Large Margin LAD Classifiers

Tibérius O. Bonates[a] Peter L. Hammer[b]

RRR 22-2007, July 30, 2007

[a]RUTCOR – Rutgers Center for Operations Research, 640 Bartholomew Road, Piscataway, NJ 08854
[b]RUTCOR – Rutgers Center for Operations Research, 640 Bartholomew Road, Piscataway, NJ 08854

Rutcor Research Report
RRR 22-2007, July 30, 2007

Large Margin LAD Classifiers

Tibérius O. Bonates Peter L. Hammer

Abstract. In this report we present an optimization approach for model construction in Logical Analysis of Data (LAD) that unifies the distinct tasks of pattern generation and creation of a discriminant function. We also investigate how accurate are the LAD models built with such an algorithm and how they compare with other classification models. The main novelty in our algorithm is that the underlying optimization model proposed is more general than other existing methods for model construction in LAD. Moreover, the resulting algorithm is practically parameter-free, which considerably simplifies the task of constructing an accurate LAD model for a new dataset.

Acknowledgements: We are deeply thankful to Alex Kogan, for helpful discussions on a preliminary version of this text. The first author would like to acknowledge the generous support of a DIMACS Graduate Student Award.

1 Introduction

In this report we present an optimization approach for model construction in LAD that unifies the distinct tasks of pattern generation and creation of a discriminant function.

Consider a dataset $\Omega = \Omega^+ \cup \Omega^-$, with $\Omega^+ \cap \Omega^- = \emptyset$. In this report we assume for the sake of simplicity that Ω is a dataset with n binary attributes. The ideas described here, however, can be applied to datasets with numerical attributes by the use of a discretization procedure, such as the one described in [7].

We recall that a LAD model consists of a set M of patterns, so that every observation in Ω^+ satisfies at least one of the positive patterns in M, and every observation in Ω^- satisfies at least one of the negative patterns in M. The construction of a LAD model is a computationally expensive task that typically involves the enumeration of a large set of patterns satisfying a certain property, followed by the selection of a relatively small subset of those.

The fact that only a fraction of the patterns generated is ultimately used for the construction of a LAD model suggests that the approach described above for model construction may not be the most efficient one. Moreover, the choice of enumerating a specific family of patterns, and the usual selection of a small number of those patterns – which is frequently guided by a greedy criterion – may not be the most appropriate for certain problems, resulting in sub-optimal LAD models. In view of this, it is appropriate to say that there is a clear need for a global criterion for choosing an overall "best" LAD model.

For a given set of patterns, the construction of a discriminant function for classification is usually done either by majority vote among positive and negative patterns, or by solving a linear program that provides a set of weights that result in an optimal separation of the two classes [8]. In this study, we propose an algorithm based on the linear programming formulation of [8] that finds an optimal discriminant function defined over the set of all patterns.

Since the total number of patterns can be extremely large, even for datasets of modest dimensions, the direct solution of the LP formulation of [8] over the set of all patterns is of no practical use. We propose the use of the classical column generation technique [15, 16, 20, 23, 30] for iteratively generating patterns as needed during the search for an optimal discriminant function. We show how a pseudo-Boolean optimization subproblem can be formulated which returns the most suitable pattern for improving the discriminant function at any given iteration. The algorithm starts from an arbitrary LAD model and proceeds until no pattern can be found that improves the current discriminant function, or until only marginal improvements of the discriminant function can be obtained. We discuss some computational issues about the implementation of the algorithm and report numerical results obtained on publicly available datasets from the UCI Machine Learning Repository [24].

2 Constructing an Optimal Discriminant Function

Several pattern generation algorithms [1, 3, 5] and selection criteria for constructing LAD models [2, 17] have been proposed. However, none of these approaches for pattern generation or model construction proved to be consistently superior to the others on the datasets used for experimentation. Indeed, practice has shown that fine tuning the control parameters that govern the construction of LAD models, including the choice of pattern generation and model construction algorithms, is a time-consuming task. It is therefore important to develop an algorithm that selects a set of patterns and a set of weights to build an optimal LAD model according to a certain performance criterion that relates to the robustness of classification of unseen observations. A natural such criterion would the *separation margin* of the classifier [27, 28]. In the context of LAD, the separation margin is simply the difference between the smallest value that the discriminant function takes over the positive points of the training set that are correctly classified and the largest value that the discriminant function takes over the negative points in the training set that are correctly classified. We recall here that the LAD discriminant function corresponding to sets of positive and negative patterns $\mathcal{P}$ and $\mathcal{N}$, and associated vectors of weights α and β (respectively) is given by

$$\Delta(\omega) = \sum_{P_i \in \mathcal{P}} \alpha_i P_i(\omega) - \sum_{N_i \in \mathcal{N}} \beta_i N_i(\omega).$$

Thus, the separation margin corresponding to a given discriminant function is

$$\min\{\Delta(\omega) : \omega \in \Omega^+, \Delta(\omega) > 0\} - \max\{\Delta(\omega) : \omega \in \Omega^-, \Delta(\omega) < 0\}.$$

By maximizing the separation margin, we expect a robust classification of unseen examples. This expectation has been confirmed in practice in many situations [9, 19, 25]. The good performance of boosting and voting-based classifiers has also been shown to be related to the implicit maximization of the separation margin of the resulting classifier [25].

2.1 Maximizing the Separation Margin

In this subsection we show how to formulate the problem of constructing an optimal discriminant function by using a linear programming formulation similar to the one described in [8].

