Massive Data Discrimination via Linear Support Vector Machines

P. S. Bradley
Microsoft Research
One Microsoft Way
Redmond, WA 98052

bradley@microsoft.com

O. L. Mangasarian
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706

olvi@cs.wisc.edu

March 31, 1999

Abstract

A linear support vector machine formulation is used to generate a fast, finitely-terminating linear-programming algorithm for discriminating between two massive sets in n-dimensional space, where the number of points can be orders of magnitude larger than n. The algorithm creates a succession of sufficiently small linear programs that separate chunks of the data at a time. The key idea is that a small number of support vectors, corresponding to linear programming constraints with positive dual variables, are carried over between the successive small linear programs, each of which containing a chunk of the data. We prove that this procedure is monotonic and terminates in a finite number of steps at an exact solution that leads to an optimal separating plane for the entire dataset. Numerical results on fully dense publicly available datasets, numbering 20,000 to 1 million points in 32-dimensional space, confirm the theoretical results and demonstrate the ability to handle very large problems.

Keywords: Support vector machines, linear programming chunking

1 Introduction

We consider a linear programming approach [4, 1] to support vector machines (SVMs) [21, 22, 7] for the discrimination between two possibly massive datasets. The principal contribution of this work is a novel method for solving linear programs with extremely large number of constraints that is proven to be monotonic and finite. In the standard SVM formulation [19, 18, 20] very large quadratic programs are solved. In contrast, the formulation here consists of solving a linear program which is considerably less difficult. For simplicity, our results are given here for a linear discriminating surface, a separating plane. However, extension to nonlinear surfaces such as quadratic [11] or more complex [5], is straightforward and merely requires transforming the input data by some nonlinear transformation into a possibly higher dimensional space in which a separating plane can again be found using linear programming.

In Section 2 the linear support vector machine (LSVM) is formulated. In Section 3 we state our linear programming chunking (LPC) algorithm and establish its monotonicity and finite termination. In Section 4 we test the LPC on publicly available data, with datasets numbering 20,000 to 1 million points, all in 32-dimensional space. Section 5 concludes the paper.

2 LSVM: The Linear Support Vector Machine

Given two point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n represented by the matrices $A \in R^{m\times n}$ and $B \in R^{k\times n}$ respectively, we wish to discriminate between them by a separating plane:

$$P = \{x \mid x \in R^n, x^T w = \gamma\}, \tag{1}$$

with normal $w \in R^n$ and 1-norm distance to the origin of $\dfrac{|\gamma|}{\|w\|_\infty}$ [13], where $\|\cdot\|_\infty$ denotes the ∞-norm. Hence we wish to satisfy, in some best sense, the inequalities:

$$Aw \geq e\gamma + e,\ Bw \leq e\gamma - e, \tag{2}$$

where e is a vector of ones of appropriate dimension. Conditions (2), which place $\mathcal{A}$ in the closed halfspace $\{x \mid x^T w \geq \gamma + 1\}$ and $\mathcal{B}$ in the closed halfspace $\{x \mid x^T w \leq \gamma - 1\}$, can be satisfied if and only if the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ are disjoint. This is not the case in many real-world applications. Hence, we attempt to satisfy (2) in some best sense, for example, by minimizing the average infeasibilities of (2) by introducing nonnegative slack variables $y \in R^m$ and $z \in R^k$ and solving the following robust linear programming (RLP) formulation originally proposed in [3] and effectively used on medical and other databases [14]:

$$\min_{w,\gamma,y,z} \left\{ \frac{e^T y}{m} + \frac{e^T z}{k} \,\middle|\, -Aw + e\gamma + e \leq y,\ Bw - e\gamma + e \leq z,\ y \geq 0,\ z \geq 0 \right\}. \tag{3}$$

