Les Cahiers du GERAD ISSN: 0711–2440

Dynamic Point Selection for the L_1 Norm Hyperplane Separation Problem

S. Blanchard, G. Caporossi,
P. Hansen

G–2005–59

August 2005

Dynamic Point Selection for the L_1 Norm Hyperplane Separation Problem

Simon Blanchard
Gilles Caporossi
Pierre Hansen

GERAD and
Service de l'enseignement des méthodes quantitatives de gestion
HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada, H3T 2A7
{simon.blanchard;gilles.caporossi;pierre.hansen}@gerad.ca

August 2005

Les Cahiers du GERAD
G–2005–59

Abstract

L_1 norm discrimination consists in finding the hyperplane that minimizes the sum of L_1 norm distances between the hyperplane and the points that lie on the wrong side of the hyperplane. This problem is difficult for datasets containing more than 100,000 points. Since few points are needed to obtain the optimal hyperplane, we propose a point selection algorithm which iteratively adds the necessary points to a reduce set of points. We evaluate different point selection criteria. The resulting method obtains in reasonable times exact solutions for problems of up to two million points.

Résumé

Le problème de discrimination en norme L_1 consiste à trouver l'hyperplan qui minimise la somme des distances en norme L_1 entre l'hyperplan et les points qui se trouvent du mauvais côté de celui-ci. Ce problème est difficile à résoudre exactement pour des ensembles contenant plus de 100 000 points à discriminer. Puisque peu de points sont essentiels afin d'obtenir l'hyperplan optimal, nous proposons une méthode de sélection de points qui ajoutera séquentiellement les points nécessaires à un ensemble réduit de points. Nous évaluons différents critères de choix des points. La méthode résultante permet de résoudre exactement dans des temps raisonnables des problèmes qui ont jusqu'à deux millions de points.

1 Introduction

Given two sets of observations in n dimensional space, the hyperplane separation problem consists in finding an hyperplane that best separates the two sets according to arbitrary criteria. Such criteria are typically variations of either minimizing the number of misclassified points or minimizing the distance between the misclassified points and the hyperplane.

Because the problem is NP-complete for all distance norms except L_1 [13], researchers (e.g. [1, 15, 20]) have developed special purpose algorithms and heuristics that can tackle problems of large size. Other researchers focused on finding alternative methods for finding a separating hyperplane that can cope with the increasing size of datasets. Robust linear programming [3] and support vector machines [21] are good examples of popular formulations that, even though quite intuitive, address a different problem than the ones mentioned above.

In this paper, we propose a fast general iterative scheme for the two group hyperplane separation problem that finds the optimal solution. We demonstrate the efficiency of the method by applying it to the problem of minimizing the sum of the L_1 norm distances between the misclassified points and the hyperplane (MSD).

In the next section, we present the L_1 norm MSD problem. Then, we introduce the dynamic point selection algorithm. Finally, we evaluate the gains obtained by using the proposed approach.

2 Problem formulation

Let A and B be sets of observations in $\Re^n$ where $A \in \Re^{m_A \ x \ n}$ and $B \in \Re^{m_B \ x \ n}$. We want to obtain an hyperplane $h\ (w, \gamma)$ so that the distance of the misclassified points to the hyperplane is minimum. w is the one by n normal vector of the plane and γ is a scalar. Traditionally, the basic problem has been formulated as follows [12]:

$$\min \quad \sum_{i \in A} y_i + \sum_{j \in B} z_j \tag{1}$$

$$\text{subject to} \quad -x^i w + \gamma \leq y_i \quad i \in A \tag{2}$$

$$x^j w - \gamma \leq z_j \quad j \in B \tag{3}$$

$$y_i, z_j \in \mathbb{R}^+,\ w \in \mathbb{R}^n,\ \gamma \in \mathbb{R} \tag{4}$$

where x^i and x^j are the coordinates of observations from A and B respectively. y_i and z_j are variables that are greater than zero if the associated observation is on the wrong side of the plane and zero otherwise. Several other formulation have been proposed (e.g. [2, 7, 8]), but this one, as well as the alternatives ones, were proven to be inexact [6, 16] as they did not prevent the null solution. Later formulations added various normalization constraints (e.g. [5, 10, 11]), however the normalization constraints added were modifying the problem; they either rules out some admissible solutions or they didn't obtain the same solution for

rotated data or both [3]. This is because they did not take into account that the admissible region of the problem is a unit sphere in a norm dual to the one used for measuring the distances between the misclassified points and the hyperplane [13].