Let $\mathbb{P}$ be the set of all positive patterns having homogeneity at least ρ^+, and let $\mathbb{N}$ be the set of all negative patterns having homogeneity at least ρ^-. Let $\mathbb{P} = \{P_1, \dots, P_{|\mathbb{P}|}\}$, $\mathbb{N} = \{N_1, \dots, N_{|\mathbb{N}|}\}$, and for every $\omega \in \Omega$ let $P_i(\omega) = 1$, if observation ω is covered by pattern P_i, and 0 otherwise. Similarly, let $N_j(\omega) = 1$ if ω is covered by N_j, and 0 otherwise. We are interested in solving the following problem:

$$
\begin{aligned}
\text{(P)}\quad \text{maximize} \quad & r + s \;-\; C\sum_{\omega\in\Omega} \epsilon_\omega \\
\text{subject to:} \quad & \\
& \sum_{i=1}^{|\mathbb{P}|} \alpha_i P_i(\omega) - \sum_{j=1}^{|\mathbb{N}|} \beta_j N_j(\omega) + \epsilon_\omega \geq r, \;\; \forall\, \omega \in \Omega^+ \qquad (1)\\
& \sum_{i=1}^{|\mathbb{P}|} \alpha_i P_i(\omega) - \sum_{j=1}^{|\mathbb{N}|} \beta_j N_j(\omega) - \epsilon_\omega \leq -s, \;\; \forall\, \omega \in \Omega^- \qquad (2)\\
& \sum_{i=1}^{|\mathbb{P}|} \alpha_i = 1 \\
& \sum_{j=1}^{|\mathbb{N}|} \beta_j = 1 \\
& r \geq 0, s \leq 0 \\
& \alpha_i \geq 0, i = 1, \ldots, |\mathbb{P}| \\
& \beta_j \geq 0, j = 1, \ldots, |\mathbb{N}| \\
& \epsilon_\omega \geq 0, \; \forall\, \omega \in \Omega,
\end{aligned}
$$

where the values of α_i $(i = 1, \ldots, |\mathbb{P}|)$ and β_j $(j = 1, \ldots, |\mathbb{N}|)$ are the weights of the positive and negatives patterns, respectively, r represents the positive part of the separation margin, s represents the negative part of the separation margin, and C is a nonnegative penalization parameter that controls how much importance is given to the violations ϵ_ω of the separating constraints (1) and (2).

Due to the potentially very large total number of patterns, it is highly unlikely that in real-world applications one will encounter a dataset where such a formulation can be directly tackled. Thus, let us consider a subset of positive patterns $\mathcal{P}^0 \subset \mathbb{P}$ and a subset of negative patterns $\mathcal{N}^0 \subset \mathbb{N}$, and let us denote by $I^0 \subset \{1, \ldots, |\mathbb{P}|\}$ and $J^0 \subset \{1, \ldots, |\mathbb{N}|\}$ the sets of indices corresponding to the elements of $\mathcal{P}^0$ and $\mathcal{N}^0$, respectively. Then, we can write a restricted version of (P) as

$$\begin{aligned}
\text{(RP)}\quad \text{maximize} \quad & r + s \; - \; C\sum_{\omega\in\Omega}\epsilon_\omega \\
\text{subject to:} \quad & \\
& r - \sum_{i\in I^0}\alpha_i P_i(\omega) + \sum_{j\in J^0}\beta_j N_j(\omega) - \epsilon_\omega \le 0, \;\; \forall\, \omega\in\Omega^+ && (3)\\
& s + \sum_{i\in I^0}\alpha_i P_i(\omega) - \sum_{j\in J^0}\beta_j N_j(\omega) - \epsilon_\omega \le 0, \;\; \forall\, \omega\in\Omega^- && (4)\\
& \sum_{i\in I^0}\alpha_i = 1 && (5)\\
& \sum_{j\in J^0}\beta_j = 1 && (6)\\
& r \ge 0, s \le 0\\
& \alpha_i \ge 0, \;\forall\, i\in I^0\\
& \beta_j \ge 0, \;\forall\, j\in J^0\\
& \epsilon_\omega \ge 0, \;\forall\, \omega\in\Omega.
\end{aligned}$$

Note that we rearranged constraints (3) and (4) in order to write (RP) in a standard maximization form. It is easy to choose $\mathcal{P}^0$ and $\mathcal{N}^0$ in such a way that (RP) has a feasible solution. We recall that the *minterm* corresponding to a given observation ω is simply the pattern with the largest possible number of literals that is satisfied by observation ω, i.e. it is the pattern given by the following conjunction: $T = \prod_{i:\omega_i=1} x_i \prod_{j:\omega_j=0} \overline{x}_j$.

Clearly, T is a pattern that covers ω and does not cover any observation from the opposite part of the dataset (since $\Omega^+ \cap \Omega^- = \emptyset$). Let us choose $\mathcal{P}^0$ and $\mathcal{N}^0$ to be the sets of minterms corresponding to the positive and negative observations in the dataset, respectively. Then, the vectors α and β given by $\alpha_i = \frac{1}{|\mathcal{P}^0|}$, $i \in \mathcal{P}^0$ and $\beta_j = \frac{1}{|\mathcal{N}^0|}$, $j \in \mathcal{N}^0$, along with $r = \frac{1}{|\mathcal{P}^0|}$, and $s = \frac{1}{|\mathcal{N}^0|}$ constitute a feasible solution to (RP). In fact, constructing a feasible set of patterns to initialize (RP) is not a difficult task; one pattern of each class would suffice. However, finding a good set of patterns to initialize the column generation algorithm is always advantageous as such patterns are likely to be part of the optimal set of patterns and their use in early iterations may significantly speed up the entire procedure.

2.2 The Pricing Subproblem

In this subsection we address the subproblem phase in the column generation algorithm. We formulate this problem as a pseudo-Boolean optimization problem that can also be seen as a weighted version of the *maximum pattern problem* [5, 14].

The solution of problem (RP) provides an optimal discriminant function for the set of patterns $\mathcal{P}^0 \cup \mathcal{N}^0$. However, that discriminant function may not be optimal with respect to the entire set of all patterns. In order to verify global optimality, we need to make sure that there is no pattern that, once added to the current set of patterns, allows for an improvement in the value of the objective function of (RP).

In the simplex method, one of the commonly used heuristics for choosing a non-basic variable to enter the current basis is to choose the variable with the best (in the maximization case, the largest) reduced cost [20, 23, 10, 30]. This criterion is usually referred to as *Dantzig's rule* or *steepest descent rule* [10, 23]. Similarly, we use the reduced cost of a specific pattern P to measure the potential improvement in the objective function of (RP) resulting from the introduction of P into the set of known patterns $\mathcal{P}_0 \cup \mathcal{N}_0$. To verify the optimality of the current solution of (RP), we solve a pricing problem that will return a pattern with the best reduced cost with respect to the current values of the dual variables of (RP). Because (RP) is a maximization problem, we are interested in finding a pattern with the largest reduced cost.

