SCISPACE
formerly Typeset

Open access • Journal Article • DOI:10.1109/69.842268

An implementation of logical analysis of data — Source link

Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan ...+2 more authors

Institutions: Rutgers University

Published on: 01 Mar 2000 - IEEE Transactions on Knowledge and Data Engineering (IEEE)

Related papers:

- Cause-effect relationships and partially defined Boolean functions
- Logical analysis of numerical data
- Logical analysis of data—An overview: From combinatorial optimization to medical applications
- Pareto-optimal patterns in logical analysis of data
- Coronary Risk Prediction by Logical Analysis of Data

Share this paper:

View more about this paper here: https://typeset.io/papers/an-implementation-of-logical-analysis-of-data-4a21vgxfgc

IDIAP RESEARCH REPORT

IDIAP
Martigny - Valais - Suisse

An Implementation of Logical Analysis of Data

Endre Boros [†] Peter L. Hammer [†]
Toshihide Ibaraki [‡] Alexander Kogan [†]
Eddy Mayoraz [*] Ilya Muchnik [†]

IDIAP–RR 96-05

July 1996

revised in 98-08

published in
IEEE Trans. on Knowledge and Data Engineering **12**–1 (2000) 292–306

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[†] RUTCOR, Rutgers University, P.O. Box 5062, New Brunswick, NJ 08903-5062
[‡] Applied Mathematics and Physics Dept., Graduate School of Engineering, Kyoto University, Kyoto 606, Japan
[*] Institut Dalle Molle d'Intelligence Artificielle Perceptive (IDIAP), P.O. Box 592, CH-1920 Martigny, Switzerland

IDIAP Research Report 96-05

An Implementation of Logical Analysis of Data

Endre Boros Peter L. Hammer Toshihide Ibaraki Alexander Kogan
Eddy Mayoraz Ilya Muchnik

July 1996

revised in 98-08

published in
IEEE Trans. on Knowledge and Data Engineering **12**–1 (2000) 292–306

Abstract. The paper describes a new, logic-based methodology for analyzing observations. The key features of the Logical Analysis of Data (LAD) are the discovery of minimal sets of features necessary for explaining all observations, and the detection of hidden patterns in the data capable of distinguishing observations describing "positive" outcome events from "negative" outcome events. Combinations of such patterns are used for developing general classification procedures. An implementation of this methodology is described in the paper along with the results of numerical experiments demonstrating the classification performance of LAD in comparison with the reported results of other procedures. In the final section we describe three pilot studies on applications of LAD to oil exploration, psychometric testing, and the analysis of developments in the Chinese transitional economy. These pilot studies demonstrate not only the classification power of LAD, but also its flexibility and capability to provide solutions to various case-dependent problems.

Acknowledgements: The partial support of ONR (Grants N00014-92-J1375 and N00014-92-J4083) is gratefully acknowledged.

The problems analyzed by LAD bear resemblance to problems encountered in various other areas, e.g. statistics, pattern recognition, clustering, classification, machine learning, neural networks, etc.

The LAD techniques were first proposed in [6, 10] for the case of binary data. This paper extends the Boolean approach to the non-binary case. It also describes an implementation of this methodology, presents the results of numerical experiments, and reports on several pilot studies.

One of the major goals of LAD is to classify new observations in a way consistent with past classifications. The available information consists of an archive of previous observations. Each observation consists of a vector of attribute values and the outcome of it. Although the LAD approach can be applied to datasets with numerical outcomes, as shown in Section 9.3, many of the well studied classification problems involve binary outcomes.

As typical examples of classification problems we mention the following:

- determination of the creditworthiness of loan applicants, based on information about their age, income, assets, profession, credit history, etc.;
- determining the benign or malignant character of breast tumors based on cell size, clump thickness, type of tumor, etc.;
- diagnosing psychiatric patients based on their answers to standard psychiatric tests;
- recognizing the presence or absence of oil in a reservoir based on electric, acoustic, and nuclear measurements.

A specific characteristic of LAD is the detection of logical patterns which distinguish observations in one class from all the other observations. A pattern characteristic for a specific class is a combination of attribute values (or sets of values) occurring together only in some observations from the considered class. For example, high readings given by several medical tests can be symptomatic for the presence of a certain disease.

The essence of LAD consists on the one hand in the detection of patterns, and on the other hand in the selection of groups of such patterns which collectively are capable of correctly classifying all known observations. An important feature of the LAD methodology is the possibility of using patterns in explaining the results of classification to human experts by standard formal reasoning.

This paper consists of three parts describing respectively an implementation of the LAD methodology, an extension of this methodology to the case of numerical data, and the results of computational and case studies involving both binary and numerical data. We introduce some basic notation and terminology in Section 2. We present various pattern generation procedures in Section 3. Section 4 describes how a classification rule (a theory) can be built from the generated patterns. In Section 5 we deal with non-binary input data, and describe a binary encoding of them. Section 6 describes how to reduce the size of the binary encoding of data. This reduction preserves the relevant information about the problem and reduces at the same time the computational difficulties at the pattern generation step. In Section 7 we describe some modifications of LAD needed to handle imperfect datasets, including classification or measurement errors and missing data.

Section 8 presents the results of numerical experiments with LAD on some datasets from the Irvine repository. It compares the performance of LAD with the best results we found in the literature. In Section 9 we describe in detail how LAD was applied in three pilot studies involving new datasets, demonstrating the use of LAD for decision support and data compression. The conclusions presented in Section 10 summarize the LAD methodology, emphasizing its essential features, including interpretability, efficiency, and flexibility.

1 Notation and Terminology

The input information in the problems to be studied in this paper consists of an archive of past observations denoted by S. Each observation (element of S) is an n-dimensional vector having as components the values of n attributes. Each observation is accompanied by its "classification", i.e. by the indication of the particular class (e.g. positive or negative) this observation belongs to. In the first part of the paper we shall assume that both the attributes and the classification are binary. We can think of the classification as being defined by a partition of S into two sets S^+ and S^- representing the positive and the negative observations respectively.

An archive (S^+, S^-) of the type described above can be naturally represented by a partially defined Boolean function ϕ, i.e. a mapping $S \rightarrow \{0,1\}$, where S is viewed as a subset of $\{0,1\}^n$. Any completely defined Boolean function (i.e. a mapping $\{0,1\}^n \rightarrow \{0,1\}$) which agrees with all the classifications in the archive will be called an *extension* of ϕ.

One of the ultimate goals of LAD is the determination of the extension $\hat{f}$ representing the unknown correct classification of all the vectors in the sample space. This goal is clearly unattainable in most real life situations, and is therefore replaced by a more realistic search for extensions that would approximate $\hat{f}$ as closely as possible according to various criteria. Such criteria may include membership in specific classes of Boolean functions, simplicity considerations, or other optimality criteria (see [6, 1]).

A common special class of Boolean functions frequently used for choosing an extension is the class of threshold (or linearly separable) functions in which the classification is decided by whether a weighted sum of the attributes does or does not exceed a certain threshold.

Frequently, the quest for a good approximation of $\hat{f}$ will be guided to a large extent by the principle "the simpler the better". In particular, we shall pay special attention to positive and negative patterns involving as few variables as possible.

Let us recall some basic Boolean terminology. A *literal* is either a Boolean variable or its negation. A *term* is a conjunction of literals. The *degree* of a term is the number of literals in it. A term T is said to *cover* a point $p \in \{0,1\}^n$ if $T(p) = 1$. The *characteristic term* of a point $p \in \{0,1\}^n$ is the unique term of degree n covering p. For example, the characteristic term of $(0,1,1,0)$ is $\overline{x}_1 x_2 x_3 \overline{x}_4$.

A term T is an *implicant* of a Boolean function f if $T(p) \leq f(p)$ for all $p \in \{0,1\}^n$. An implicant is called *prime* if it is minimal, i.e. any term obtained by dropping a literal from it is not an implicant.

Given a partially defined function ϕ specified by the pair (S^+, S^-), let us define the two extreme extensions of it, denoted by ϕ^+ and ϕ^-, by

$$\phi^+(p) = \begin{cases} 1 & \text{if } p \notin S^- \\ 0 & \text{if } p \in S^- \end{cases},$$

and

$$\phi^-(p) = \begin{cases} 1 & \text{if } p \in S^+ \\ 0 & \text{if } p \notin S^+ \end{cases}.$$

The prime implicants of ϕ^+ covering at least one point in S^+ are called *positive prime patterns* of the partially defined function ϕ. The *negative prime patterns* of ϕ are defined simply as the positive prime patterns of its complement $\overline{\phi}$ (i.e. the negative prime patterns of ϕ are the prime implicants of $\overline{\phi^-}$ covering at least one point in S^-).

Patterns play a key role in LAD, since they admit a clear interpretation by human experts. Indeed, a positive pattern can be viewed simply as a combination of values, taken by a (usually small) number of attributes, that has never appeared in any of the negative observations, but did appear in some of the positive observations. Therefore, the fact that a new observation is covered by a positive pattern can be viewed as an indication of the positive character of this. Negative patterns, in their turn, provide similar indication about the negative character of a new observation.

