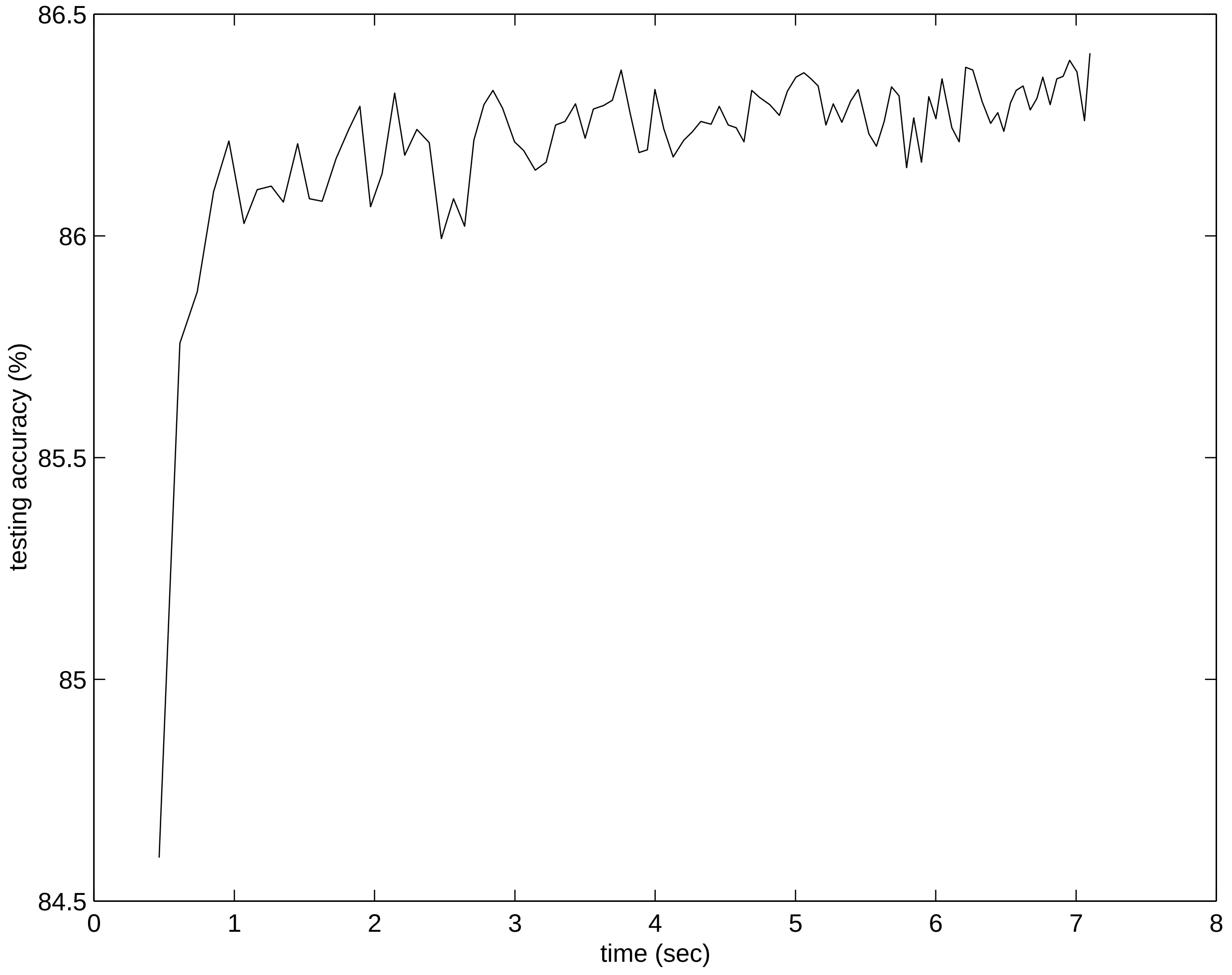

Figure 2: SPSVM, block size 1000

[18], on a medium-scale instance of the NDC problem.