According to the usual optimality criterion of the primal simplex method, if there is no pattern (not in $\mathcal{P}^0 \cup \mathcal{N}^0$) with a nonnegative reduced cost, then the current solution of (RP) is optimal to (P); otherwise, a new pattern can be added to the set $\mathcal{P}^0 \cup \mathcal{N}^0$ and the algorithm proceeds. Notice here that we allow the introduction of a pattern whose reduced cost is zero. Since the column generation algorithm is equivalent to applying the simplex algorithm to a large formulation, there is a possibility of having degenerate iterations in which no progress is made in terms of the objective function, exactly as in the case of the simplex method [10, 23, 30]. We will refer to a pattern with nonnegative reduced cost at a given iteration as a *candidate pattern*.

Note that since $r + s$ corresponds to the separation margin of the current discriminant function, solving the pricing problem corresponds to asking if the current separation margin can be enlarged by the addition of a pattern. Thus, the column generation algorithm applied to problem (P) attempts to increase the separation margin by including patterns that make the distinction between the sets Ω^+ and Ω^- more pronounced.

From now on, let us assume that k iterations of the column generation algorithm have been executed. We will refer to the corresponding sets of patterns as $\mathcal{P}^k$ and $\mathcal{N}^k$, and to the corresponding version of problem (RP) as (RP^k). Let λ and μ be the vectors of dual variables associated to constraints (3) and (4), respectively, at an optimal solution of (RP^k). Similarly, let θ^+ and θ^- be the dual variables corresponding to constraints (5) and (6). The reduced costs corresponding to a positive pattern P_i and a negative pattern N_j are given by $\theta^+ + [\lambda^\top, -\mu^\top]\pi_i$ and $\theta^- + [-\lambda^\top, \mu^\top]\nu_j$ [10, 30, 20], respectively, where $\pi_i = [P_i(\omega)]_{\omega\in\Omega}$ and $\nu_j = [N_j(\omega)]_{\omega\in\Omega}$ are the characteristic vectors of the observations satisfying P_i and N_j, respectively.

Let us define binary decision variables x_i and x_i^c $(i = 1, \ldots, n)$ associated to attribute a_i in the following way: (i) $x_i = 1$ and $x_i^c = 0$ if the literal associated to $a_i = 1$ is used in the resulting pattern; (ii) $x_i = 0$ and $x_i^c = 1$ if the literal associated to $a_i = 0$ is present in the resulting pattern. Thus, the positive pricing subproblem associated to optimal vectors of dual variables λ^* and μ^* of (RP) can be formulated as the following pseudo-Boolean optimization problem:

$$\text{(SP}^+\text{)} \qquad \text{maximize} \quad \sum_{\omega\in\Omega^+} \lambda^*_\omega \left(\prod_{i:\omega_i=0} \overline{x_i} \prod_{j:\omega_j=1} \overline{x^c_j} \right) - \sum_{\omega\in\Omega^-} \mu^*_\omega \left(\prod_{i:\omega_i=0} \overline{x_i} \prod_{j:\omega_j=1} \overline{x^c_j} \right)$$

subject to:

$$x_j, x^c_j \in \{0,1\},\ j = 1,\ldots,n,$$

where each term $\prod_{i:\omega_i=0} \overline{x_i} \prod_{j:\omega_j=1} \overline{x^c_j}$ takes the value 1 whenever ω is covered by the resulting conjunction, and 0 otherwise. In addition to the formulation of problem (S^k), we need to enforce that the resulting conjunction is a positive pattern.

Clearly, problem (SP$^+$) is an instance of the family of pseudo-Boolean optimization problems whose solution was addressed in [4]. In fact, we utilize a simple modification of our branch-and-bound algorithm BBPBF to solve (SP$^+$). The only difference between the algorithm described in [4] and the one used in this report to study (SP$^+$) is that whenever we consider a candidate solution for problem (SP$^+$) we first check whether it satisfies the definition of a pattern, according to the homogeneity level required. If the solution satisfies the definition, it is considered as a feasible solution, otherwise it is not taken into account and the algorithm proceeds normally. This procedure is equivalent to the introduction of a large negative penalty for violating the definition of a pattern.

Dual variable θ^+ appears in the formulation of (SP$^+$) because the problem is formulated with the goal of constructing a positive pattern, and therefore it is implicit that the column corresponding to this pattern has a coefficient equal to 1 in the row corresponding to constraint (5). Thus, the value of θ^+, despite being a constant, is part of the formulation of (SP$^+$).

While (SP$^+$) is aimed at finding the best positive candidate pattern at the current iteration, it may be necessary to search for a negative candidate pattern as well. In fact, even if we have already found a positive candidate pattern, it may be advantageous to generate a negative candidate pattern, if there is any. We suggest to solve both (SP$^+$) and its negative counterpart, (SP$^-$), at every iteration. For the negative pricing problem (SP$^-$), we simply minimize the same objective function as in problem (SP^k) — as opposed to maximizing in (SP$^+$) — and require that the resulting conjunction satisfies the condition of being a negative pattern. Here, as in the case of (SP$^+$), a slight modification of algorithm BBPBF can be used to solve (SP$^-$).

The solutions of (SP$^+$) and (SP$^-$) provide either: (a) a certificate of optimality of the current discriminant function, in case the objective function value of (SP$^+$) is strictly positive and that of (SP$^-$) is strictly negative; or (b) a new positive or negative pattern (or both) to be added to our current set of known patterns. In general, if at iteration k we generate new patterns $P^* \in \mathbb{P}$ and $N^* \in \mathbb{N}$, then we define the sets of patterns for the next iteration as $\mathcal{P}^{k+1} \leftarrow \mathcal{P}^k \cup \{P^*\}$ and $\mathcal{N}^{k+1} \leftarrow \mathcal{N}^k \cup \{N^*\}$.