Consider for example the partially defined Boolean function given in Table 1. It can be checked that the positive prime patterns of this function are a_2 and $a_1 a_3$, while the negative prime patterns are $\overline{a}_1 \overline{a}_2$, $\overline{a}_1 a_3$, and $\overline{a}_2 \overline{a}_3$.

	a_1	a_2	a_3
	1	1	0
S^+	0	1	0
	1	0	1
	1	0	0
S^-	0	0	1
	0	0	0

Table 1: Partially-Defined Boolean Function

2 Pattern Generation

The symmetric definition of positive and of negative patterns leads to symmetric generation procedures. For the sake of brevity, we shall describe here only procedures for generating positive patterns. The generation of negative patterns proceeds in a similar way.

The most straightforward approach to pattern generation is based on the use of combinatorial enumeration techniques. In view of the existence of various possible measures of quality of any given pattern, it is important for the pattern generation procedure not to miss any of the "best" patterns. Pattern generation techniques can follow a "top-down" or a "bottom-up" approach.

The top-down approach starts by associating to every positive observation its characteristic term. Such a term is obviously a pattern, and it is possible that even after the removal of some literals the resulting term will remain a pattern. The top-down procedure systematically removes literals one-by-one until arriving to a prime pattern.

The bottom-up approach starts with terms of degree one that cover some positive observations. If such a term does not cover any negative observations, it is a pattern. Otherwise, literals are added to the term one by one as long as necessary, i.e. until generating a pattern.

Our pattern generation process is guided by two natural objectives: on the one hand we give preference to the generation of short patterns (simplicity principle), and on the other hand, we attempt to cover every positive observation by at least one pattern (comprehensiveness principle). This is achieved by a hybrid bottom-up – top-down approach which favors the bottom-up strategy. We start by using the bottom-up approach to generate all the patterns of very small degrees (usually up to 4 or 5), and use then a top-down approach to cover those positive observations (if any) that remained uncovered after the bottom-up step.

The bottom-up pattern generation is based on term enumeration. The number of terms of degree d over n Boolean variables is $2^d\binom{n}{d}$. Even for n fixed at a moderate value (say, for $n = 20$), this is a very rapidly growing function of d. Therefore, the term enumeration method used for pattern generation must be extremely selective.

The method used in this paper is a breadth-first enumerative technique which produces at each stage d all the positive prime patterns of degree d, as well as the list of the so-called "candidate" terms to be examined at stage $d+1$ of the algorithm. A *candidate term* is any term that covers at least one negative and at least one positive observation. The terms of degree d examined by the algorithm at stage d are all those from which one gets a candidate term of degree $d-1$ (generated at stage $d-1$) by eliminating any of its literals. The terms of degree d examined by the algorithm are then partitioned into the following three sets:

- $\mathcal{P}_d$, consisting of those terms which cover at least one positive and no negative observations;
- $\mathcal{C}_d$, consisting of those terms which cover at least one positive and at least one negative observation;
- $\mathcal{G}_d$, consisting of all the remaining terms.

It is easy to see that the set $\mathcal{P}_d$ consists of all positive prime patterns of degree d, and the set $\mathcal{C}_d$

```
input:          $B^+, B^- \subset \{0,1\}^n$   – Sets of positive and negative Boolean points.
                $D$                            – Maximal degree for patterns to be generated.
output:         $\mathcal{P}$                  – Set of prime patterns of degree up to $D$.
initialization: $\mathcal{P} := \emptyset$;
                $\mathcal{C}_0 := \{\emptyset\}$;   – contains the empty term
main loop:      for $d := 1$ to $D$ loop
                  if $d < D$ then $\mathcal{C}_d := \emptyset$; end if        – No need to create $\mathcal{C}_D$.
                  for $T$ through $\mathcal{C}_{d-1}$ loop
                    $p :=$ maximal index of literal in $T$;
                    for $s := p+1$ to $n$ loop                               – Try to add all possible
                      for $l_{\mathrm{new}}$ through $\{l_s, \overline{l_s}\}$ loop   –   literals with large indices.
                        $T' := T \bigwedge l_{\mathrm{new}}$;
                        for $i := 1$ to $d-1$ loop                           – Check whether all
                          $T'' := T'$ with $i^{\mathrm{th}}$ literal dropped;  –   subterms are in $\mathcal{C}_{d-1}$.
                          if $T'' \notin \mathcal{C}_{d-1}$ then goto $\diamondsuit$; end if
                        end loop
                        if $1 \in T'(B^+)$ then                              – One positive point is covered.
                          if $1 \notin T'(B^-)$ then                         – No negative point is covered.
                            $\mathcal{P} := \mathcal{P} \cup \{T'\}$;        – $T'$ is a prime pattern.
                          elseif $d < D$ then                                – One negative point is covered.
                            $\mathcal{C}_d := \mathcal{C}_d \cup \{T'\}$;    – $T'$ is a candidate pattern.
                          end if
                        end if
                        $\diamondsuit$
                      end loop
                    end loop
                  end loop
                end loop
```

Figure 1: Algorithm for the enumeration of prime patterns up to a given degree.

consists of all candidate terms of degree d. We note that the set $\mathcal{G}_d$ is eliminated from any further consideration.

Additional reductions in the volume of computations are obtained by examining the terms of $\mathcal{C}_d$ in the lexicographic order induced by the linear order

$$x_1 \prec \overline{x}_1 \prec x_2 \prec \overline{x}_2 \prec \ldots$$

of the literals. Since it is sufficient to generate each term only once, the terms of degree $d+1$ are generated from $\mathcal{C}_d$ by adding in all possible ways to a term $T \in \mathcal{C}_d$ a literal which is larger (in this order) than any literal in T. Indeed, let the indices of the literals in T be $i_1 < i_2 < \ldots < i_d$. Suppose that a term T' is obtained by adding to T a literal of index $i < i_d$. Let T'' be the term whose literals have the indices $i_1, i_2, \ldots, i, \ldots, i_{d-1}$. Clearly, T' can also be obtained by adding to T'' the literal of index i_d. If $T'' \notin \mathcal{C}_d$, T' does not have to be examined, because the term T' is then neither a prime pattern, nor a candidate term. If $T'' \in \mathcal{C}_d$, then T' was already considered, because T'' is lexicographically smaller than T. This algorithm is summarized in the pseudo-code of Figure 1.

The efficient implementation of this algorithm requires some special data structure to store the sets $\mathcal{C}_d$. The data structure has to allow to traverse $\mathcal{C}_d$ in a lexicographic order, to add an element efficiently, and, even more importantly, to permit rapid access to a term T'' having $d-1$ literals in common with a given term T. The choice of data structure is complicated by the size of $\mathcal{C}_d$, which can easily have hundreds of thousands of elements. The data structure used in the implementation of this algorithm is described in [13].

The generation of patterns in our program starts with the bottom-up stage, and proceeds until all patterns of degree up to a certain (problem dependent) D are generated. At that time, those observations in the training set which are not covered by any patterns generated so far, are extracted and used in a top-down procedure to generate some additional patterns that cover them. The set of patterns can then be reduced by eliminating the "redundant" ones. A pattern is called *redundant* if all the observations covered by it are also covered by another pattern, which in addition covers some other observations. Our computational experiments show that this simple procedure produces drastic reductions in the number of patterns.

In many cases it is important to restrict attention to prime patterns with special properties. A typical example occurs when reliability considerations require the usage of only those prime positive (negative) patterns that cover at least a certain number of positive (negative) observation points. Additionally, only those prime positive (negative) patterns are to be considered whose appropriately defined distance to the set of negative (positive) observations is "large". The distance from a term to a set of points can be defined as the average of Hamming distances[1] between the term and individual points. For example, for a partially defined Boolean function in Table 1 the average distance from a_2 to S^- is 1, while the average distance from $\overline{a}_1\overline{a}_2$ to S^+ is 4/3. One of the advantages of the algorithm described in Figure 1 is the ease with which it can be modified to incorporate such additional requirements.

A particularly important property of certain datasets, that has to be taken into account in pattern generation, consists in the presence of some so-called *monotone* variables. Given a Boolean function f, the variable x_i is called *positive* in f if for any Boolean vector $x^* \in \{0,1\}^n$,

$$f(x_1^*, \ldots, x_{i-1}^*, 0, x_{i+1}^*, \ldots, x_n^*) \leq f(x_1^*, \ldots, x_{i-1}^*, 1, x_{i+1}^*, \ldots, x_n^*).$$

Similarly, the variable x_i is called *negative* in f if for any Boolean vector $x^* \in \{0,1\}^n$,

$$f(x_1^*, \ldots, x_{i-1}^*, 0, x_{i+1}^*, \ldots, x_n^*) \geq f(x_1^*, \ldots, x_{i-1}^*, 1, x_{i+1}^*, \ldots, x_n^*).$$

Finally, the variable x_i is called *monotone* in f if it is either positive or negative in f.

