DIMACS Technical Report 2002-50
December 2002

Spanned Patterns for the Logical Analysis of Data

by

Gabriela Alexe[1] and Peter L. Hammer[1,2]

RUTCOR-Rutgers Center for Operations Research
Rutgers, The State University of New Jersey
640 Bartholomew Rd.
Piscataway, NJ 08854-8003, U.S.A.
e-mail: {alexe@rutcor.rutgers.edu, hammer@rutcor.rutgers.edu}

[1] The partial support provided by ONR grant N00014-92-J-1375 and DIMACS is gratefully acknowledged.
[2] Permanent member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS was founded as an NSF Science and Technology Center, and also receives support from the New jersey Commission on Science and Technology.

ABSTRACT

In a finite dataset consisting of positive and negative observations represented as real valued n-vectors, a *positive* (*negative*) *pattern* is an interval in $\mathbf{R}^n$ with the property that it contains sufficiently many positive (negative) observations, and sufficiently few negative (positive) ones. A pattern is *spanned* if it does not include properly any other interval containing the same set of observations. Although large collections of spanned patterns can provide highly accurate classification models within the framework of the *Logical Analysis of Data*, no efficient method for their generation is currently known. We propose in this paper an incrementally polynomial time algorithm for the generation of all spanned patterns in a dataset, which runs in linear time in the output; the algorithm resembles closely the Blake and Quine consensus method for finding the prime implicants of Boolean functions. The efficiency of the proposed algorithm is tested on various publicly available datasets. In the last part of the paper, we present the results of a series of computational experiments which show the high degree of robustness of spanned patterns.

1 Introduction

Logical Analysis of Data (*LAD*) is a method based on combinatorics, optimization, and Boolean logic, for data analysis. *LAD* was first introduced in [10], [8] as a method for the analysis of binary data, and extended later in [6] to the analysis of datasets having numerical independent variables and binary outcomes (positive and negative observations). *LAD* produces highly accurate, completely reproducible, and robust classification models with high explanatory power, along with novel information about observations and attributes. *LAD* has been successfully applied for the analysis of datasets from different areas e.g., medicine, design of biomaterials, economics, finance, oil exploration and seismology. Computational studies ([7], [4], [14], [2]) show that the accuracy of the *LAD* models compares favorably with that of other machine learning and statistical models.

A central problem in *LAD*, as well as in some other areas of artificial intelligence, machine learning, data mining, etc, is the extraction of positive and negative rules (or *patterns*) from data, and their aggregation into a classification *model* capable of distinguishing between positive and negative observations in the dataset. The two basic concepts used in *LAD* are those of patterns and of models.

To clarify these concepts, let us consider two finite, disjoint sets Ω^+ and Ω^- of vectors of $\mathbf{R}^n$, called respectively *positive* and *negative observations. LAD* identifies two families $\mathcal{F}^+$ and $\mathcal{F}^-$ of intervals in $\mathbf{R}^n$, such that

(i) the union of intervals in $\mathcal{F}^+$, respectively $\mathcal{F}^-$, includes Ω^+, respectively Ω^-;

(ii) the proportion of positive observations in each interval I in $\mathcal{F}^+$ exceeds a prescribed threshold, while the proportion of negative observations is below a (possibly different) threshold; similarly, the proportion of negative observations in each interval I in $\mathcal{F}^-$ exceeds a prescribed threshold, while the proportion of positive observations is below a (possibly different) threshold.

The intervals I in $\mathcal{F}^+$ (respectively, $\mathcal{F}^-$), are called *positive* (respectively, *negative*) *patterns*. The positive patterns I with $I \cap \Omega^- = \varnothing$ are called *pure positive patterns. Pure negative patterns* are defined in a similar way.

There are two important classes – those of prime and of spanned patterns – which are widely used in *LAD* models. An interval I in $\mathcal{F}^+$ is called a *positive prime* (respectively, *spanned*) *pattern*, if it is the inclusionwise maximal (respectively, minimal) positive pattern containing the observations covered by *I*. *Negative prime* and *negative spanned patterns* are defined in a symmetric way.