The linear program (3) generates a separating plane P that approximately satisfies the conditions (2) in the following sense. Each positive value of y_i determines the distance $\frac{y_i}{\|w\|'}$ [13, Theorem 2.2] between a point A_i of $\mathcal{A}$ lying on the wrong side of the bounding plane $x^T w = \gamma + 1$ for $\mathcal{A}$, that is A_i lying in the open halfspace $\{x \mid x^T w < \gamma + 1\}$, and the bounding plane itself. Here $\|\cdot\|'$ denotes the dual norm to the norm $\|\cdot\|$ used in measuring the distance between the planes. Similarly $\frac{z_i}{\|w\|'}$ is the distance between a misclassified point B_i of $\mathcal{B}$ and the bounding plane $x^T w = \gamma - 1$ for the set $\mathcal{B}$. Thus the objective function of the linear program (3) minimizes the average sum of distances, weighted by $\|w\|'$, of misclassified points to the bounding planes. The separating plane P (1) is midway between the two bounding planes and parallel to them. The LSVM consists of parametricly adding another term, $\frac{\|w\|'}{2}$, to the objective function of the RLP (3). The justification for this term is as follows. The separating plane P (1) generated by the RLP linear program (3) lies midway between the two parallel planes $w^T x = \gamma + 1$ and $w^T x = \gamma - 1$. The distance, measured by some norm $\|\cdot\|$ on R^n, between these planes is precisely $\frac{2}{\|w\|'}$ [13, Theorem 2.2]. The appended term to the objective function of the RLP (3), $\frac{\|w\|'}{2}$, is the reciprocal of this distance, which has the effect of maximizing the distance between these two bounding planes to obtain better separation. If we use the ∞-norm to measure the distance between the planes, then the dual to this norm is the 1-norm and accordingly $\|w\|' = \|w\|_1$. Thus, appending $\lambda\frac{\|w\|_1}{2}$ to $(1-\lambda)$ times the objective function of (3), with $\lambda \in [0, 1)$, leads to the following linear programming formulation of our LSVM where $e^T s = \|w\|_1$ at optimality:

$$\min_{w,\gamma,y,z,s} \left\{ (1-\lambda)(\frac{e^T y}{m} + \frac{e^T z}{k}) + \frac{\lambda}{2} e^T s \;\middle|\; \begin{array}{rcl} -Aw + e\gamma + e & \leq & y, \\ Bw - e\gamma + e & \leq & z, \\ -s \leq w & \leq & s, \\ y \geq 0, z & \geq & 0 \end{array} \right\}. \tag{4}$$

Points $A_i \in \mathcal{A}$ and $B_i \in \mathcal{B}$ appearing in active constraints of the linear program (4) with positive dual variables constitute the *support vectors* of the problem. For computing the optimal separating plane, these are the *only* relevant data points. Their number is usually small and it is proportional to the generalization error of the classifier [19]. We note that the support vector machine problem is usually formulated using the 2-norm squared in the objective [21, 2, 19, 18, 20] thus leading to a quadratic program instead of our linear program (4), which is considerably more difficult to solve than our linear program (4).

3 LPC: Linear Programming Chunking

We consider a general linear program

$$\min_x \left\{ c^T x \,|\, Hx \geq b \right\}, \tag{5}$$

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 (5), 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 [8], [6, pp 198-200,428-429] and others [17, pp 243-248], but it differs from active set methods [10, 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 (5) *Let $[H \;\; b]$ 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 (5) and all subproblems (6) below, have vertex solutions. At iteration $j = 1, \ldots$ compute x^j by solving the following linear program:

$$x^j \in arg\ vertex\ min \left\{ c^T x \;\middle|\; \begin{array}{lll} H^{(j\ mod\ \ell)}\, x & \geq & b^{(j\ mod\ \ell)} \\ \bar{H}^{(j\ mod\ \ell)-1}\, x & \geq & \bar{b}^{(j\ mod\ \ell)-1} \end{array} \right\}, \tag{6}$$

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 (6) satisfied as equalities by x^j) with positive optimal Lagrange multipliers at iteration j. Stop when $c^T x^j = c^T x^{j+\nu}$ for some input integer ν. Typically $\nu = 4$.

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 $\{c^T x^j\}$ of objective function values is nondecreasing and is bounded above by the minimum of $\min\limits_x \{c^T x \,|\, Hx \geq b\}$.