Monotone variables appear frequently in real life problems, being associated with attributes having a well-defined impact on the outcome. For example, the fact that a person is employed can only help to qualify for a credit card (positive attribute), while the fact that someone is a smoker can only make it harder to qualify for a low cost life insurance (negative attribute).

It is well known that the prime implicants of a Boolean function do not contain the negations of positive variables, or the non-negated forms of negative variables. Consequently, the a priori knowledge of the monotone character of some variables immediately disqualifies certain patterns from consideration. The pattern generation procedure described in this section can be easily modified to prevent the generation of unwanted patterns. This modification may reduce substantially the number of patterns generated by our procedure.

After the completion of the pattern generation stage, the program proceeds to the theory formation stage described in the next section.

3 Theory Formation

Every positive (negative) observation point is covered by at least one positive (negative) pattern, and is not covered by any negative (positive) patterns that have been generated. Moreover, individual patterns can also provide indications about the positive (negative) character of new observation points, and it can therefore be expected that an adequately chosen collection of patterns can be used for constructing a general classification rule. This rule is actually an extention of a partially defined Boolean function, and will be called below a *theory*.

A good classification rule should capture all the significant aspects of the phenomenon. Since patterns capture dependencies inherent to the observations in the archive, it is natural to make use of as many patterns as possible in building a theory. On the other hand, there are significant difficulties in using too many patterns to build a theory. First of all, the number of patterns that can be generated may be too large to allow the effective utilization of all of them. Also, some of the patterns may occur only in rare cases, and they are redundant when more typical patterns are taken into account.

[1]The Hamming distance between a term and a point is the number of variables in which they conflict.

Let us denote by $P_1, \ldots, P_r$ the generated positive patterns and by $N_1, \ldots, N_s$ the generated negative patterns. The proposed method of building a theory consists of defining a weighted sum of positive and negative patterns and classifying new observations according to the value of this weighted sum:

$$\Delta = \sum_{k=1}^{r} w_k^+ P_k + \sum_{l=1}^{s} w_l^- N_l.$$

Such a weighted sum will be called hereafter a *discriminant*. The weights of the patterns are chosen in such a way that large positive (negative) values of the discriminant will be indicative of the positive (negative) character of the new observation. Low absolute values of the discriminant indicate that either there is not enough evidence about the positive/negative character of the observation, or that the available evidence is conflicting. Therefore, new observations will be classified only when the corresponding absolute values of the discriminant exceed a problem-dependent threshold.

By using non-negative weights for all positive patterns ($w_k^+ \geq 0$) and non-positive weights for all negative patterns ($w_l^- \leq 0$), and by assigning non-zero weights to a sufficiently large number of patterns, we can make sure that the discriminant is positive (negative) on all the positive (negative) observations in the archive.

Before discussing ways of assigning weights to patterns, we shall first describe how to select a subset of the patterns generated so far. This subset should on the one hand be of reasonable size, but on the other hand it should allow us to distinguish between the positive and the negative observations. We describe below the selection process for the case of positive patterns, since the case of negative patterns is similar.

Let us assign to each of the generated positive patterns $P_1, P_2, \ldots, P_r$ binary $(0-1)$ variables $y_1, y_2, \ldots, y_r$ with the convention that $y_k = 1$ (0) means that P_k is (is not) in the selected subset. Let us also define $a_{jk} = 1$ if the positive observation point p_j is covered by the pattern P_k, and $a_{jk} = 0$ otherwise. It is clear that, in order to distinguish p_j from the negative observation points, at least one of the positive patterns covering it must be selected, i.e.

$$\sum_{k=1}^{r} a_{jk} y_k \geq 1. \tag{1}$$

In order to distinguish all the positive points from all the negative ones, the vector $(y_1, y_2, \ldots, y_r)$ characterizing the selected subset will have to satisfy the covering constraints (1) for all positive observation points p_j. In order to produce a small subset of patterns satisfying these requirements we shall solve the following *set covering problem*:

$$\begin{array}{rll} \min & \sum_{k=1}^{r} y_k & \\ \text{s.t.} & \sum_{k=1}^{r} a_{jk} y_k \geq 1, & \text{for all positive } p_j \\ & y_k \in \{0,1\}, & k = 1, \ldots, r. \end{array} \tag{2}$$

In order to complete this procedure, we have to clarify some details.

- To increase the discriminating power of the selected subset of patterns, we may require that each positive point be covered by several patterns; this can be achieved by increasing the right-hand side of the constraints (1).
- In order to give preference to patterns possessing some special properties (e.g. low degree, high covering power, etc.) the objective function $\sum_{k=1}^{r} y_k$ can be replaced by a weighted sum $\sum_{k=1}^{r} c_k y_k$ with appropriately chosen weights c_k. For example, c_k can be chosen equal to the degree d_k of the corresponding pattern (or even to d_k^2).

The set covering problem of the type described above is a typical, frequently occurring problem in operations research, having numerous applications in a variety of areas. This problem is well studied, and numerous exact algorithms and heuristic approximation methods are known for its solution.

Although this problem is NP-complete, reasonably good solutions can be easily obtained using, for example, one of the simple greedy heuristics. Typically, such a procedure selects patterns one-by-one, until all points are covered. At each step the algorithm selects a pattern which covers as many of the not yet covered observation points as possible.

By solving the set covering problem (2), a subset of positive patterns is selected to be included in the discriminant. A subset of negative patterns to be also included in the discriminant is selected in a similar way. Since all the positive (negative) patterns will have positive (negative) weights in the discriminant, what remains to be done is to determine the absolute values of the weights. The simplest possible choice is to define all $|w_k| = 1$, assigning thus equal importance to all the patterns. On the other hand, the number of observation points q_k covered by a pattern P_k can be viewed as an indication of its relative importance justifying the choice $|w_k| = q_k$. The relative importance of patterns can be emphasized even stronger by choosing $|w_k| = q_k^2$, or q_k^3, or 2^{q_k}. These choices were implemented in the program and proved to be useful in numerical experiments. This approach can be generalized by choosing weights on the basis of appropriately defined distances from a pattern to the sets of positive (S^+) and negative (S^-) observations in the training set. Another reasonable point of view emphasizing the role of simple (i.e. short) patterns defines $|w_k| = \frac{1}{d_k}$, where d_k is the degree of the pattern P_k.

In view of the possible disparity between the numbers of positive and of negative patterns in the discriminant, the weights may have to be normalized by a constant factor assuring that

$$\sum_{k=1}^{r} w_k^+ = -\sum_{l=1}^{s} w_l^- = 1. \tag{3}$$

Let us consider again the partially defined function in Table 1. It can be seen immediately that the set covering problem for the positive patterns shows that both a_2 and a_1a_3 have to be included in the discriminant. On the other hand, the set covering problem for the negative patterns shows that only $\overline{a}_1\overline{a}_2$ and $\overline{a}_2\overline{a}_3$ have to be included in the discriminant. Taking into account that each of these two negative patterns covers two negative points, while the coverage of a_2 is 2, and the coverage of a_1a_3 is 1, we can construct the following discriminant with pattern weights chosen to be their coverage normalized as in (3):

$$\Delta = \frac{2}{3} * a_2 + \frac{1}{3} * a_1a_3 - \frac{1}{2} * \overline{a}_1\overline{a}_2 - \frac{1}{2} * \overline{a}_2\overline{a}_3.$$

A more sophisticated approach to weight selection aiming to increase as much as possible the separating power of the discriminant, is based on the use of linear programming. The weights are then determined by solving the following linear program:

$$\begin{array}{lll} \max & g_p + g_n & \\ \text{s.t.} & \sum_{k=1}^{r} w_k^+ P_k(p) \geq g_p, & \text{for all positive } p \\ & \sum_{l=1}^{s} w_l^- N_l(p) \leq -g_n, & \text{for all negative } p \\ & \sum_{k=1}^{r} w_k^+ - \sum_{l=1}^{s} w_l^- = 1 & \\ & w_k^+ \geq 0, & k = 1, \dots, r \\ & w_l^- \leq 0, & l = 1, \dots, s. \end{array} \tag{4}$$

where $P_k(p)$ ($N_l(p)$) is 1 if P_k (N_l) covers p, and is 0 otherwise.

The normalization of weights constraint (3) can of course be added to the linear program. Also, the upper and lower bounds g_p and g_n can be assumed to be equal in a simplified model.

The following sections of the paper will describe logical analysis procedures for problems with non-binary data. It will be seen that the proposed way of handling problems with numerical data is to reduce them to binary ones. Since the resulting binary problems may be of large size, the efficiency of binary procedures is crucial. The numerical experiments and the pilot studies described in Sections 8 and 9 indicate clearly the power of the procedures described above.

4 Data Binarization

All the data in the problems studied in the previous sections were binary $(0, 1)$. Many real life problems appear naturally in this form. Many more problems, however, use more complex data, especially *numerical* ones, as well as *nominal* ones (e.g. color, shape, type, etc.). To make such problems amenable to the techniques of LAD and to possibly reveal the underlying logical structure of the phenomenon, the problems have to be transformed into a binary format. The subject of this section is the description of such a binary encoding (or simply, binarization) procedure.