The major criterion of usefulness of a category of patterns is reflected in its "robustness", i.e., in the degree of similarity or dissimilarity between its performance on the "training set" (i.e., the dataset on which it was learned) and on another dataset called "test set". It will be seen that in the case of spanned patterns, the basic performance measures of patterns (especially their *prevalence* and *homogeneity*, to be defined in Section 2), remain relatively stable on new datasets. This robustness of spanned patterns explains to a large extent their importance in *LAD*.

Given a dataset Ω, a collection of positive and negative patterns with the property that every positive (negative) observation in the dataset belongs to at least one of the positive (negative) patterns in the collection, defines an *LAD model* of the corresponding classification problem. The role of prime and spanned patterns in constructing accurate *LAD* models was analyzed in [12] and [2] through extensive computational studies. In particular, in [12] it was shown that in general, models based on spanned patterns make fewer classification errors, but leave more observations uncovered than those based on prime patterns. Moreover, it was proved in [2] that models consisting of larger collections of patterns are usually more accurate than those which consist of small subsets of patterns. Therefore, *LAD* models using large collections of spanned patterns play an important role whenever classification errors can have substantial undesirable effects (e.g. in the case of medical diagnosis).

Pattern generation is a central problem of *LAD*. In earlier implementations of *LAD*, pattern generation (in the binary case) was carried out by using two enumeration techniques, called *bottom-up* and *top-down* [7]. The *top-down* approach [7] starts by associating to every positive (negative) observation its "characteristic term" (which can be viewed as an interval reduced to one point), and systematically removes literals (i.e. eliminates the corresponding restrictions on the interval), until arriving to a prime (pure) positive (negative) pattern. The *bottom-up* approach [7] starts by intervals defined by one non-redundant constraint, and systematically adds non-redundant constraints to each of them, until generating a (pure) pattern. In practice, these two approaches are combined in a hybrid method, which applies the *bottom-up* procedure until generating all the patterns defined by at most d (usually 4 or 5) non-redundant constraints, and applies then the *top-down* procedure to cover those observations which remained uncovered after the bottom-up step. All these procedures have an exponential complexity in both input and output, and can run only on datasets of restricted size.

While an efficient algorithm for enumerating all prime patterns of a dataset [3], as well as a branch-and-bound algorithm for constructing the positive and the negative pattern of maximum coverage [9] have been recently developed, no specific method is yet available for the systematic enumeration of large collections of spanned patterns. In view of the exponentially large number of spanned patterns, this task can only be accomplished with the help of efficient algorithms running in total polynomial time. The description of such an algorithm, running in fact in incremental polynomial time, and resembling the consensus method of Blake [5] and Quine [17] for finding the prime implicants of a Boolean function, is the aim of this paper.

This paper is organized as follows. After introducing in Section 2 several definitions and notations, we present in Sections 3 and 4 a consensus-type algorithm and an accelerated version of it for the generation of all spanned patterns associated to a dataset. Section 5 describes an implementation of the accelerated algorithm, and analyzes its efficiency. Section 6 presents computational evidence showing the robustness of the class of spanned patterns.

2 Definitions and Notations

Let Ω be a dataset consisting of m observations $\omega^1,\dots, \omega^m$. Each observation is represented as a vector $\omega^i = \{ a^i{}_1,\dots, a^i{}_n \}$ in $\mathbf{R}^n$ (indicating the values $a^i{}_j$ of the attributes $A_1,\dots, A_n$), having an outcome which can be "positive" or "negative". The set of positive observations is denoted Ω^+, and the set of negative observations Ω^-. In this paper we shall assume that Ω^+ and Ω^- are disjoint.

For the sake of simplicity, we shall usually assume that all the a^i_j's belong to the set $\{0,1,..., k_j\}$. As a matter of fact, it is easy to notice that this assumption is not restrictive, since every dataset can be brought to this form with the help of a simple transformation. Such a transformation, called *discretization*, uses a set of cutpoints for partitioning the domain of each attribute into a finite number of intervals (see e.g., [4]).