(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 (6) 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 (5) provided all active constraints at x^j have positive multipliers for $j \geq \bar{j}$.

This theorem is significant not only for the support vector application presented here, but also as a fundamental computational approach for handling linear programs with massive constraints for which the subproblems (6) of the LPC Algorithm 3.1 have vertex solutions. To establish its validity we first prove a lemma.

Lemma 3.3 *If $\bar{x}$ solves the linear program (5) and $(\bar{x}, \bar{u}) \in R^{n+m}$ is a Karush-Kuhn-Tucker (KKT) [12] point (i.e. a primal-dual optimal pair) such that $\bar{u}_I > 0$ where $I \subset \{1, \ldots, m\}$ and $\bar{u}_J = 0$, $J \subset \{1, \ldots, m\}$, $I \cup J = \{1, \ldots, m\}$, then*

$$\bar{x} \in arg\min\{c^T x \,|\, H_I x \geq b_I\} \tag{7}$$

where H_I consists of rows H_i, $i \in I$ of H, and b_I consists of elements b_i, $i \in I$.

Proof The KKT conditions [12] for (5) satisfied by $(\bar{x}, \bar{u})$ are:

$$c = H^T \bar{u},\ \bar{u} \geq 0,\ \bar{u}^T (H\bar{x} - b) = 0,\ H\bar{x} - b \geq 0,$$

which under the condition $\bar{u}_I > 0$ imply that

$$H_I\bar{x} = b_I,\ \bar{u}_J = 0,\ H_J\bar{x} \geq b_J.$$

We claim now that $\bar{x}$ is also a solution of (7), because $(\bar{x}, \bar{u}_I)$ satisfy the KKT sufficient optimality conditions for (7):

$$c = {H_I}^T\bar{u}_I,\ \bar{u}_I > 0,\ H_I\bar{x} = b_I. \diamond$$

Proof of Theorem 3.2

(i) By Lemma 3.3, c^Tx^j is a lower bound for c^Tx^{j+1}. Hence the sequence $\{c^Tx^j\}$ is nondecreasing. Since the the constraints of (6) form a subset of the constraints of (5), it follows that $c^Tx^j \leq \min_x \{c^Tx \,|\, Hx \geq b\}$.

(ii) Since there are a finite number of (feasible and infeasible) vertices to the original linear program (5) as well as of the subproblems (6), it follows that from a certain $\bar{j}$ onward, a finite subset of these vertices will repeat infinitely often. Since a repeated vertex gives the same value for c^Tx, it follows, by the nondecreasing property of $\{c^Tx^j\}$ just established, that all vertices between repeated vertices also have the same objective value c^Tx and hence: $c^Tx^j = c^Tx^{j+1} \leq \min_x \{c^Tx \,|\, Hx \geq b\}$, $\forall j \geq \bar{j}$.

(iii) Let $\bar{j}$ be as defined in the theorem and let the index $t \in \{1, \ldots, m\}$ be that of some active constraint at iteration $\bar{j}$ with positive multiplier, that is: $H_tx^{\bar{j}} = b_t$, $u_t^{\bar{j}} > 0$, that has become inactive at the next step, that is: $H_tx^{\bar{j}+1} > b_t$. We then obtain the following contradiction by part (ii) above and the KKT saddlepoint condition:

$$0 = c^Tx^{\bar{j}+1} - c^Tx^{\bar{j}} \geq {u^{\bar{j}}}^T(\bar{H}^{\bar{j}}x^{\bar{j}+1} - \bar{b}^{\bar{j}}) \geq u_t^{\bar{j}}(H_tx^{\bar{j}+1} - b_t) > 0.$$