Let (x^*, x^{c*}) be an optimal solution of (SP$^+$). In practice, by adding a new positive pattern to the current set of patterns we are actually adding the following column to problem

(RP^k):

$$\begin{bmatrix} \left(-\prod_{i:\omega_i=0} \overline{x_i^*} \prod_{j:\omega_j=1} \overline{x_j^{c*}} \right)_{\omega\in\Omega^+} \\ \left(\prod_{i:\omega_i=0} \overline{x_i^*} \prod_{j:\omega_j=1} \overline{x_j^{c*}} \right)_{\omega\in\Omega^-} \\ 1 \\ 0 \end{bmatrix}.$$

2.3 The Column Generation Algorithm

The column generation algorithm as described above alternates the solution of problem (RP) with the solution of (SP$^+$) and (SP$^-$). While problem (RP) optimizes the discriminant function over the current set of patterns, problems (SP) verify the optimality of the discriminant function obtained by (RP) and, if necessary, produce new patterns to improve the current discriminant function. Thus, the iterative process stops at one of the following events: (i) the maximum number of iterations is reached; (ii) no candidate pattern can be found; (iii) no significant improvement of the objective function of (RP) takes place after a large number of iterations. For the latter case, we need to define two control parameters: a tolerance value corresponding to the smallest increase in the objective function of (RP) to be regarded as a significant increase, and the maximum number of iterations to be allowed without a significant increase. These control parameters are important, since in practice it is possible to have long sequences of iterations in which the objective function of (RP) remains unchanged or changes only slightly.

Although column generation algorithms usually approach the neighborhood of a near-optimal solution in a reasonably small number of iterations [21, 23], achieving or proving optimality can take a very large number of iterations. Typically, towards the end of the algorithm's execution, the graph corresponding to the progress in the objective function value exhibits a long "tail", reflecting the failure in making significant progress during a large number of iterations. This phenomenon is known as the *tailing-off effect* [16, 20, 21, 23, 29]. Aside from numerical instability due to the finite precision used in the implementation of such algorithms, the tailing-off phenomenon can be explained by the fact that the solutions produced by (RP) may be interior points to the original problem (P). Therefore, as the algorithm approaches an optimal solution, a significant amount of "cleaning up" may be necessary to arrive at a vertex of the original polyhedron [23]. This may require the generation of several columns in order to make small adjustments in the values of some variables of (RP). In our experiments, we used the tolerance parameter equal to 10^{-4} and the maximum number of degenerate iterations equal to 10. The maximum number of iterations allowed for the entire execution of the algorithm was set to $1,000$, but was never reached in our experiments.

Figure 1 summarizes the algorithm. We use the notation (RP(k)), (SP(k)$^+$) and (SP(k)$^-$) to denote the optimization problems solved at iteration k.

```
1. Input: $\Omega^+, \Omega^-$, initial sets $\mathcal{P}^0$ and $\mathcal{N}^0$ of patterns. Set $k = 0$.
2. Solve (RP($k$)), with optimal primal and dual solutions $(r^*, s^*, \alpha^*, \beta^*)$ and
   $(\lambda^*, \mu^*, \theta^{+*}, \theta^{-*})$.
3. Solve (SP($k$)$^+$) and (SP($k$)$^-$), with optimal positive and negative
   patterns $P^*$ and $N^*$.
4. If at least one of the stopping criteria is met, stop.
5. If $P^*$ or $N^*$ (or both) are candidate patterns, define $\mathcal{P}^{k+1} = \mathcal{P}^k \cup \{P^*\}$ and
   $\mathcal{N}^{k+1} = \mathcal{N}^k \cup \{N^*\}$ accordingly, and set $k = k + 1$.
6. Go to Step 2.
```

Figure 1: Pseudocode of the column generation algorithm.

As an alternative, our algorithm can be implemented with the utilization of the original BBPBF algorithm for the solution of problems (SP$^+$) and (SP$^-$). By not imposing specific limits on the coverage of the conjunctions generated by BBPBF (i.e., by not requiring that the conjunctions are patterns with prescribed minimum prevalence and homogeneity), we allow the algorithm to self-adjust to the given training set. The algorithm will generate the types of patterns required for a robust classification, without the need for an initial estimation of how much homogeneity or prevalence should be required for the given training data. Such a modification reduces even more the complexity of the LAD training procedure, as it removes the need to fine tune the homogeneity and prevalence parameters. In fact, this is the version of the algorithm actually utilized in the computational experiments reported in the next section.

2.4 Overfitting

An important remark with respect to the termination criterion of our algorithm is that it may force termination at a sub-optimal solution. In many applications of optimization algorithms, reaching optimality is not a real concern. In our current context, this is also true. The large margin criterion is simply a heuristic objective function that we use for guiding our search for a good classifier. In machine learning, finding an optimal solution frequently corresponds to selecting from a certain family of functions one that fits best a given dataset. This can clearly lead to *overfitting* [22, 12, 18] the given data. Indeed, as suggested in [12], by solving the optimization problem (P) we are trying to find the best possible separation of the training data, while the actual objective function should be to provide a good separation for unseen observations.

The early termination of our iterative algorithm due to the tailing-off phenomenon can be seen as a simple procedure for avoiding overfitting the training data. Another simple stopping criterion used in our experiments refers to the patterns obtained by solving problems (SP$^+$)

and (SP^-). If the coverage of the patterns obtained during a number of consecutive iterations has always been less than a certain percentage of the training set, we stop the algorithm. The rationale behind this test is that the inclusion of patterns of low coverage suggests that, at the current iteration, no significant global trend in the training data can be found in order to improve the separation margin. To avoid making small adjustments fo fit local characteristics of the training sample, we stop the algorithm and report the current solution as the best one found.

Other usual overfitting-preventing mechanisms involve imposing an explicit limit on, or penalizing the complexity of the classifier built [12, 18, 22]. As suggested above, this mostly accounts for preventing the construction of a discriminant function based on a large collection of patterns, many of which cover small subsets of the training data. Our algorithm allows a certain level of extra control over such issues via the setting of parameter C in problem (RP), by means of which we can define how much emphasis we want to give to an accurate separation of the training data.