Let l and u be two vectors in $\mathbf{R}^n$ with $l_j \le u_j$ for $j = 1,\ldots, n$. The set of all n -vectors $(x_1,..., x_n)$ satisfying $l_j \le x_j \le u_j$ will be called an *interval* and denoted $I = [l_1, u_1] \times \ldots \times [l_n, u_n]$. The *coverage* $cov(I)$ of the interval I is the set of observations contained in I, i.e. $cov(I) = I \cap \Omega$; we shall frequently distinguish between the *positive* and the *negative coverages* of I, defined as $cov^+(I) = I \cap \Omega^+$ and as $cov^-(I) = I \cap \Omega^-$, respectively. The ratios $|cov(I)| / |\Omega|$, $|cov^+(I)| / |\Omega^+|$ and $|cov_-(I)| / |\Omega^-|$ will be called the *prevalence*, the *positive prevalence*, and the *negative prevalence* of I, respectively, and will be denoted by π_I, π^+_I, and π^-_I. The *degree* of the interval I, denoted $deg(I)$, is the number of attributes A_i for which at least one of the inequalities $l_i \le A_i$ or $A_i \le u_i$ is non-redundant.

An interval P is called a *pure positive pattern* if $\pi^+_P > 0$ and $\pi^-_P = 0$, and it is called a *pure negative pattern* if $\pi^-_P > 0$ and $\pi^+_P = 0$. The ratio π^+_P / π_P is called the *homogeneity* χ_P of the pattern P. In most studies we are only interested in those positive patterns, whose homogeneity exceeds a certain fixed threshold λ, usually equal to at least 0.8 or 0.9. Similarly, in the case of negative patterns we usually require the homogeneity not to exceed 0.1 or 0.2.

A positive (negative) pattern P is called *maximal* (or *strong* [12]) if its positive (respectively, negative) coverage is maximal with respect to set inclusion. Given a subset of observations T, the *interval spanned by* T, denoted $Span(T)$, is the inclusionwise minimal interval containing T. If a pattern is spanned by a subset T, we shall simply call it a *spanned pattern*. Clearly, every pattern spanned by a set of observations T can be represented as $[u_1, v_1] \times \ldots \times [u_n, v_n]$, where for every $j = 1,..., n$, $u_j = \min_i a^i_j$, $v_j = \max_i a^i_j$, with i running over all the observations $(a^i_1,\ldots, a^i_n)$ in T.

Given a dataset Ω, the *Spanned Positive Pattern Generation* (*SPPG*) problem consists in generating all the positive patterns spanned by all the subsets of observations of Ω^+. *The Spanned Negative Pattern Generation* (*SNPG*) problem is defined similarly. Because of the perfect analogy between these problems, we shall discuss below only the *SPPG* problem.

SPPG is a hard problem, since the number of pure spanned patterns may be exponential in the size of Ω^+. For example, if we assume that all the observations in Ω are positive, and that Ω^+ consists of the set of m records $\{(1,0,....,0),...., (0,....,0,1)\}$, where the i-th record is simply the i-th unit vector, then it is easy to see that there are 2^m - 1 distinct (pure) positive patterns spanned by the elements in Ω^+. Another reason for which *SPPG* is a hard problem is that determining the maximum size spanned pattern was shown ([9]) to be *NPC*.

3 *SPAN* - A Consensus-Type Algorithm for Generating All Positive Spanned Patterns

In this section we shall describe a consensus-type method for solving the *SPPG* problem, along with an implementation of it, which runs in incremental polynomial time. Since the introduction of the consensus method for finding the prime implicants of a Boolean function ([5], [17]), several other consensus methods appeared in the literature. Malgrange [16] uses a consensus-type approach to find all the maximal submatrices consisting of ones of a 0-1 matrix. Another consensus-type algorithm is developed in [1] for finding all maximal bicliques of a graph. A generalization of the consensus method for pseudo-Boolean functions is presented in [10].

Consensus-type methods enumerate all the maximal objects of a certain collection, by starting from a sufficiently large set of objects, and systematically completing it by the application of two simple operations. (*i*) The operation of *consensus adjunction* associates to a pair of objects in the given collection one or more new objects, and adds them to the collection. (*ii*) The operation of *absorption* removes from the collection those objects which are "dominated" by other objects in the collection. The two operations are repeated as many times as possible, leading eventually to a collection consisting exactly of all the maximal objects.