The simplest non-binary attributes are the so-called "nominal" (or descriptive) ones. A typical nominal attribute is "shape" whose values can be "round", "triangular", "rectangular", and so on. The binarization of such an attribute is accomplished in a straightforward way by associating with each value v_s of the attribute x a Boolean variable $b(x, v_s)$ such that[2]

$$b(x, v_s) = \begin{cases} 1 & \text{if } x = v_s \\ 0 & \text{otherwise} \end{cases} .$$

In the vast majority of cases values of many of the attributes are ordered. This happens, for example, when the attributes are numerical (temperature, weight, etc.). This can also happen when attributes take nominal, but comparable values, like "frequently", "rarely", "never".[3]

Binarization of ordered attributes is a common practice in many areas of human activity. For example, body temperature, blood pressure, pulse rate, and many other medical parameters are termed "normal" or "abnormal" depending on whether they are inside or outside a certain interval. In many other examples a parameter (e.g. cholesterol level) is termed "normal" or "abnormal" depending on whether it is above or below a certain threshold. In all these examples a binarization is implicitly accomplished by comparing the values of numerical attributes with certain standard "cut-points" (critical values).

We shall follow this approach in our binarization procedure, and shall associate for this purpose two types of Boolean variables to the the given numerical ones.

Boolean variables of the first type, called *level variables*, are associated with every cut-point, and show whether the value of the original attribute is above or below the cut-point. In other words, for every attribute x and cut-point t we shall introduce a Boolean variable $b(x, t)$ such that

$$b(x, t) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases} .$$

Boolean variables of the second type, called *interval variables*, are associated with every pair of cut-points, and show whether the value of the original attribute is inside or outside the interval having the two cut-points as its end-points. In other words, for every attribute x and pair of cut-points t', t'' $(t' < t'')$ we shall introduce a Boolean variable $b(x, t', t'')$ such that

$$b(x, t', t'') = \begin{cases} 1 & \text{if } t' \leq x < t'' \\ 0 & \text{otherwise} \end{cases} .$$

Since the number of observation points in the training set is finite, each ordered attribute x takes only a finite number of different values in the training set. Let these values be

$$v_1 < v_2 < \ldots < v_q.$$

Since two cut-points t' and t'' such that

$$v_{s-1} < t', t'' \leq v_s$$

[2]Notice that in the special case of nominal attributes which are binary, i.e. take only two values, these values are renamed as 0 and 1, and no additional Boolean variables are introduced.

[3]Remark that the same attribute "color" with values "green", "yellow", "red" may be nominal in some situations (e.g. when describing an apple), or ordered in other situations (e.g. when describing a traffic light).

produce identical Boolean variables (on the training set), it is not necessary to consider any cut-points outside the set $\{v_1, v_2, \ldots, v_q\}$. In order to make this binarization procedure more robust with respect to measurement errors (in the case of numerical attributes) we will use the cut-points $t_s = \frac{1}{2}(v_{s-1}+v_s)$ instead of the cut-points v_s. Let us call a cut-point *essential* if there exist both a true and a false observation points in the training set, such that one of them has $x = v_s$ while the other one has $x = v_{s-1}$. Clearly, it is sufficient to use only essential cut-points in the binarization procedure.

A simple example of this binarization procedure is shown in Table 2, where the first and the fourth attributes are numerical, while the second and the third ones are nominal. Here we present all the level and interval Boolean variables corresponding to all the eseential cut-points. It will be seen later that only a small subset of these variables will be actually needed.

	x_1	x_2	x_3	x_4
S^+	1	green	yes	31
	4	blue	no	29
	2	blue	yes	20
	4	red	no	22
S^-	3	red	yes	20
	2	green	no	14
	4	green	no	7

Original Table (A)

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}
a	0	0	0	1	0	0	1	1	1	0	0	0	0
b	1	1	1	0	1	0	0	1	1	0	0	0	0
c	1	0	0	0	1	0	1	1	0	1	1	0	1
d	1	1	1	0	0	1	0	1	1	0	0	0	0
e	1	1	0	0	0	1	1	1	0	0	1	1	1
f	1	0	0	1	0	0	0	0	0	1	1	0	0
g	1	1	1	1	0	0	0	0	0	0	0	0	0

Binarized Table (B)

Table 2: Binarization
(B) is obtained from (A) using the following level variables:

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
$x_1 \geq 1.5$	$x_1 \geq 2.5$	$x_1 \geq 3.5$	$x_2 = green$	$x_2 = blue$	$x_2 = red$	$x_3 = yes$	$x_4 \geq 17$	$x_4 \geq 21$

and the following interval variables:

b_{10}	b_{11}	b_{12}	b_{13}
$1.5 \leq x_1 < 2.5$	$1.5 \leq x_1 < 3.5$	$2.5 \leq x_1 < 3.5$	$17 \leq x_4 < 21$

We have described above a binary encoding of our data without paying attention to the number of Boolean variables created in this way. In reality, many of the Boolean variables introduced by this procedure may not be needed to explain the phenomenon. Such a size reduction is not only useful but actually necessary in order to prevent insurmountable computational difficulties at the pattern generation stage. In the next section we shall describe a systematic procedure to accomplish this size reduction.

5 Support Sets

We recall that the archive of observation points is *partitioned* into a set of true (S^+) and a set of false (S^-) points, i.e. we assume that no observation point is simultaneously true and false. This basic property has to be maintained by any correct binary encoding of the archive. Clearly, the binary encoding described in the previous section does preserve this property. The subject of this section is to reduce the size of a binary archive by eliminating as many redundant attributes as possible, while preserving the basic property mentioned above.

A set of binary attributes is called a *support set* if the archive obtained by the elimination of all the other attributes will remain "contradiction-free" (i.e. it will not contain simultaneously true and false observations). A support set is called *irredundant* if no proper subset of it is a support set. For example, the first four attributes in Table 2-B form an irredundant support set.

By examining the true point a and the false point e in Table 2-B we see that they differ only in the attributes $b_1, b_2, b_4, b_6, b_9, b_{11}, b_{12}, b_{13}$. Therefore, if all these eight attributes were omitted, the resulting archive would have a contradiction (the resulting observations a' and e' would be identical). If we associate a Boolean variable y_i to the attribute b_i in such a way that $y_i = 1$ means that the attribute b_i is retained in a support set, and $y_i = 0$ means that it is omitted from it, then

$$y_1 + y_2 + y_4 + y_6 + y_9 + y_{11} + y_{12} + y_{13} \geq 1$$

for every support set. Similar inequalities can be associated with every pair of true and false observations.

More generally, given an archive with Boolean attributes $b_1, \ldots, b_q$ and variables $y_1, \ldots, y_q$ associated to them as above, we shall define for every pair of true and false vectors p' and p'' the subset $I(p', p'')$ of those indices where p' and p'' differ (e.g. $I(a, e) = \{1, 2, 4, 6, 9, 11, 12, 13\}$ in the example above). It is easy to see that that $(y_1, \ldots, y_q)$ is the characteristic vector of a support set if and only if it satisfies the following system of inequalities:

$$\sum_{i \in I(p', p'')} y_i \geq 1 \qquad \forall p' \in S^+, \forall p'' \in S^-. \tag{5}$$

In the case of the example in Table 2-B, this system is the following:

$$\begin{array}{c}
\text{(a,e)}\\ \text{(a,f)}\\ \text{(a,g)}\\ \text{(b,e)}\\ \text{(b,f)}\\ \text{(b,g)}\\ \text{(c,e)}\\ \text{(c,f)}\\ \text{(c,g)}\\ \text{(d,e)}\\ \text{(d,f)}\\ \text{(d,g)}
\end{array}
\begin{pmatrix}
1&1&0&1&0&1&0&0&1&0&1&1&1\\
1&0&0&0&0&0&1&1&1&1&1&0&0\\
1&1&1&0&0&0&1&1&1&0&0&0&0\\
0&0&1&0&1&1&1&0&1&0&1&1&1\\
0&1&1&1&1&0&0&1&1&1&1&0&0\\
0&0&0&1&1&0&0&1&1&0&0&0&0\\
0&1&0&0&1&1&0&0&0&1&0&1&0\\
0&0&0&1&1&0&1&1&0&0&0&0&1\\
0&1&1&1&1&0&1&1&0&1&1&0&1\\
0&0&1&0&0&0&1&0&1&0&1&1&1\\
0&1&1&1&0&1&0&1&1&1&1&0&0\\
0&0&0&1&0&1&0&1&1&0&0&0&0
\end{pmatrix}
\begin{pmatrix}
y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6\\ y_7\\ y_8\\ y_9\\ y_{10}\\ y_{11}\\ y_{12}\\ y_{13}
\end{pmatrix}
\geq
\begin{pmatrix}
1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1
\end{pmatrix}, \tag{6}$$

where each row of the matrix corresponds to a pair of true/false observations (shown next to each row), and where each column represents a Boolean attribute. The system of inequalities (5) has usually an exponential number of solutions. The irredundant support sets correspond to componentwise minimal $0-1$ vectors satisfying this system. A smallest support set can be obtained by solving the set covering problem

$$\begin{array}{ll}
\min & \sum_{i=1}^{q} y_i \\
\text{s.t.} & \sum_{i \in I(p',p'')} y_i \geq 1, \quad \forall p' \in S^+, \forall p'' \in S^- \\
 & y_i \in \{0,1\}, \qquad\quad i = 1, \ldots, q.
\end{array} \tag{7}$$

It can be easily checked that $\{7, 9, 10\}$ is an irredundant support set in the example above. The resulting partially defined Boolean function is exactly the function presented earlier in Table 1. Now we can interpret its patterns as follows:

Positive Patterns:

$x_4 \geq 21$,

$x_3 = yes$ and $1.5 \leq x_1 < 2.5$.