3 Computational Experiments

The algorithm proposed in this study was implemented in C++, using the *MS Visual C++ .NET 1.1* compiler. The linear programming formulations solved as part of the column generation algorithm were solved using the callable library of the Xpress-MP optimizer [11]. The computer used for carrying out the experiments reported here was an *Intel Pentium*® 4, 3.4GHz, with 2GB of RAM.

Problems (SP^+) and (SP^-) were not necessarily solved to optimality in our experiments. Given the satisfactory performance of the truncated versions of the BBPBF algorithm, as reported in [4], we established explicit limits on the execution of the variant of the BBPBF algorithm utilized in this study. The maximum number of branch-and-bound nodes explored in the initialization procedure was set to 1,000, while the number of additional nodes explored during the remainder of the search was set to 2,000. The maximum time limit of 30 minutes was also imposed, but never reached in our experiments.

In Figure 2, we show six histograms illustrating a typical example of how the frequencies of values of the discriminant function evolve during the execution of our algorithm. The graphs correspond to the application of our training algorithm to a randomly extracted sample of 90% of the observations in the "heart" dataset, from the UCI Repositrory [24]. The empty region in the middle of the graphs is the separation margin between positive and negative points. The graphs shows how our algorithm iteratively improves the separation margin as it finds candidate patterns and adjusts the weights of the discriminant function accordingly.

Although all observations are correctly separated in each of the iterations, we can see that a large number of observations are on the border of the separating region in early iterations, i.e. their corresponding constraints from sets (3) and (4) are binding. The observations on the border of the margin of separation can be seen as "support observations", just as in the case of support vector machines (see, e.g., [26, 28]). Clearly, the ideal situation would be to have just a few observations on the border, and a larger number of observations as we move towards the extreme regions of the graph. Discriminant functions with such a distribution can be expected to perform better on unseen examples than the ones shown in Figure 2(a) and 2(b), for instance. In subsequent iterations (see Figures 2(c) and 2(d)) we can notice a better distribution of the observations in terms of discriminant value, with substantially less observations on the border of the separating region. Finally, in Figures 2(e) and 2(f) we see that while trying to increase margin of separation, the algorithm adjusts the discriminant function causing a simulatneous increase in the number of support observations. There is here a natural trade-off between the margin of separation and the number of support observations. The early termination criteria discussed in Section 2 cause the automatic interruption of the algorithm at iteration 72, after a sequence of 10 iterations has taken place without a substantial increase in the size of the margin of separation.

Note that here, as in the case of support vector machines, there is a natural interpretation of the dual solution of problem (RP(k)). The values of the dual variables can be used to describe a *dual classifier* consisting of a linear combination of the support observations (described as 0-1 vectors in the pattern-space corresponding to iteration k). The smaller the number of supporting observations, the simpler the dual classifier will be.

In Figure 3 we present the values of the positive and negative margins during the execution of the same experiment on the "heart" dataset. It can be seen that roughly after iteration 50 the algorithm stops making significant improvements in terms of increase in the separation margin (the tailing-off phenomenon described in the previous section). That fact seems to be accompanied by an increase in the number of observations at the border of the separation margin, as discussed above (see Figure 2). This observation helped us to adjust the parameters for early termination that prevent the tailing-off phenomenon and avoid overfitting the discriminant function to the training sample.

Table 1 report the accuracy of five algorithms implemented in the Weka package, the maximum patterns-based LAD version described in [5] (termed CAP-LAD), and those of LAD models built utilizing the algorithm described in this report (termed LM-LAD, for Large Margin LAD). The results were obtained as the average result of a 10-fold cross-validation experiment. We display the highest accuracy for each dataset in boldface characters. Moreover, at the bottom of the table we display the so-called Borda count [6] of each algorithm. The Borda count in Table 1 is obtained by ranking the seven algorithms based on their accuracies on a given dataset. The most accurate algorithm is assigned a score of 7, the second best is assigned a score of 6, and so on until a score of 1 is assigned to the algorithm with the worst accuracy on that dataset. The Borda count of an algorithm is the sum of these scores over the ten datasets, and summarizes its comparative performance.

In Table 2 we present the counts of wins, losses, and ties for the pairwise comparison of the seven algorithms. Each pairwise comparison in Table 2 is made with respect to the 95% confidence interval presented in Table 1.

While being superior to some of the other algorithms for some datasets, the performance of the LM-LAD algorithm is, strictly speaking, generally inferior to that of other algorithms on the datasets used in our experiments. The Borda count shown in Table 1 summarizes

Dataset	SMO	J48	Rand.For.	Mult.Perc.	S. Log.	CAP-LAD	LM-LAD
breast-w	0.965 ± 0.011	0.939 ± 0.012	**0.967 ± 0.009**	0.956 ± 0.012	0.963 ± 0.013	0.966 ± 0.011	0.942 ± 0.024
credit-a	0.864 ± 0.025	0.856 ± 0.031	**0.882 ± 0.027**	0.831 ± 0.032	0.873 ± 0.025	0.867 ± 0.025	0.815 ± 0.044
hepatitis	0.772 ± 0.084	0.652 ± 0.086	0.722 ± 0.101	0.727 ± 0.065	0.764 ± 0.090	**0.779 ± 0.104**	0.738 ± 0.091
krkp	**0.996 ± 0.003**	0.994 ± 0.003	0.992 ± 0.003	0.993 ± 0.002	0.975 ± 0.005	0.993 ± 0.002	0.962 ± 0.031
boston	0.889 ± 0.028	0.837 ± 0.045	0.875 ± 0.024	**0.893 ± 0.031**	0.874 ± 0.021	0.855 ± 0.029	0.840 ± 0.045
bupa	0.701 ± 0.045	0.630 ± 0.041	0.731 ± 0.046	0.643 ± 0.020	0.662 ± 0.048	**0.734 ± 0.052**	0.678 ± 0.034
heart	**0.837 ± 0.039**	0.799 ± 0.052	0.834 ± 0.051	0.815 ± 0.025	0.834 ± 0.040	0.826 ± 0.032	0.814 ± 0.033
pima	0.727 ± 0.029	0.722 ± 0.026	0.736 ± 0.030	0.726 ± 0.023	0.729 ± 0.031	**0.747 ± 0.022**	0.682 ± 0.023
sick	0.824 ± 0.027	**0.926 ± 0.020**	0.832 ± 0.023	0.852 ± 0.049	0.808 ± 0.023	0.825 ± 0.019	0.815 ± 0.041
voting	0.961 ± 0.018	0.960 ± 0.015	0.961 ± 0.016	0.944 ± 0.025	**0.961 ± 0.014**	0.952 ± 0.021	0.945 ± 0.025
Borda count	52	28	52	35	41	50	20

Table 1: Classification accuracy and Borda counts of Weka algorithms, CAP-LAD and LM-LAD.