When measuring distances using the L_1 norm, one must add the following constraints to the problem (1)-(4):

$$-1 \leq w_l \leq 1 \quad \forall l = 1...n \tag{5}$$

$$\exists q \ / \ |w_q| = 1 \quad q \in (1...n). \tag{6}$$

If taking constraint (5) into account is easy, constraint (6) cannot be handled directly without using binary variables. This constraint says that at least one of the w variable must be equal to 1 or -1. In 2 dimensional space, the admissible solutions for w are illustrated in Figure 1 as a bold line while the traditional solution sphere for the L_2 norm (Euclidian distance) is shown as a dashed line. The $2n$ cases it implies ($w_1 = -1, w_1 = 1, w_2 = -1, w_2 = 1..., w_n = -1, w_n = 1$) could be considered separately as shown in [13]. This implies solving the $2n$ corresponding linear programs one after the other and keeping the best solution h^* as optimal solution.

3 A dynamic point selection algorithm

Very few constraints are active in the final simplex basis. Active constraints are those that are satisfied to equality; they either correspond to points that are misclassified or on the separating hyperplane. Contrarily, if a point respects the following *condition O* for one of the $2n$ linear programs it may be omitted for that LP.

Condition O. *A point may be omitted for the resolution of one of the* $2n$ *problems if adding it would not change its solution. A sufficient condition is that is does not lie on the hyperplane obtained and is not misclassified.*

Based upon this property, our algorithm solves the problem using a subset of the points and iteratively adds points that do not respect the *condition O*. If no point needs to be

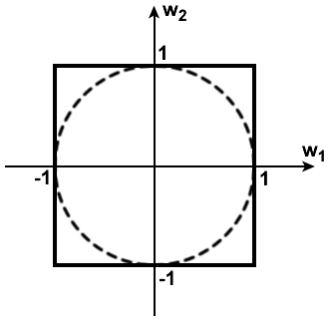

Figure 1: Admissible regions in R^2, for problems using L_1 and L_2

added, the optimal hyperplane h^* is found. To describe the algorithm, we define S as the working set, i.e. the set of points that is used to construct the current linear program. The omitted points are noted $\overline{S}$.

Solving the problem with S yields a separating hyperplane $h(w, \gamma)$. We note H the set of points that are misclassified or on $h(w, \gamma)$. The set E of omitted misclassified points is defined by $E = H \cup \overline{S}$. The algorithm is described in Figure 2.

As no point is ever removed from S, the convergence proof of the algorithm is obvious. The rate of convergence however, depends on the choice of the points to be in the initial set S.

The convergence property is not affected by the choice of the points added as long as some points are added to S while E is not empty. It is therefore theoretically not necessary to add all misclassified points and one may also add points that are correctly classified. However, our experiments using various strategies at *Step 4* lead us to the simple rule described above.

3.1 Creating the initial working set

To prove the optimality of a solution $h^*(w^*, \gamma^*)$, it is necessary that all points not satisfying the *condition O* for this solution are in the working set. In the support vector machine terminology, this set of points (vectors) is dubbed as support vectors. To reduce the number of iterations (and the computing time), a good initial working set should at least contain all the support vectors for the optimal hyperplane $h^*(w^*, \gamma^*)$.

For the MSD problem, points that do not respect *condition O* are associated with active constraints. As $2n$ linea programs must be considered simultaneously, even if in linear programming constraints that are inactive at the optimal solution can be removed from the problem without changing the solution, some points that respect the *condition O* for the optimal hyperplane must be considered. In fact, if we were to include only the misclassified points and the points that lie on the surface of the optimal hyperplane, we would obtain what we call a "reversed" hyperplane (see Figure 3) with optimal value 0 (for the restricted set of points). If only points that do not respect the *condition O* were used as initial working set, almost if not all the omitted points would be added at the next

Step 1. Create initial working set S	
Do {	
Step 2. Find	h using S
Step 3. Set	$E = \overline{S} \cap H$
Step 4. Add	$S \leftarrow S \cup E$
} while $(E \neq \oslash)$	