Negative Patterns:

$x_3 = no$ and $x_4 < 21$,

$x_3 = no$ and $1.5 \leq x_1 < 2.5$,

$x_4 < 21$ and ($x_1 < 1.5$ or $x_1 \geq 2.5$).

An important modification of the set covering problem (7) has to be carried out in the case of datasets involving monotone attributes. The support set should be chosen in such a way that the resulting partially defined function has an extension in which all the level Boolean variables corresponding to original positive (negative) attributes remain positive (negative); no similar restriction applies to the interval Boolean variables. In order to accomplish this, some constraints of the set covering problem (7) have to be strengthened.

To illustrate this point, let us assume that in the previous example the attribute x_3 (and therefore, the variable b_7) is negative. In that case, the comparison of the true point a and the false point f, which differ in b_1, b_7, b_8, b_9, b_{10}, and b_{11}, generates the inequality

$$y_1 + y_7 + y_8 + y_9 + y_{10} + y_{11} \geq 1.$$

However, if $y_1 = y_8 = y_9 = y_{10} = y_{11} = 0$ and $y_7 = 1$, then the variable b_7 cannot be negative in any extension of the partially defined Boolean function for any resulting support set, since the reduced true point a and false point f will differ only in b_7, where the value of a is 1 and that of f is 0. Therefore, in the set covering problem the above inequality has to be strehgthened to

$$y_1 + y_8 + y_9 + y_{10} + y_{11} \geq 1.$$

More generally, the set $I(p', p'')$ corresponding to any pair consisting of a positive vector p' and a negative vector p'' should be reduced by eliminating those indices i for which b_i is positive, $p'_i = 0$, and $p''_i = 1$, or for which b_i is negative, $p'_i = 1$, and $p''_i = 0$. Since the constraints of the set covering problem resulting in this way become stronger, the minimum value of the objective function will be at least as high as before.

Let us remark here that the main reason to eliminate some of the variables and use a smaller support set is to reduce the computational complexity of the pattern and theory generation. Intuitively it is clear that the smaller the chosen support set, the less information we keep, and therefore the less classification power we have. It is therefore important to keep a healthy balance between the discriminating power and the computational cost of later steps. It implies that for our purposes an approximate solution to (7) may be as good as (or even better than) the optimal solution. Not only that the discriminating power of the generated support set is not expected to be smaller, but also the computational cost of an approximation algorithm is much smaller than that of an exact method.

Furthermore, in order to increase and/or assure a certain minimum level of discriminating power of the generated support sets we shall introduce modifications, involving the changes of some of the parameters of the set covering problem (7). We shall describe below three such changes, involving

1. the right-hand sides of the constraints,
2. the coefficients of the objective function, and
3. the coefficients of the left-hand sides of the constraints.

1) The goal of the first modification is to assure that the true and the false points can be distinguished in more than one attribute. In order to accomplish this objective, the constant 1 appearing on the right-hand sides of the inequalities (5) should be replaced by a higher value, say μ:

$$\sum_{i \in I(p', p'')} y_i \geq \mu \qquad \forall p' \in S^+, \forall p'' \in S^-. \tag{8}$$

The choice of μ should be guided by balancing two conflicting objectives: on the one hand, each pair consisting of a true and a false point should be separated in several attributes, while on the other hand, the selected support set should remain of a reasonable size. Also, in order to maintain the feasibility of the system, the values of μ cannot exceed a certain bound (the Hamming distance between the set of positive and the set of negative points).

2) A further improvement of the quality of the support set can be achieved by estimating the discriminating power of the individual attributes and using these estimates as weights in the objective function of the weighted set covering problem:

$$\begin{array}{lll} \min & \sum_{i=1}^{q} u_i y_i & \\ \text{s.t.} & \sum_{i \in I(p',p'')} y_i \geq \mu, & \forall p' \in S^+, \forall p'' \in S^- \\ & y_i \in \{0,1\}, & i = 1, \ldots, q. \end{array} \tag{9}$$

There are numerous ways of choosing values for the coefficients u_i. In particular, it seems reasonable to use various statistical measures like entropy, variance, Fisher's exact statistic, etc.

3) A third way of modifying our procedure for the selection of a support set consists in replacing the sums of variables appearing on the left-hand sides of the constraints (5) with weighted sums

$$\sum_{i \in I(p',p'')} \lambda(i;p',p'') y_i \geq \mu \qquad \forall p' \in S^+, \forall p'' \in S^-. \tag{10}$$

The coefficient $\lambda(i;p',p'')$ "measures" how well the cut-point t_i separates the pair (p',p'') of true and false observations. One possible way of defining $\lambda(i;p',p'')$ is to use the distances from this cut-point to the corresponding attribute values of p' and p''. More precisely, $\lambda(i;p',p'')$ can be chosen to be equal to the minimum of the two distances above, normalized by the spread of the corresponding attribute.

In view of the three modifications described above, the problem of finding a good support set has the following general form:

While this problem is a typical integer programming problem, and can be solved by any of the numerous methods available in the operations research literature, we shall describe below a simple "greedy" method for its approximate solution. There are two reasons for using an approximate technique. First, any exact solution method may require an excessive amount of computation, while greedy methods are very fast. Second, our experience seems to indicate that the classification performance of LAD does not change perceptibly when using support sets generated by this greedy procedure.

$$\begin{array}{lll} \min & \sum_{i=1}^{q} u_i y_i & \\ \text{s.t.} & \sum_{i=1}^{q} c_{ji} y_i \geq \mu_j, & j = 1, \ldots, m \\ & y_i \in \{0,1\}, & i = 1, \ldots, q, \end{array} \tag{11}$$

where all the coefficients are non-negative.

```
input:          ${u_i}$      – q-vector of objective function coefficients
                ${c_{ji}}$   – m×q-matrix of constraints coefficients
                ${\mu_j}$    – m-vector of right-hand side coefficients
output:         ${y_i}$      – characteristic q-vector of support set
initialization: $\boldsymbol{y} := \boldsymbol{0}$;
                $\boldsymbol{s} \in I\!R^m$; $\boldsymbol{s} := \boldsymbol{0}$;
main loop:      while $\boldsymbol{s} \not\geq \boldsymbol{\mu}$ loop
                    choose $i^* \in \{i = 1, \ldots, q : y_i = 0\}$
                        to maximize $\frac{1}{u_i} \sum_{j:\mu_j - s_j > 0} \min\{1, \frac{c_{ji}}{\mu_j - s_j}\}$;
                    $y_{i^*} := 1$;
                    $s_j := s_j + c_{ji^*}, j = 1, \ldots, m$;
                end loop
```

Figure 2: Greedy heuristic for the solution of problem (11).

The greedy method is an iterative procedure that builds a support set by adding variables to it one by one. The method starts with the empty set, and at each step it evaluates the potential incremental contribution of each variable to the satisfaction of the constraints, weighted against the cost of the variable in the objective function. A variable having the maximal evaluation is then added to the set. This process is repeated untill all the constraints are satisfied. The pseudo-code for the algorithm is shown in Figure 2. Various heuristic solutions, similar to the algorithm presented in Figure 2, are known for the generalized set covering problem (11) (see for example [17]).

At the end of the process the resulting support set is checked for redundant variables, i.e. variables whose removal from the support set will not violate the constraints (10). If such variables are found, they are removed one by one.

6 Imperfect Data Sets

We shall describe in this section three common imperfections of data sets, and some simple modifications implemented in our program in order to handle these problems. The most common imperfections of the data sets are

1. classification errors,
2. missing attribute values, and
3. small errors in the measurement of values of numerical attributes.

6.1 Classification Errors

Let us first consider the case of a negative observation point erroneously classified as a positive one. Assuming that the vast majority of observations are correctly classified, this situation will manifest itself in this erroneously positive point not being covered by any positive pattern of small degree. If the number of points in this situation is significant, some longer patterns will be generated to cover them. If the number of such points is very small, their classification may be considered erroneous, and the points may be omitted from further consideration.

Let us now consider the case of a positive observation point erroneously classified as negative. This situation will manifest itself during the pattern generation process when some candidate terms covering a large number of positive points turn out to also cover some isolated negative points. Since these negative points are likely to be misclassified, they will be disregarded in the evaluation of the candidate terms. The precise way of implementing this idea is to consider a candidate term to be a positive pattern if the ratio of the negative and the positive points covered by it does not exceed

$$\tau\frac{|S^-|}{|S^+|},$$

where $|S^-|$ and $|S^+|$ represent the number of the positive and the negative points, and τ is a problem dependent parameter. In our numerical experiments the best results were obtained when τ was chosen between 0 and 0.2 depending on the type of problem.