(iv) By (ii) a finite number of vertices repeat infinitely for $j \geq \bar{j}$ all with constant c^Tx^j. Since active constraints with positive multipliers at iteration j remain active at iteration $j+1$ by (iii) and hence have a positive multiplier by assumption of part (iv), the set of active constraints with positive multipliers will remain constant for $j \geq \tilde{j}$, for some $\tilde{j} \geq \bar{j}$ (because there are a finite number of constraints) and hence x^j will remain a fixed vertex $\bar{x}$ for $j \geq \tilde{j}$. The point $\bar{x}$ will satisfy all the constraints of problem (5) because all constraints are eventually imposed on the infinitely repeated vertex $\bar{x}$. Hence $c^T\bar{x}$ which lower-bounds the minimum of (5) is also a minimum of (5) because $\bar{x}$ is feasible. Hence the algorithm can be terminated at $j = \tilde{j}$. $\diamond$

Remark 3.4 *We have not excluded the possibility (never observed computationally) that the objective function remains constant over a finite number of iterations then increases. Theorem 3.2 asserts that eventually the objective function will be constant and equal to the minimum for all iterates $j \geq \bar{j}$. Of course, one can check the satisfaction of the KKT conditions for optimality, although this does not seem to be needed in practice.*

Remark 3.5 *In order to handle degenerate linear programs, i.e. those with active constraints with zero multipliers at a solution, we have modified the computational implementation of the LPC Algorithm 3.1 slightly by admitting into* $[\bar{H}^j \quad \bar{b}^j]$ *all active constraints at iteration* j *even if they have zero multipliers. We note that the assumption of part (iv) of Theorem 3.2 is required to prove that* x^j *is a solution of (5) for all* $j > \bar{j}$.

4 Computational Tests

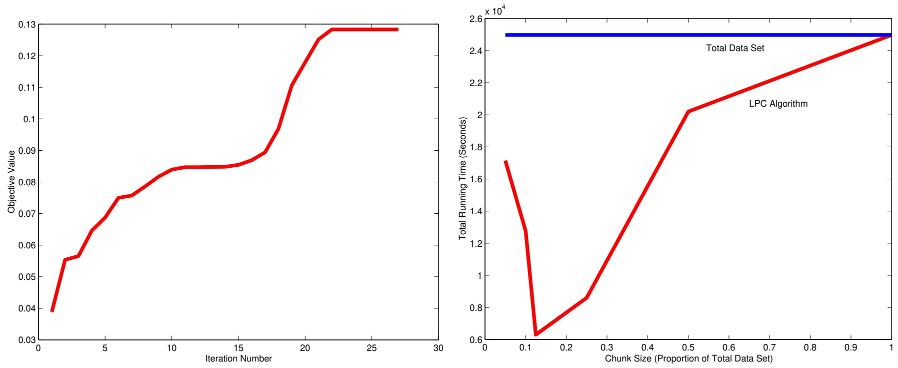
Data sets used to evaluate the LPC Algorithm 3.1 were generated using the publicly available Synthetic Classification Data Set Generator (SCDS)[15]. Fully dense data sets $\mathcal{A}$ and $\mathcal{B}$ of various size were created with 32 attributes, 4 of which are relevant, with no missing attribute values and a 10% ratio of misclassified points to simulate noisy data. We used a small version of this dataset with 20,000 points for prototyping and obtaining accurate running times on a Sun SparcStation 20 with 98 megabytes of RAM. For larger problems with 200,000 points and more we used the Computer Sciences Department Ironsides cluster of 4 Sun Enterprise E6000 machines, each with 16 UltraSPARC II processors for a total of 64 processors and 8 gigabytes of RAM. All linear programs were solved with MINOS [16] called from a C implementation of the LPC algorithm. The parameter λ of (4) was set to 0.05 in all runs.

Figures (a)-(b) show results for **200,000 points in** R^{32}. Figure (a) depicts monotonic values of the minima of subproblems (6) at each iteration of the LPC algorithm converging to the optimum of (5). Figure (b) depicts total running time of the LPC algorithm until a constant objective value is obtained. Note that the LPC algorithm reduced running time to 1.75 hours, a 74.8% reduction from the 6.94 hours for a full dataset linear program, when it used 12.5% of the data at a time. In fact for all chunk sizes less than 100% the LPC reduced running time.

For the problem with **500,000 points in** R^{32} the LPC algorithm with 12.5% chunking of the full dataset obtained a separating plane in 25.91 hours in 8 iterations. The objective function remained constant for the next $\nu = 4$ iterations.