	SMO	J48	Rand.For.	Mult.Perc.	S. Log.	CAP-LAD
J48	4-1-5					
Rand.For.	1-1-8	4-1-5				
Mult.Perc.	0-4-6	2-2-6	0-2-8			
S.Log.	1-1-8	1-2-7	0-2-8	1-2-7		
CAP-LAD	0-1-9	2-1-7	0-0-10	2-1-7	1-0-9	
LM-LAD	0-0-10	0-1-9	0-1-9	0-0-10	0-0-10	0-1-9

Table 2: Matrix of wins, losses and ties.

this observation. However, the accuracies obtained by our algorithm are reasonably close to those of the other algorithms (as can be seen from Table 2). Indeed, as discussed in [13], the direct comparison of the accuracy of two algorithms in the experimental setup utilized here must be regarded cautiously, while the conclusion that the two accuracies are not statistcally different is to be trusted. Moreover, if for each dataset we compare the accuracy of LM-LAD with the highest accuracy of the other six algorithms, we can see that LM-LAD achieves on average 94.3% of the accuracy of the best algorithm.

It is important to mention that the LM-LAD algorithm was not calibrated for any of the ten benchmark datasets. In fact, the C parameter was fixed at 1.0 in all experiments and no constraint has been imposed on the degree, prevalence or homogeneity of the patterns generated.

4 Conclusions

In this report we have described a novel algorithm for constructing LAD models that unifies the usually distinct tasks of generating a set of patterns and defining a discriminant function. We have also investigated how accurate are the LAD models built with such an algorithm and how they compare with other classification models.

The main novelty in our algorithm is that the underlying optimization model proposed here is more general than other existing methods for model construction in LAD. Other existing approaches for pattern generation have limitations of a certain type: on the types of patterns generated (spanned [1], prime [3], maximum [5]) and on the degrees of the patterns enumerated (except for the case of maximum patterns). Moreover, the discriminant functions are in general of a very simple form, with patterns of the same class having equal weights [5, 8]. In each of the previous LAD implementations, the choices of a model and a discriminant function were limited by one or more such factors, none of which is present in our approach.

Indeed, the pricing problems (SP^+) and (SP^-) can produce patterns of any type, including patterns of arbitrary degree. Moreover, the iterative procedure presented here allows for the construction of a globally optimal discriminant function.

In addition to this conceptual simplification in the construction of a LAD model, our algorithm is practically parameter-free, having only a few technical control parameters primarily related to the underlying optimization process. Our computational experiments suggest that the set of values described here for these parameters is quite stable, and therefore fixing the parameters at these values allows the straighforward use of the algorithm without any need for careful calibration in order to obtain reasonable results. While the LAD models constructed by our algorithm are not optimal in terms of accuracy, they have accuracies which are typically close to those of the most accurate algorithms used in our experiments.

Moreover, the optimization model utilized by our algorithm is quite simple and can be easily modified to account for different objective functions, or to include alternative penalty terms. The similarities between our algorithm and the standard support vector machines algorithm suggest that some improvement could be made by the use of ideas coming from the SVM literature, such as the use a quadratic formulation, or the use of an alternative parameter ν that bounds from above the fraction of misclassified observations in the training dataset.

Finally, the LM-LAD algorithm described in this report can be seen as a first version of a LAD procedure that completely dispenses with the calibration of parameters such as degree, homogeneity, and prevalence of patterns.

References

[1] Alexe G., P.L. Hammer. Spanned patterns for the logical analysis of data. *Discrete Appl. Math.*, 154(7):1039–1049, 2006.

[2] Alexe G., S. Alexe, P.L. Hammer, A. Kogan. Comprehensive vs. Comprehensible Classifiers in Logical Analysis of Data. *Discrete Applied Mathematics, in press.*

[3] Alexe S., P.L. Hammer. Accelerated Algorithm for Pattern Detection in Logical Analysis of Data. *Discrete Appl. Math.*, 154(7):1050–1063, 2006.

[4] Bonates T.O., P.L. Hammer. A Branch-and-Bound Algorithm for a Family of Pseudo-Boolean Optimization Problems. Technical Report RRR 21-2005, RUTCOR - Rutgers Center for Operations Research, Rutgers University, 2007.

[5] Bonates T.O., P.L. Hammer, A. Kogan. Maximum patterns in datasets. *Discrete Appl. Math. (accepted)*, 2007.

[6] Borda J.C. Mémoire sur les élections au scrutin. *Histoire de l'Academie Royale des Sciences*, 1781.

[7] Boros E., P.L. Hammer, T. Ibaraki, A. Kogan. Logical Analysis of Numerical Data. *Mathematical Programming*, 79:163–190, 1997.

[8] Boros E., P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik. An Implementation of Logical Analysis of Data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, 2000.

[9] Boser B.E., I.M. Guyon, V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[10] Chvátal V. *Linear Programming.* Freeman, New York, 1983.

[11] Dash Associates. Xpress-Mosel Reference Manuals and Xpress-Optimizer Reference Manual, Release 2004G, 2004.

[12] Dietterich T.G. Overfitting and undercomputing in machine learning. *ACM Computing Surveys (CSUR)*, 27(3):326–327, 1995.

[13] Dietterich T.G. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[14] Eckstein J., P.L. Hammer, Y. Liu, M. Nediak, B. Simeone. The Maximum Box Problem and its Application to Data Analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

[15] Gilmore P.C., R.E. Gomory. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9(6):849–859, 1961.