6.2 Missing Attribute Values

In the case of missing data we can view Boolean attributes as taking in fact three values: 0, 1, and $*$, where $*$ signifies the absence of information about the value of the attribute. Missing values influence various stages of our procedure. To start with, if the value of an original attribute is missing, then the values of all the corresponding binary variables are considered missing. At the support set construction stage, the unavailability of the value of a variable prevents this variable from being used to distinguish the corresponding observation point from other points. Consequently, in the set covering problem (11) the constraint corresponding to a pair consisting of a positive and a negative observation points should only involve those variables whose values are known in both points of the pair (i.e. the coefficients corresponding to the variables with missing values should be 0).

The existence of $*$-s necessitates the modification of the concept of covering an observation point by a term. We shall say that a term covers ***weakly*** a point if there exists an assignment of $0-1$ values to the $*$-s in this observation point such that the resulting $0-1$ point is covered by the term. Similarly, we shall say that a term covers ***strongly*** a point if for any assignment of $0-1$ values to the $*$-s in this observation point the resulting $0-1$ point is covered by the term.

These modifications in the concept of covering are introduced to make it possible to redefine the concept of a pattern in the case of missing attribute values. The aim of this redefinition is to guarantee the robustness of the patterns generated by our procedure. The robustness is interpreted as the property of a term to be a pattern in the original sense for all $0-1$ assignments to the $*$-s in the dataset. We shall call a term a ***robust positive (negative) pattern*** if this term covers strongly at least one positive (negative) observation point, and does not cover even weakly any negative (positive) observation point. Clearly, if the dataset does not have any missing attribute values, the definition of a robust pattern reduces to the original definition of a pattern.

The pattern generation procedure described in this paper can be easily modified to generate robust patterns in the case of missing attribute values, and this modification is implemented in our program. Several of the numerical experiments reported in Section 8 (e.g. Australian Credit Card, Congressional Voting) could not have been run without this modification.

A theoretical development of the logical analysis of datasets with missing attribute values is presented in [3, 2].

6.3 Measurement Errors

Numerous datasets contain small inaccuracies in the values of numerical attributes due to imprecise instrumentation or human errors. While such inaccuracies in themselves are unlikely to lead to ambiguities, their effect can be magnified in the binarization process. Indeed, if the numerical value of an attribute is close to a cut-point, then the value of the binarized variable can depend strongly on minor inaccuracies. Obviously, such a situation may lead to substantial distortions. In order to avoid distorted binarizations, the value of a binary variable is safer to be considered missing (and to be represented, as above, by $*$) if the numerical value of the attribute is very close to the corresponding cut-point. Formally, for every attribute x and cut-point t we shall introduce a level variable $b(x,t)$ such that

$$b(x,t) = \begin{cases} 1 & \text{if } x \geq t + \sigma_x \\ 0 & \text{if } x < t - \sigma_x \\ * & \text{if } t - \sigma_x \leq x < t + \sigma_x \end{cases},$$

where σ_x is an attribute-dependent measure of accuracy; clearly, a similar modification has to be done also for the interval variables.

Obviously, this modification of the binarization procedure may produce binary problems with missing attribute values even when no data are missing in the original formulation of the problem. The resulting problem with (possibly) missing attribute values is, of course, handled as in the previous subsection.

7 Numerical Experiments with the Irvine Repository

The logical analysis of data can be useful in achieving a large variety of objectives. These objectives are application-dependent and include, among others, classification, detection of meaningful logical dependencies, data compression, decision support, etc.

This section reports on numerical experiments with the use of LAD for classification. The choice of this application was motivated by the public availability of datasets which make comparisons with other methods possible. The next section will report on the applicability of LAD for the detection of logical dependencies, data compression, and decision support, using several new real life problems studied at RUTCOR.

For our experiments with classification we used some datasets from the Repository of Machine Learning Databases and Domain Theories maintained by the University of California at Irvine [15], restricting, for the sake of simplicity, our choice to problems with two classes.

The implemented version of the general methodology of LAD specifies a number of options defined by particular values of some parameters set by the user. Typical parameters include, among others, the minimum number of observations which must be covered by each of the chosen patterns, the maximum allowable degree of a pattern, the way of choosing weights of patterns in the discriminant, the minimum required coverage of each observation point, the minimum average distance from each chosen positive (negative) pattern to the set of negative (positive) observation points, etc. The proper selection of parameter values is problem-dependent, and allows to adjust the general methodology to the specific requirements of various areas of applications.

The accuracy of LAD on the datasets described below was estimated by randomly selecting a subset of the dataset (the "training" set), using it for constructing theories, and then evaluating their performance on the complement of the training set (called the "testing" set). The training sets consisted of either 50% or 80% of the observations in the archives. The accuracies of the estimates were obtained as the averages of 20 runs with randomly selected training sets. Table 3 reports the average values and the corresponding standard deviations of the accuracy estimates.

We shall present below a brief description of the datasets from the Irvine repository which were used in our experiments.

Australian Credit Card. The dataset, submitted to the repository by J. Quinlan, consists of 690 records of MasterCard applicants, 307 of which are classified as positive and 383 as negative. While 37 records have some missing data, they were not omitted from our analysis. Each record consists of 15 attributes (4 Boolean, 5 nominal, and 6 numerical).

C. Carter and J. Catlett [4] reported experiments on this dataset using a decision tree technique, and obtained a correct prediction rate of 85.5%, using a training set of 71%. In a testing set of 200 observations, 171 points were classified correctly, 18 points were misclassified, and 11 points were not classified at all.

It is interesting to notice that the LAD techniques applied to this problem detect a degree 1 pattern consisting of the 9-th attribute alone, which correctly predicts the outcome in 85.51% of the cases. Furthermore, it is interesting to notice that the inclusion of additional patterns in the discriminant does not seem to improve the prediction accuracy.

Boston Housing. This dataset was created by D. Harrison and D. Rubinfeld in 1978 and contains 506 records describing housing values in the suburbs of Boston, depending on observations consisting of one binary and 12 continuous attributes. This dataset is split into two classes depending on whether the housing value is above or below the median. Using training sets of 80% of the observations, [16] reports correct prediction rates ranging from 82% to 83.2%.

Breast Cancer (Wisconsin). The dataset, compiled by O. Mangasarian and K.P. Bennett, is widely used in the machine learning community for comparing learning algorithms. It is, however, difficult to use it for rigorous comparisons since its content is constantly updated.

Each observation describes a cytological test specified by 9 numerical attributes with integer values between 1 and 10. A binary output variable indicates the benign or malignant nature of the tumor. Currently, the dataset consists of 699 observations, of which 683 are complete (i.e. have no missing data) and were used in our experiments. Discounting repetitions, there are 213 distinct observations about malignant tumors and 236 distinct observations about benign ones.

A comparison of 5 different algorithms applied to this dataset is reported in [16] with correct prediction rates ranging from 94.5% to 96.2%, based on training sets consisting of 80% of the data. Even better results are reported by G. Herman and K. Yeung in [11]. Since their analysis is restricted to a subset of this dataset, and since it uses a much larger proportion of the data for training, no meaningful comparisons seem to be possible.

Congressional Voting. The dataset, contributed by J. Schlimmer, includes votes for each of the U.S. House of Representative Congressmen in the 98th Congress, 2nd session 1984, on 16 key issues identified by the Congressional Quarterly Almanac (CQA, Volume XL).

The dataset contains 435 data (267 Democrats and 168 Republicans). Each of the 16 votes is represented by a nominal attribute taking the values "for", "against", or "did not vote". The "outcome" associated with each observation is simply the party affiliation of each Congressman (Democrat or Republican). The problem is to identify the party membership on the basis of the results of the 16 votes. In our experiments "did not vote" was treated as a missing value. This interpretation led to the removal from the dataset of observations, corresponding to 6 Congressmen who did not vote on most of the issues, in order to make the dataset consistent.

According to [12], correct predictions rates reported in the literature about this dataset range from 84% to 95.6%.

Diabetes. This dataset, compiled by the National Institute of Diabetes and Digestive and Kidney Diseases, was contributed to the repository by V. Sigillito. The dataset consists of 768 complete observations, about $\frac{2}{3}$ of which correspond to patients showing signs of diabetes, and the rest exhibiting no such signs. Each patient is described by 8 numerical attributes. Using training sets of 80%, [16] reports correct prediction rates ranging from 71.4% to 74.4%. Further, [19] reports a 76% correct prediction rate using 75% of the data for training.

Heart Disease (Cleveland). The Cleveland Clinic Foundation heart disease dataset, contributed to the repository by R. Detrano, contains 303 observations, 165 of which describe healthy people and 138 sick ones; 7 observations are incomplete, and 2 of the observations of healthy people have identical attribute values. Each observation is described by 13 attributes, including 3 Boolean (e.g. sex), 4 nominal (e.g. type of chest pain), and 6 numerical (e.g. age). The output indicates the angiographic status of the disease, i.e. whether the narrowing of the vessel diameter is above or below 50%. According to [12] and [18] many algorithms have been tried on this problem with prediction accuracies ranging from 60.5% to 80.6%. No information is available about the size of the training set used in the best of these results.