Figure 2: The dynamic point selection algorithm

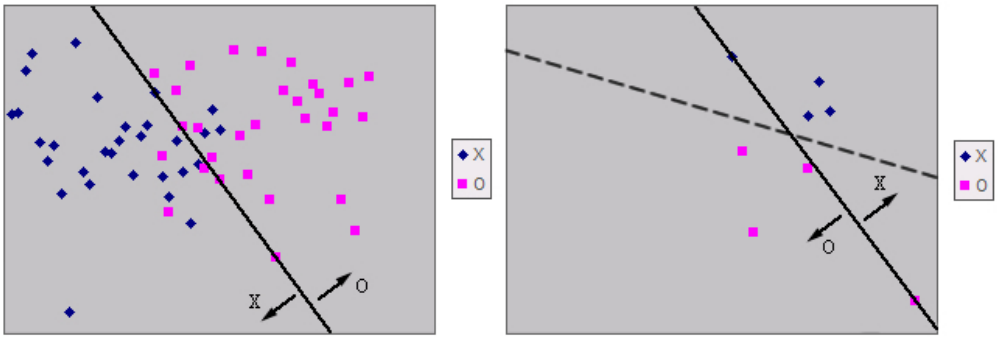

Figure 3: Example of a complete set (left), a working set (right) with a reversed optimal hyperplane.

iteration. In this case, another of the $2n$ problems to be considered will happen to provide a better solution (on the working set) than the one from which h^* was obtained.

For this reason, and to reduce the number of iterations, even if the problem is initialized with a good (or even optimal) hyperplane, we must also consider the addition of well classified points. In our experiments, we started with a heuristically obtained hyperplane and initialized the working set with all the points not satisfying *condition O* as well as a sufficient number of points that are close to the hyperplane. In practice, we added well classified points until the sum of their distances to the hyperplane was at least as big as the sum of the distances of the included misclassified points to the hyperplane.

Additionally, even though they are inactive with the heuristic solution, points that are well classified need to be added to prevent the reversal of the hyperplane. The preferred way of selecting the points is to sort them by proximity to the heuristic hyperplane and then select the closest ones until the sum of their distances to the hyperplane definitely surpasses that of the misclassified ones. Doing so prevents the reversal of the hyperplane. This is subsequently referred to as PLANCG.

If no initial hyperplane is available, it is possible to create the working set with a random sample selection from the complete set. The method, in this case, is similar to chunking as expressed in [19]. This method is hereafter referred to as RANDCG. Our experiments indicate that including twice as many points as there should be with the optimal solution brings the best trade-off between preventing the reversal of the hyperplane and including the fewest points possible. This percentage can be roughly estimated using prior knowledge of the problem or by using an heuristic.

3.2 Using an upper-bound to improve speed

Let $h(w, \gamma)$ be the hyperplane obtained from S and $\overline{d}$ be the corresponding sum of distances from all misclassified points (from S and $\overline{S}$) and h, then T is an upper bound on the optimal solution value $d*$.

Some of the $2n$ linear programs yield bad solutions because of the dimension restrictions on their hyperplanes. Since adding points to S cannot decrease the objective value, any of the $2n$ linear programs whose solution on S is worse than $\overline{d}$ cannot improve the current solution $h(w, \gamma)$ with value $\overline{d}$, and this linear program needs not to be optimized any further. This leads to substantial savings in optimization times.

This can further be improved by sorting the order in which the $2n$ problems are solved using a non-optimal hyperplane, according to importance of absolute value w_l. This is based on the assumption that even after adding new points to the working sets, we have a greater chance of finding the optimal solution of that iteration on one of the two admissible surfaces of the hyperplane that have the most importance when calculating the distance between the points and the hyperplane.

3.3 Similarities with other algorithms

Chunking related methods [4, 18, 19, 22] take advantage from the fact that removing observations that are well classified will not change anything to the objective function value. Initially, they were proposed to deal with the large memory needs of support vector machines, which require keeping in memory a matrix of size $(m_A + m_B)^2$. Traditional chunking [22] involves keeping the misclassified points from the previous iteration's hyperplane, adding omitted misclassified points and repeating until no omitted point is misclassified. Because this does not guarantee that the matrix, as it increases in size, will fit in memory, Osuna et al. [18] fixed it in size and proved that it will still converge to the complete set optimal solution. Their algorithm therefore requires adding and removing from memory the same number of observations. Bradley and Mangasarian's chunking algorithm [4] decomposes the data into blocks, each representing a different subproblem. At each iteration, the subproblem with the best optimal fonction value is used to add all the misclassified points from its own block, to all the other subproblems where they were omitted.