For the problem with **1 million points in** R^{32} the LPC algorithm with 2% chunking of the full dataset obtained a separating plane in 231.32 hours in 63 iterations. The objective function remained constant for the next $\nu = 4$ iterations.

Remark 4.1 *There exists a chunk size which is empirically best in the sense that it balances the computational overhead of constructing the LPC subproblems (6) with the computational overhead of solving the LPC subproblems. For small chunk sizes, the computational burden is dominated by subproblem construction since there are many and solving them is not expensive. For large chunk sizes, the computational burden is dominated by solving large subproblem LPs. Figure (b) indicates that the chunk size of 12.5% is best for this dataset of 200,000 points in* R^{32}.



(a) Objective function values versus iteration number for chunk size of 5% of original dataset of 200,000 points in R^{32}. Objective function increases until iteration 22 and is flat thereafter.

(b) Total running time (seconds) versus chunk size for iterative LPC algorithm for dataset of 200,000 points in R^{32}. LPC time reduction of 25.2% at 12.5% chunk size.

5 Conclusion

We have proposed and implemented a linear programming chunking algorithm for discriminating between massive datasets. The algorithm, significant in its own right as a linear programming decomposition algorithm, is very effective for discrimination problems with large datasets that may not fit in machine memory and for problems taking excessive time to process. The algorithm uses support vector ideas by keeping only essential data points needed for determining a separating plane. The algorithm can handle extremely large datasets because it deals with small chunks of the data at a time and is guaranteed to terminate in a finite number of steps. Although we have not discussed parallelization here, this can be easily implemented by splitting the data among processors and sharing only support vectors among them. This would allow us to handle problems with extremely large datasets on a network of workstations or PCs.

Acknowledgements

This research is supported by National Science Foundation Grants CCR-9322479, CCR-9729842 and CDA-9623632, and by Air Force Office of Scientific Research Grant F49620-97-1-0326 as Mathematical Programming Technical Report 98-05, May 1998.

References

[1] K. P. Bennett. (1998). Combining support vector and mathematical programming methods for induction. Unpublished manuscript, Department of Mathe-

matical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180.

[2] K. P. Bennett and J. A. Blue. (1997). A support vector machine approach to decision trees. In *Proceedings of IJCNN'98*, pages 2396–2401, Anchorage, Alaska. http://www.math.rpi.edu/~bennek/.

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

[4] P. S. Bradley and O. L. Mangasarian. (1998). 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. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

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

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

[7] C. Cortes and V. Vapnik. (1995). Support vector networks. *Machine Learning*, 20:273–279.

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

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

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

[11] O. L. Mangasarian. (1965). Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452.

[12] O. L. Mangasarian. (1994). *Nonlinear Programming.* SIAM Classic in Applied Mathematics 10, Philadelphia.

[13] O. L. Mangasarian. (1999). Arbitrary-norm separating plane. *Operations Research Letters* 23, to appear. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07.ps.Z.

[14] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.

[15] G. Melli. (1997). Synthetic classification data sets (scds). http://fas.sfu.ca/cs/people/GradStudents/melli/SCDS/.

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

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

[18] E. Osuna, R. Freund, and F. Girosi. (1997). Improved training algorithm for support vector machines. In *Proceedings of IEEE NNSP'97, Amelia Island, FL, September 1997, 276-285.* http://www.ai.mit.edu/people/girosi/home-page/svm.html.

[19] E. Osuna, R. Freund, and F. Girosi. (1997). Training support vector machines: An application to face detection. In *IEEE Confernece on Computer Vision and Pattern Recognition, Puerto Rico, June 1997, 130-136.* http://www.ai.mit.edu/people/girosi/home-page/svm.html.

[20] J. Platt. (1999). Sequential minimal optimization: A fast algorithm for training support vector machines. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press. http://www.research.microsoft.com/∼jplatt/smo.html.

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

[22] G. Wahba. (1999). Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69–88, Cambridge, MA. MIT Press. ftp://ftp.stat.wisc.edu/pub/wahba/index.html.