The average correct prediction rates and their standard deviations (over 20 runs) obtained by using the LAD classifier are reported in Table 3 along with the best prediction rates found in the literature. The table reports two estimates of the accuracy of LAD, one using 50% and the other one using 80% of the dataset for training. For simplicity, in calculating the correct prediction rates of LAD, those observations that were not classified by the LAD classifier were counted as errors.

Since the results reported in the literature do not always reveal all the important parameters of the experiments (e.g., number of runs, standard deviation, sample size), no rigorous statistical statement about the comparative performance of LAD is possible. As can be seen from Table 3, LAD's performance compares quite favorably with the best performances reported in the literature.

In order to put LAD's performance in the right perspective, it should be also added that the results used in the comparisons were not obtained by the use of a single "universally good" method. In fact, for each of the datasets we selected the best results reported in the literature, regardless of the particular method used.

The conclusion emerging from these experiments is that LAD is competitive with the well established classification methods.

8 Some Pilot Studies

In order to illustrate the applicability of LAD techniques we report below on the results of three pilot studies with real data taken from various fields of applications. The evaluation of the performance of

	LAD classifier				Other approaches			
	50% training		80% training		best found		% used for	
DataBase	AVG	STD	AVG	STD	AVG	STD	training	reference
Australian Credit Card	85.4	1.2	85.5	2.6	85.5	*	71%	[4]
Boston Housing	84.0	1.6	85.2	3.0	83.2	3.1	80%	[16]
Breast Cancer (Wisconsin)	96.9	0.9	97.2	1.3	96.2	0.3	80%	[16]
Congressional Voting	96.2	1.1	96.6	1.8	95.6	*	66.6%	[12]
Diabetes	71.9	1.9	72.3	2.4	76	*	75%	[19]
Heart Disease (Cleveland)	82.3	1.7	83.8	5.2	80.6	3.1	*	[18]

(*) not available

Table 3: Correct Prediction Rates of LAD and Other Methods

LAD was done as described in the beginning of the previous section (although the training set size was usually much smaller). The studies demonstrate not only the classification power of LAD, but also show its flexibility and capability to provide answers to various case-dependent problems.

8.1 Oil Exploration

In view of the high price of drilling for oil, it is important to establish with a high degree of accuracy the composition of the soil (collector or not), and in case the soil is a collector, to know whether it collects oil. Technically, the answer to these questions is obtained by using a number of "well logging tools" which provide several electrical, nuclear, and acoustical measurements taken at various depth levels. These measurements give some information about soil porosity, hydrocarbon saturation, permeable bed thickness, and permeability. They allow the experts to draw conclusions about the nature of the collector and its content.

The Chevron Corporation kindly provided us with two datasets coming from potential oil fields. The datasets consist of 1632 and 2393 observations respectively. Each observation contains 7 measurements, and was classified by an expert as either "effective" or "defective".

Surprisingly, the logical analysis of the first dataset has provided a theory consisting of one single pattern (having degree 3). Moreover, this single pattern theory has a prediction accuracy of over 94%. The pattern was discovered using only 3% of the dataset for training, and the accuracy was evaluated on the remaining 97%.

Similarly, the logical analysis of the second dataset has provided a theory consisting also of one single pattern, involving this time only two attributes, and having a prediction accuracy of 92.3%. This result was also obtained using 3% of the dataset for training.

According to the domain experts, these results compare very favorably with the ones obtained by using other techniques.

As an interesting additional benefit of the logical analysis of these two datasets, it can be seen that 3 of the 7 measurements taken in each observation were not used in either of the two patterns, and are therefore redundant.

It is particularly interesting to notice that LAD has supplied, beside the theories mentioned above, an extremely simple "rule of thumb" for evaluating observations in the first dataset. This rule of thumb, having an accuracy of "only" 90% consists of a single binary variable indicating whether the fifth original attribute takes a value above or below a particular cut point. This result is in a sharp contrast with the fact that the correlation of none of the original variables with the outcome exceeds (in absolute value) 0.63.

8.2 Psychometric Testing

A typical testing procedure in psychometrics consists in the analysis of answers provided by patients to standard questionnaires constructed to diagnose certain psychiatric conditions. The questions in such questionnaires are called items. Some of these questionnaires can be very long. It is therefore quite natural to look for smaller subsets of items in a questionnaire capable of providing the same conclusions about the state of a patient as the entire set. Clearly, in the LAD terminology, such subsets will represent (minimal) support sets.

A recently completed pilot study ([14]) analyzed a dataset concerning the Beck Depression Inventory (BDI). The dataset was compiled at the New Hope Guild Centers, Brooklyn, NY. It consists of 280 patients selected randomly from a large population. The BDI is a multiple-choice questionnaire with 21 questions with 4-fold responses (denoted by 0,1,2, and 3). The *scale score* is the sum of item-responses, and ranges from 0 to 63. Patients with higher scale scores are more severely depressed. A four-fold classification based on scale scores is shown in Table 4.

Scale Score	Level of Depression
1 – 9	normal
10 – 18	mild-moderate
19 – 29	moderate-severe
30 – 63	extremely severe

Table 4: BDI Scale Score Classification

The pilot study aimed at developing a simple way to distinguish depressive patients (having a score exceeding 18) from non-depressive ones (having a score of 18 or less).

By applying LAD to a training set consisting of 50% of the dataset, it was found that it is sufficient to use one binary variable for each item. Moreover, a support set consisting of only 16 Boolean variables was found, allowing for a compression of the original test from 21 4-valued items to 16 2-valued ones.

A theory discriminating depressive and non-depressive patients was developed. It consisted of 12 positive and 11 negative patterns, each of them involving at most three binary variables.

The evaluation of this theory on the testing set gave 90.2% correct predictions, 8.9% errors, and left 0.9% of the patients not classified. The accuracy of this classification was judged as excellent by the domain experts. Moreover, the analysis provided the experts with a number of additional valuable results, including:

- a large number of positive and negative patterns, many of which were found by the experts to be significant as descriptors of certain homogeneous groups of patients (similar to syndromes in medicine);
- construction of several psychometric tests, all providing the same classification of the patients, but using different subsets of the original questions, having minimal pairwise intersections;
- construction of adaptive testing procedures giving substantial reductions in the number of questions each patient has to answer.

It is interesting to note that although the theory was produced for the binary classification of patients as depressive or non-depressive, it was possible to closely approximate the original test scores using linear regression techniques with individual patterns serving as variables ($R^2 = 0.82$).

This study can be extended to much longer psychometric questionnaires. For example, the frequently used Minnesota Multiphase Personality Inventory-2 (MMPI-2) test contains 567 items. Another extension can proceed in the direction of providing not only one, but two or more support sets, having minimal pairwise intersections. This would allow the application to the same patient of several equivalent tests involving different items, as a way of verifying the validity of that patient's answers.

8.3 Labor Productivity in China[4]

Using the official China Statistical Yearbooks, a dataset was compiled in [8, 7] for the analysis of labor productivity in the developing Chinese economy. The dataset provides 290 annual observations for the 29 Chinese provinces for the period 1985 – 1994. Each observation includes values of 9 attributes. Three of the attributes represent classical economic parameters: labor productivity (LABPRO), capital / labor ratio (CAPLAB), and firm size (FIRMSIZE). Four additional attributes represent the relative positions of various types of ownership, as measured by the proportions of production originating from state-owned enterprises (SOE), collectively-owned enterprises (COE), individually-owned enterprises (IND), and "other" enterprises (OTHER) consisting mostly of foreign-owned enterprises[5]. The TIME attribute, running from 1985 to 1994, simply represents the year of the observation. Finally, in [8, 7] China was partitioned into four geographic regions: Southeast (1), Northeast (2), Southwest (3), and Northwest (4), represented by the values of the attribute REGION. Datasets of this type are frequently studied using the techniques of Data Envelopment Analysis (DEA), see e.g. [5]. LAD techniques provide an alternative approach to this problem.

The value of LABPRO was chosen to be the outcome variable, with the other 8 attributes serving as input variables. The methodology of LAD allows the binarization of input variables when the outcome variable is binary. The binarization of the outcome attribute LABPRO was accomplished by using two threshold values λ^+ and λ^- ($\leq \lambda^+$). The observations with a LABPRO above λ^+ were considered positive, while the observations with a LABPRO below λ^- were considered negative. The observations between λ^- and λ^+ were eliminated from the analysis. A particularly important part of the study concentrated on the "gap-free" case $\lambda^- = \lambda^+$, when no observation had to be eliminated from the dataset.

In a first group of experiments, the 49 observations with "very high" labor productivity ($\lambda^+ = \hat{\lambda} + \sigma_\lambda$, where $\hat{\lambda}$ is the mean, and σ_λ is the standard deviation of LABPRO) were compared with the 36 observations with "very low" labor productivity ($\lambda^- = \hat{\lambda} - \sigma_\lambda$).