As a future research line, we plan to introduce parallelism in the projection SVM algorithms. Specifically, two approaches can be followed:

- Parallelization of step 2, by means of parallel projection methods [11].
- A parallel block-dynamic approach: each processor solves step 2 on a different constraint block, and the best point is selected. The objective function (5) can be used as a merit function.

Another future research line concerns scalability issues. In harder problems (for example, NDC problems with a larger expansion factor), it is expected that static-block ASVM implementations will have a worse performance, since they will work with a much less representative sample, as b decreases. However, a block-dynamic SPSVM implementation may still sweep a broad portion of the problem, for enough execution time.

The authors are indebted to Prof. Ubaldo García-Palomares (Universidad Simón Bolívar, Venezuela), for interesting observations that greatly influenced this work.

References

[1] K.P. Bennett and O.L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, *Optimization Methods and Software* **1** (1992) 23–34.

[2] R.H. Bielschowsky, A. Friedlander, F.A.M. Gomes, J.M. Martínez and M. Raydán, An adaptive algorithm for bound constrained quadratic minimization, *Investigación Operativa* **7** (1997) 67–102.

[3] J.P. Boyle and R.L. Dijkstra, A method for finding projections onto the intersections of convex sets in Hilbert spaces, *Lecture Notes in Statistics* **37** (1986) 28–47.

[4] P.S. Bradley and O.L. Mangasarian, Massive data discrimination via linear support vector machines, *Optimization Methods and Software* **13** (2000) 1–10.

[5] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* **2-2** (1998) 121–167.

[6] V. Cherkassky and F. Mullier, *Learning from Data - Concepts, Theory and Methods* John Wiley & Sons, New York (1998).

[7] N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines, *Cambridge University Press* (2000)

[8] M.A. Diniz-Ehrhardt, Z. Dostal, M.A. Gomes-Ruggiero, J.M. Martínez and S. A. Santos, Nonmonotone strategy for minimization of quadratics with simple constraints, to appear in *Applications of Mathematics*.

[9] U.M. García-Palomares, Aplicación de los métodos de proyección en el problema de factibilidad convexa: un repaso, *Investigación Operativa* **4-3** (1994) 229–245.

[10] U.M. García-Palomares, Preconditioning projection methods for solving algebraic linear systems, *Numerical Algorithms* **21** (1999) 157–164.

[11] U.M. García-Palomares and F.J. González Castaño, Incomplete projections algorithms for solving the convex feasibility problem, *Numerical Algorithms* **18** (1998) 177–193.

[12] D. Heath, A geometric framework for machine learning, *PhD Thesis, Department of Computer Science, Johns Hopkins University, USA* (1992).

[13] Y.-J. Lee and O.L. Mangasarian, SSVM: A smooth support vector machine, *Computational Optimization and Applications*, to appear.

[14] 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, USA (July 2000).

[15] J. Makhoul, El-Jaroudi and R. Schwartz, Formation of disconnected decision regions with a single hidden layer, in *Proceedings of the International Joint Conference on Neural Networks* **1** (1989) 455–460.

[16] O.L. Mangasarian, Optimization in machine learning, Mathematical programming technical report 95-01, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, USA (January 1995).

[17] O.L. Mangasarian and D.R. Musicant, Successive overrelaxation for support vector machines, *IEEE Transactions on Neural Networks* **10** (1999) 1032–1037.

[18] O.L. Mangasarian and D.R. Musicant, Active support vector machine classification, Technical report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, USA (April 2000).

[19] O.L. Mangasarian and D.R. Musicant, Lagrangian support vector machines, Technical report 00-06, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, USA (June 2000).

[20] N. Megiddo, On the complexity of polyhedral separability, *Discrete and Computational Geometry* **3** (1988) 325–337.

[21] D.R. Musicant, NDC: normally distributed clustered datasets, *www.cs.wisc.edu/ musicant/data/ndc/* (2000).

[22] V.N. Vapnik, *The nature of statistical learning theory* Springer, New York (1995).