Our algorithm, for the L_1 norm MSD problem, is less concerned about memory requirements and more about the computational limitations required when solving the $2n$ linear programs, each approximately of size $(m_A + m_B) \times n$. The objectives, however are essentially the same: reduce the number of points needed in the working set to a minimum and keep the number of iterations low as to minimize the number of times the solver is called. To keep the number of included points to a minimum, our method also takes into account that some well classified points by the optimal solution are needed. Thus, the initial working set is chosen according to the distance between the points and an heuristic hyperplane. This strategy has been successfully used by Glen [9] in his MIP heuristic. As opposed to the traditional chunking method, when our method is applied to the MSD

problem, we need to select a better initial working set, with both misclassified and well classified points, to keep the number of iterations and the number of points to a minimum.

4 Results

4.1 Methodology

Two algorithms are tested. The first is *RANDCG*, which randomly selects a sample of points to include in the initial working set and then iteratively adds misclassified points until the optimal solution of the complete set is found. The second is *PLANCG*, which selects both misclassified and well classified points from an heuristic solution and then iteratively adds misclassified points until the optimal solution of the complete set is found.

The two algorithms were programmed with C++ using CPLEX 8.0's C++ libraries. David Musicant's NDC Dataset Generator (NDC) [17] was used to generate the datasets. The NDC algorithm's basic steps go as follows:

- Randomly generate a number of groups centers,
- Distribute points around the groups following a multivariate normal distribution,
- Randomly generate a separating hyperplane,
- Assign the points their class labels based on their group center's position to the hyperplane.

By stretching the covariance matrix, it is possible to increase or decrease problem difficulty, even though the theoretical rate is not available. As in [1], the number of groups was made to be equal to the number of variable used in the datasets. The computer used is a Xeon 3.06GHz with 2Gb of ram.

Even though there can be many ways for obtaining an initial hyperplane for *PLANCG*, we decided to use *RANDCG* on a sample to produce the heuristic solution needed for *PLANCG*. 10% random samples were used. The samples provided sufficiently good initial solutions so that there was no need to use bigger sample sizes.

4.2 Simulations

4.2.1 Capacity The results shown in Table 1 indicate drastic saves in computational times. Problems of less than a hundred thousand points are typically solved between 0.5%-2% of the complete set times, depending on whether an initial hyperplane was used or not.

4.2.2 Robustness Additional iterations, when adjusting the hyperplane, influence little the time. This is probably because of the hot-starts between iterations. The difference in times between *PLANCG* and *RANDCG* is explained by the number of points that are needed in the working set. Because *RANDCG* selects the initial points at random, many of

Table 1: Results with approximately 4% of the points misclassified. Averages of three runs.

	Original	**Orig. w/ Bound**	**Dynamic Selection**	
Size			RANCG	PLANCG
20000	1450	241	15	7
30000	3134	498	30	9
50000	7786	966	73	18
75000	17038	2420	183	35
100000	46102	4706	302	70
250000	-	-	2420	674
500000	-	-	24291	2981
750000	-	-	63033	8010
1000000	-	-	-	17937
1500000	-	-	-	38441
2000000	-	-	-	122856

Table 2: Number of needed points to obtain the optimal solution, for the two dynamic point selection methods.

	Dynamic Selection	
Size	RANCG	PLANCG
20000	2934	1294
30000	4316	1909
50000	7113	3278
75000	10773	4947
100000	14107	6627
250000	35167	16342
500000	69002	32785
750000	103664	48325
1000000	-	65441
1500000	-	97229
2000000	-	130246

points are never needed in finding the optimal solution and unnecessarily slow the problem. On average, *RANDCG* needs 3.55 well classified points for every misclassified points in the optimal solution in order to find it. *PLANCG*, because it initially chooses points that are likely to be needed as the hyperplane adjusts itself, needs only 1.66 well classified points for every misclassified points in order to find the optimal solution. Averages of needed points for tested problems are displayed in Table 2.