Several very important patterns have been uncovered by the logical analysis of this dataset. Some typical examples are the following:

P1: REGION is not 4, and TIME $\geq$ 1988
N1: SOE $\geq$ 71.44%, and TIME $\leq$ 1987

The positive pattern P1 by itself represents a theory, having the remarkable accuracy rate of 100%! The negative pattern N1 can also be considered a theory, and it also has a high accuracy (94%). It is interesting to note that these patterns highlight not only the obvious growth of productivity over time, but also the more subtle impact of both ownership type and geographic location on labor productivity.

In a second group of experiments, the 123 observations of "high" productivity ($\lambda^+ = \hat{\lambda}$) were compared with the 167 observations of "low" productivity ($\lambda^- = \hat{\lambda}$). The logical analysis of this dataset produced a theory with 12 positive and 15 negative patterns, having a prediction accuracy of 92.7%.

The results of these experiments show clearly that the input attributes play an important role in determining labor productivity. Moreover, the results also show that some of these attributes have a consistently positive impact on productivity (COE, TIME, Southeastern location), while some other attributes have a consistently negative impact on productivity (SOE,IND[6], Northwestern location). These conclusions reached by LAD are in full agreement with those obtained by using econometric techniques in [8, 7], reconfirm them, and provide a further insight into the mechanism of productivity changes.

A third group of experiments utilized the results described above with the aim of creating a decision support system for increasing productivity. The precise goal of this study was to find out

[4]Details of this study can be found in [9].

[5]This ownership classification comes from the official Chinese statistics.

[6]The classification by the State Statistical Bureau of China includes in IND only enterprises employing at most 7 workers.

which attributes have to be modified in order for a province to surpass a certain productivity level. In our study this critical level was chosen to be $\hat{\lambda}$, since for this value a theory distinguishing observations with LABPRO $< \hat{\lambda}$ from those with LABPRO $> \hat{\lambda}$ has already been developed in the experiments described above. We selected pairs of observations (p, p') for which LABPRO increased in a year from below $\hat{\lambda}$ to above $\hat{\lambda}$. We compared then the positive and negative patterns covering the observation p before the transition with the patterns covering the observation p' after the transition.

The analysis showed that no negative patterns cover p', unless they already cover p. Similarly, every positive pattern covering p also covers p'. Moreover, in most cases, either the positive coverage of p' increased compared with that of p, or the negative coverage of p' decreased, or both. In order to confirm these observations, similar experiments were carried out using 15 different levels of productivity. As a non-standard application of these transition rules, a logic-based decision support system was constructed to inform decision makers about various changes in the values of the attributes which can lead to productivity increases, thus allowing the choice of the optimal alternative.

As a concrete application, it was shown that in order to increase the productivity of the province of Guizhou in 1995 above $\hat{\lambda}$, the level of COE should be increased from 16.41% in 1994 to at least 17.9% in 1995. It is not known at the time of this writing whether this increase did actually occur, since the 1995 statistical data are not yet published.

9 Conclusions

A common feature of many practical problems appearing in science, technology, administration, etc., is the need to analyze and interpret archives of observations. Various analytic machineries, some classical (e.g. statistics) and some more modern (e.g. machine learning, neural networks) were built to respond to such needs.

The logical analysis of data is concerned with such problems, and utilizes logic-based procedures for this goal. Originally, LAD was developed for the analysis of binary data, both in the input and the output. Later this methodology was extended for handling numerical data.

The basic concepts of LAD are:

- *cut points* of numerical attributes and associated *binary variables*;
- *support sets* consisting of minimal subsets of attributes capable of explaining the entire archive;
- *patterns* consisting of combinations of a (usually small) number of attribute ranges characteristic for a certain (positive or negative) outcome;
- *theories* consisting of collections of patterns providing a complete description of the archive;
- *discriminants* providing classifications by using weighted combinations of the patterns in the theory.

The key feature of LAD consists in the detection of hidden patterns governing the analyzed phenomena. This approach can produce a large number of such patterns, but normally retains only small fractions of them capable of providing complete theories explaining the phenomena. In the case of numerical data, the patterns are described in terms of binary variables associated to the original ones dichotomized around some critical values.

While the uses of LAD include pattern detection, support set selection, theory formation, etc., classification is perhaps the only major application of LAD which can be directly compared with other methods. The classification performance of LAD was tested on some standard datasets used by the machine learning community. Despite the fact that LAD's performance was compared to the best reported results produced by several methods (see end of Section 8), LAD was found to perform favorably in five of the six datasets considered. Therefore LAD, as a classification technique, appears to be competitive with the well established methods of this area.

A number of pilot studies are briefly described in the last section of the paper. In order to illustrate the variety of potential applications of LAD, we have included here studies on oil exploration, psychometric testing, and the analysis of labor productivity in China. The pilot studies indicate both the power and the flexibility of LAD. Indeed, LAD was successful in providing meaningful insights into the nature of the problems analyzed, and was even able to provide solutions to problems as diverse as the construction of various equivalent psychometric tests, or the development of a decision support system for increasing labor productivity in China.

In conclusion, LAD is a new methodology with several appealing features including:

- wide applicability, not limited by assumptions about special properties of data archives;
- high classification accuracy comparable to and frequently exceeding other methods;
- robustness with respect to noise and measurement errors, and capability of handling some missing data;
- great flexibility in achieving varied, problem dependent goals;
- strong explanatory power based on the use of patterns easily interpretable by domain experts.

It should be noted that research on LAD is at its beginning, and that much further research is needed for a better understanding of the mathematical and computational aspects of LAD, as well as its domains of applicability.

Acknowledgment

The authors thank an anonymous referee for a careful reading of the manuscript and for many useful comments.

References

[1] Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Error-free and best-fit extensions of partially defined Boolean functions. RUTCOR Research Report, RRR 14-95, RUTCOR – Rutgers University's Center for Operations Research, April 1995.

[2] Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Boolean analysis of incomplete examples. RUTCOR Research Report, RRR 7-96, RUTCOR – Rutgers University's Center for Operations Research, February 1996.

[3] Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Extensions of partially defined Boolean functions with missing data. RUTCOR Research Report, RRR 6-96, RUTCOR – Rutgers University's Center for Operations Research, January 1996.

[4] Chris Carter and Jason Catlett. Assessing credit card applications using machine learning. *IEEE Expert*, pages 71–79, Fall 1987.

[5] A. Charnes, W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operations Research*, 6(2):429–444, 1978.

[6] Yves Crama, Peter L. Hammer, and Toshihide Ibaraki. Cause-effect relationships and partilly defined Boolean functions. *Annals of Operations Research*, 16:299–326, 1988.

[7] Alexander B. Hammer. *The Decline of the "Dinosaurs" in a Restructured and Emerging Economy – Comparing the Relative Efficiency of State and Non-State Enterprises in China's Industrial Sector*. Senior Honors Thesis, Department of Economics, Brandeis University, Waltham, MA, 1995.

[8] Alexander B. Hammer. The dynamics of development in modern Chinese industry. RUTCOR Technical Report, RTR 1-96, RUTCOR – Rutgers University's Center for Operations Research, April 1996. Presented at the International Conference on Management Science and the Economic Development of China, Hong Kong, July 1996.

[9] Alexander B. Hammer, Peter L. Hammer, and Ilya Muchnik. Logical analysis of Chinese productivity patterns. RUTCOR Technical Report, RTR 1-96, RUTCOR – Rutgers University's Center for Operations Research, April 1996. Presented at the International Conference on Management Science and the Economic Development of China, Hong Kong, July 1996.

[10] Peter L. Hammer. *Partially Defined Boolean Functions and Cause-Effect Relationships.* Presented at the International Conference on Multi-Attrubute Decision Making Via OR-Based Expert Systems, University of Passau, Passau, Germany, April 1986.

[11] G.T. Herman and K.D. Yeung. On piecewise-linear classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):782–786, 1992.

[12] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[13] Eddy Mayoraz. C++ tools for logical analysis of data. RUTCOR Technical Report, RTR 1-95, RUTCOR – Rutgers University's Center for Operations Research, July 1995.

[14] Ilya Muchnik and Leonid Yampolsky. A pilot study of the concurrent validity of logical analysis of data, as applied to two Beck inventories. RUTCOR Technical Report, RTR 2-95, RUTCOR – Rutgers University's Center for Operations Research, July 1995.

[15] P.M. Murphy and D.W. Aha. *UCI Repository of Machine Learning Databases: Machine Readable Data Repository.* Department of Computer Science, University of California, Irvine, CA, 1994.

[16] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[17] Shizuo Senju and Yoshiaki Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15(4):B–196 – B–207, December 1968.

[18] J.W. Shavlik, R.J. Mooney, and G.G. Towell. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143, 1991.

[19] J.W. Smith, J.E. Evelhart, W.C. Dickinson, W.C. Knowler, and R.S. Johannes. Using the Adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*, pages 261–265. IEEE Computer Society Press, 1988.