The proposed consensus-type method for spanned pattern generation starts from a dataset $\Omega = \Omega^+ \cup \Omega^-$, and a family of spanned (say, positive) patterns, the union of which includes Ω^+. Let $P = [a_1, b_1] \times ... \times [a_n, b_n]$ and $P' = [a'_1, b'_1] \times ... \times [a'_n, b'_n]$ be a pair of positive spanned patterns, and let P'' be the (spanned!) pattern $[a''_1, b''_1] \times ... \times [a''_n, b''_n]$, where $a''_i = \min \{ a_i , a'_i \}$, and $b''_i = \max \{ b_i , b'_i \}$, $i = 1,..., n$. If P'' is a positive pattern, then it is called the *consensus* of the patterns P and P'. In this way, a pair of positive spanned patterns may have at most one consensus, which is the pattern spanned by the observations in $cov(P) \cup cov(P')$. We have to remark that the consensus adjunction operation of two patterns P and P' uses besides the information given by the positive patterns P and P', the information given by the set Ω^- of negative observations.

We say that the positive spanned pattern P *absorbs* the positive spanned pattern P' if simply $P = P'$.

Let us consider the dataset Ω in $\mathbf{R}^n$ consisting of m^+ positive and m^- negative observations, and let us consider an algorithm A for the *SPPG* problem, which outputs sequentially all the positive spanned patterns $P_1,..., P_\beta$ of Ω^+. Let us denote by $\tau(k)$ the running time of A until the output of P_k, for $k = 1, 2, \dots, \beta$, and by τ^* the total running time of A.

We recall that (according to [13]), an algorithm is said to run in *polynomial total time* if its total running time τ^* is polynomially bounded in the size of the input and output. Similarly, an algorithm runs in *incremental polynomial time* if it runs in polynomial total time, and the running time between any two consecutive outputs is polynomially bounded in the size of the input and output.

Algorithm *SPAN* for generating all positive spanned patterns

1. **Initiate** the collection C with the m^+ positive patterns spanned by each individual observation in Ω^+.

2. **Repeat** the following two operations **until** the collection C cannot be enlarged anymore:

(i) *Consensus adjunction*: If there is a pair of patterns P, P' in C, having a consensus P'', add P'' to C.

(ii) *Absorption*: If there is a pair of patterns P, P' in C, such that P' absorbs (is the duplicate of) P, then eliminate P' from C.

Clearly, the two transformations above can be replaced by the following equivalent one:

(*) If there is a pair of patterns P and P' in C, having a consensus P'', not absorbed by any element in C, add P'' to C.

Theorem 1. *Algorithm SPAN terminates and at termination the final list C contains all the pure positive patterns spanned by subsets of observations in* Ω^+.

Proof. The algorithm stops after a finite number of steps, since the number of spanned patterns is finite, and once a spanned pattern is in C, it can never reenter this list.

Let us prove now that the final list C consists of all the positive spanned patterns. Assume that P is a positive spanned pattern which is not contained in the list C when the algorithm stops. Let $cov(P) = \{v_1,..., v_k\}$, and let $P_1,..., P_k$ be the patterns spanned by each of the observations $v_1,..., v_k$, respectively.

We remark that the spanned patterns $P_1,..., P_k$ are contained in the initial list C, and they were never deleted from C during the application of the algorithm, since the coverage $cov(P_i) = 1$ for each P_i, $i = 1,..., k$, and the coverage of any pattern produced by consensus adjunction is at least 2.

We also remark that if S' and S'' are two arbitrary subsets of $cov(P)$, then the consensus of the patterns Span(S') and Span(S'') exists and it is a spanned pattern included in P, and contained in the final list C.