[16] Gilmore P.C., R.E. Gomory. A Linear Programming Approach to the Cutting Stock Problem-Part II. *Operations Research*, 11(6):863–888, 1963.

[17] Hammer P.L., A. Kogan, B. Simeone, S. Szedmák. Pareto-Optimal Patterns in Logical Analysis of Data. *Discrete Applied Mathematics*, 144(1-2):79–102, 2004.

[18] Hastie T., R. Tibshirani, J.H. Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer, 2001.

[19] Krauth W., M. Mezard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A Mathematical General*, 20:L745–L752, August 1987.

[20] Lasdon L.S. *Optimization Theory for Large Systems.* Dover Publications Inc., 2002.

[21] Lubbecke M.E., J. Desrosiers. Selected Topics in Column Generation. *Oper. Res*, 53(6):1007–1023, 2005.

[22] Mitchell T.M. *Machine Learning.* McGraw-Hill Higher Education, 1997.

[23] Nazareth J.L. *Computer solution of linear programs.* Oxford University Press, Inc. New York, NY, USA, 1987.

[24] Newman D.J., S. Hettich, C.L. Blake, C.J. Merz. UCI Repository of machine learning databases, 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[25] Schapire R.E., Y. Freund, P. Bartlett, W.S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

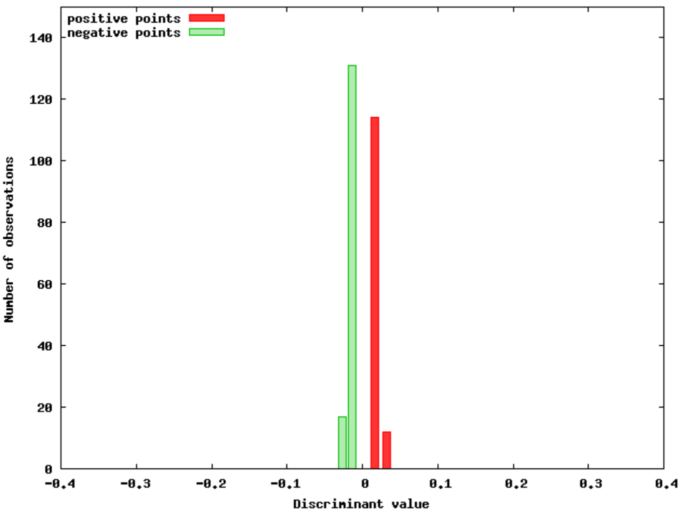
[26] Schölkopf B., Smola A.J. *Learning with Kernels.* MIT Press Cambridge, Mass, 2002.

[27] Shawe-Taylor J., N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

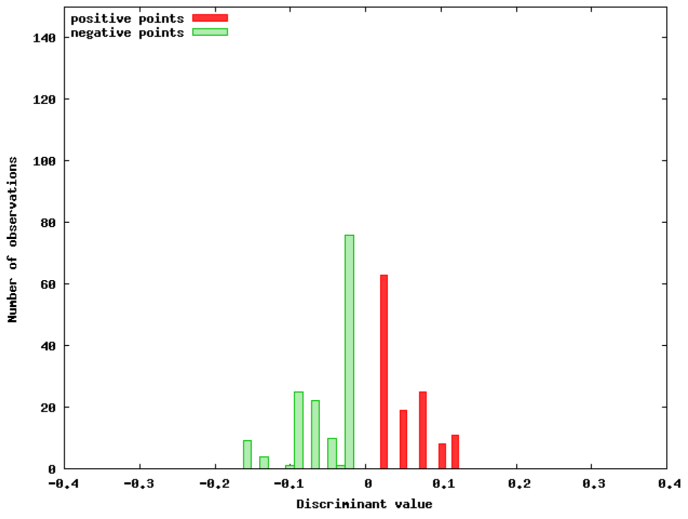
[28] Smola A.J., P. Bartlett, B. Schölkopf, D. Schuurmans. *Advances in Large-margin Classifiers.* The MIT Press, 2000.

[29] Wilhelm W.E. A Technical Review of Column Generation in Integer Programming. *Optimization and Engineering*, 2(2):159–200, 2001.

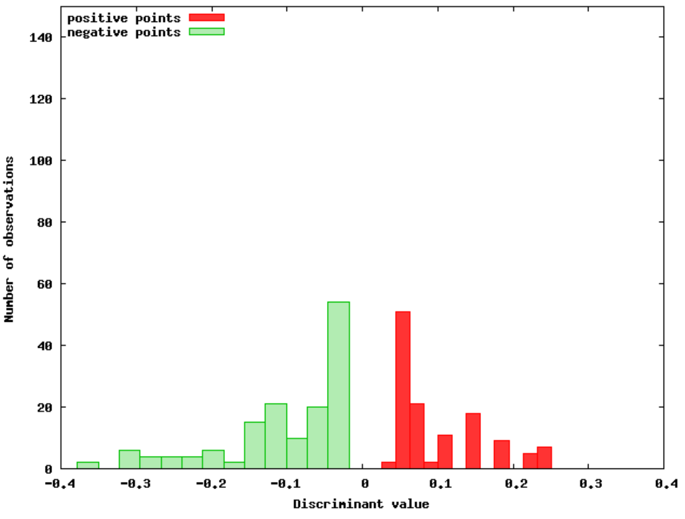
[30] Wolsey L.A. *Integer Programming.* John Wiley & Sons, New York, 1998.