Another important factor is the difficulty of separation. Since the number of points needed is proportional to the difficulty of separation, the more difficult is a problem the lesser the advantage of using the dynamic point selection method should be.

The dynamic point selection algorithm does not seem to be especially sensible to the number of variables in the problem, at least, not more than the traditional L_1 formulation itself. We did, however, notice a little more variation in the resulting times when the problems are more difficult with more variables.

5 Conclusion

We proposed a dynamic point selection algorithm that, when applied to the problem of minimizing the sum of the L_1 norm distances between the misclassified points and the hyperplane, finds the exact solution of the problems within a fraction of the complete set times. It can solve, in reasonable time, problems up to two million points. Further developments include the application of the generic dynamic point selection algorithm to other norms and other problem definitions.

References

[1] Audet, C., P. Hansen, A. Karam, C.D. Ng, S. Perron (2004). "Exact Solution of L_∞-norm and L_2-norm Plane Separation", *Les Cahiers du GERAD* G–2004–84, Technical Report, HEC Montréal, 14 p.

[2] Bajgier, S.M., A. Hill (1982). "An experimental comparison of statistical and linear programming approaches to the discriminant problem", *Decision Sciences*, 13, 604–618.

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

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

[5] Cavalier, T.M., J.P. Ignizio, A.L. Soyster A.L. (1989). "Discriminant Analysis via mathematical programming: Certain problems and their causes", *Computers and Operations Research*, 16(4), 353–362.

[6] Freed, N., F. Glover (1986). "Resolving certain difficulties and improving the classification power of LP discriminant analysis formulations", *Decision Science*, 17, 589–595.

[7] Freed, N., F. Glover (1986). "Evaluating alternative Linear Programming Models to solve the two-group discrimination problem", *Decision Science*, 17, 151–162.

[8] Freed, N., F. Glover (1981). "A linear programming approach to the discriminant problem", *Decision Science*, 12, 68–74.

[9] Glen, J.J. (2003). "An iterative mixed integer programming method for classification accuracy maximizing discriminant analysis", *Computer & Operations Research*, 30, 181–198.

[10] Glover, F., S. Keene, B. Duea (1986). "A new class of models for the discrimination problem", *Decision Sciences*, 19, 269–280.

[11] Glover, F. (1990). "Improved Linear programming Models for Discriminant analysis", *Decision Science*, 21, 771–785.

[12] Joachimsthaler, E.A., A. Stam (1990). "Mathematical-Programming Approaches for the Classification Problem in 2-Group Discriminant-Analysis", *Multivariate Behavioral Research*, 25(4), 427–454.

[13] Mangasarian, O.L. (1999). "Arbitrary-norm separating plane", *Operations Research Letters*, 24, 15–23.

[14] Mangasarian, O.L. (2000). "Data Selection for Support Vector Machine Classifiers", *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, Boston (USA), 64–70.

[15] Marcotte, P., G. Marquis, G. Savard (1995). "A new implicit enumeration scheme for the discriminant analysis problem", *Computers Operations Research*, 22(6), 625–639.

[16] Markowski, E.P., C.A. Markowski (1987). "An experimental comparison of several approaches to the discriminant problem with both qualitative and quantitative variables", *European Journal of Operational Research*, 28(1), 74–78.

[17] Musicant, D.R. (1998). *NDC: Normally Distributed Clustered datasets* [online], Madison, University of Wisconsin. <http://www.cs.wisc.edu/dmi/svm/ndc/>

[18] Osuna, E., R. Freund, F. Girosi (1997). "An Improved Training Algorithm for Support Vector Machines", *Proceedings of Neural Networks for Signal Processing VII*, Amelia Island (USA), 276–285.

[19] Platt, J. (1998). "Sequential minimal optimization: A fast algorithm for training support vector machines", Technical Report 98–14, Microsoft Research, Redmond, Washington, April 1998.

[20] Rubin, P.A. (1990). "Heuristic solution procedures for a mixed-integer programming discriminant model", *Managerial Decisions and Economics*, 11(4), 255–266.

[21] Vapnik, V. (1998). *Statistical Learning Theory*, Springer-Verlag, New York, 736 p.

[22] Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York, 399 p.