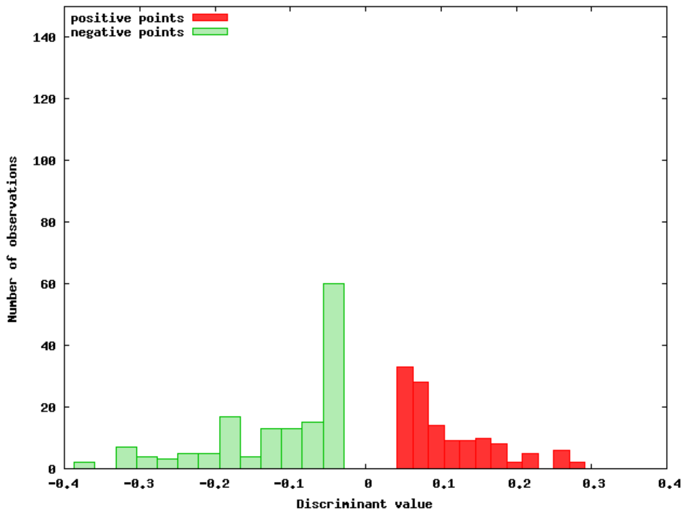
(a) At iteration 2



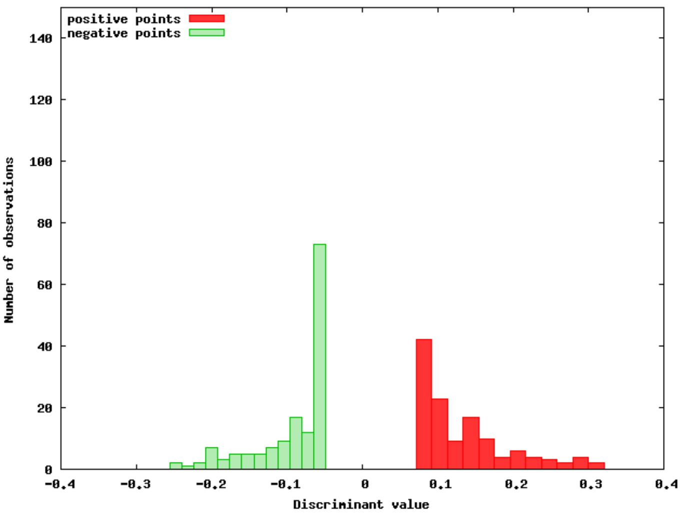
(b) At iteration 5



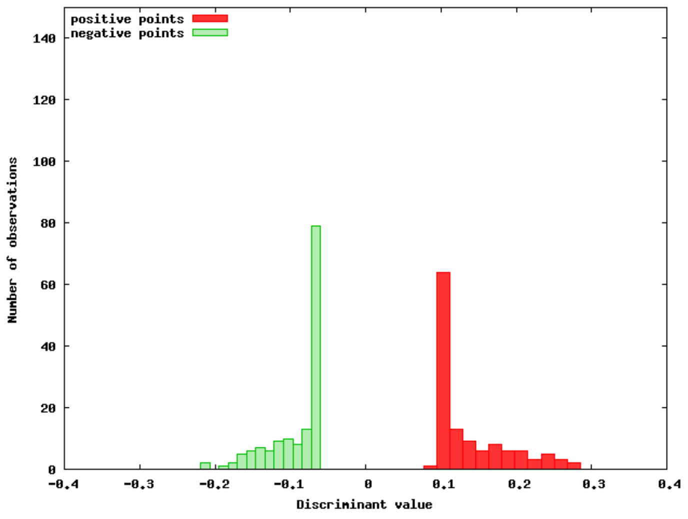
(c) At iteration 10



(d) At iteration 20



(e) At iteration 50



(f) At iteration 72 (last)

Figure 2: Histograms of observations at different discriminant levels for the “heart” dataset.

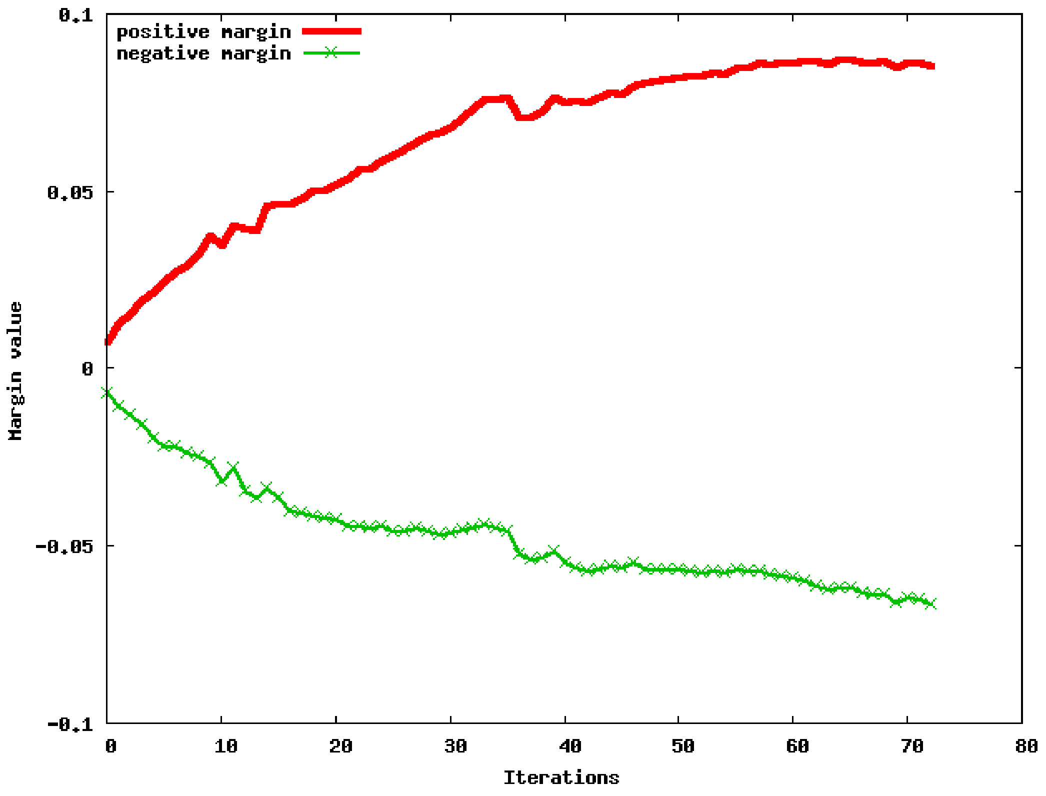


Figure 3: Behavior of the positive and negative margins for the “heart” dataset.