Since algorithm *SPAN* performs the consensus adjunction transformation for every pair of patterns in the current list C, eventually it must perform the consensus adjunction for the pair P_1 and P_2. Let $P_{1,2}$ be the resulting consensus. According to the second remark, $P_{1,2}$ will be added to C, if not already there. Continuing in this way, the algorithm performs eventually the consensus adjunction for the pair $P_{1,2}$ and P_3, producing the pattern $P_{1,2,3}$, which will be added to C if not already there, and so on. Finally, the algorithm must perform the consensus adjunction for the pair of patterns $P_{1,...,k-1}$ and P_k, and the resulting pattern $P_{1,...,k}$ will be added to C if not already there. Since both $P_{1,...,k}$ and P are spanned patterns, $P_{1,...,k} \subseteq P$, and since obviously $cov(P_{1,...,k}) \supseteq cov(P)$, it follows that $P_{1,...,k} = P$. This contradicts the assumption that P is not in the final list C. ■

The size of the current list C during the execution of algorithm *SPAN* never exceeds $\beta+1$. The complexity of this algorithm can be seen to be $O(\varphi(m, n)\beta)$, where $\varphi(m, n)$ is a polynomial in m and n.

4 *SPIC* – An Accelerated Generation Algorithm of Spanned Patterns via Input Consensus

We shall describe below a variant of algorithm *SPAN* which runs in incremental polynomial time. In algorithm *SPAN*, an input list C of patterns was updated step by step, and consensuses were systematically generated between pairs of patterns in C. In the proposed accelerated algorithm *Spanned Patterns via Input Consensus Algorithm* (*SPIC*), we shall still update the list C, but shall restrict pattern formation to pairs of patterns consisting of one pattern belonging to the initial list C_0 (which remains unchanged during the execution of the algorithm), and one pattern belonging to the updated list C. Clearly, the number of pairs to be examined for pattern formation is substantially reduced in this way. It will be seen in Theorem 2 that *SPIC* produces all spanned patterns, with a substantial reduction in the worst case running time. The computational experience in applying *SPIC* to several real life datasets, presented in section 5 shows the high efficiency of this algorithm.

Algorithm *SPIC*
Let C_0 be the collection of patterns spanned individually by each one of the observations in Ω^+.

1. **Initiate** $C := C_0$.
2. **Repeat** the following operation **until** the collection C cannot be furthermore enlarged:
   If there is a pair of patterns P_0 in C_0 and P in C, having a consensus P' not contained in C, add P' to C.

Theorem 2. *Algorithm SPIC generates all spanned patterns, runs in incremental polynomial time, with* $\tau(k) = \mathrm{O}(m^+k(n+m+n\log_2 k))$, $k = 1,\dots, \beta$, *the total running time being* $\mathrm{O}(\beta\, m^+(m+nm^+))$.

Proof. The correctness of algorithm *SPIC* follows directly from the proof of Theorem 1, in view of the fact that the patterns $P_1,\dots, P_k$ are in C_0. Let us prove now that *SPIC* runs in incremental polynomial time. Assume that the pure spanned patterns are labeled $P_1,\dots, P_\beta$, in the order in which they are produced by the algorithm. For $k = 1, 2, \dots, \beta$, the pattern P_k is added to the list at time $\tau(k)$. Note that at most $\mathrm{O}(km^+)$ consensus adjunctions can be completed until time $\tau(k)$ (to avoid absorptions, we assume that consensus adjunction is performed only for pairs P_0 in C_0, P in C, such that $P_0 \not\subset P$. Each such transformation requires $\mathrm{O}(n)$ time for consensus formation (i.e., creation of the candidates for consensus), $\mathrm{O}(m)$ time for checking if the candidate is a positive pattern, and $\mathrm{O}(n\log_2 k)$ time for checking by binary search whether the candidate is in the list C (this requires a data structure which maintains $\{P_1, P_2, \dots, P_k\}$ as an ordered list throughout the algorithm). Therefore, $\tau(k) = \mathrm{O}(m^+k(n + m + n\log_2 k))$ for $k = 1, 2, \dots, \beta$. Since $\beta \leq 2^{m+}$, the total running time of algorithm *SPIC* is $\tau^* = \mathrm{O}(m^+ \beta\ (n + m + nm^+)) = \mathrm{O}(m^+ \beta\, (m + nm^+))$. ■

Example. Let us illustrate algorithm *SPIC* for the dataset $\Omega = \{v_1 = [1,0,2], v_2 = [0,2,0], v_3 = [3,1,1], v_4 = [2,0,2]\}$, all the observations of which, with the exception of v_2, are positive.

- The input collection C_0 is $\{P_1 = [1,1] \times [0,0] \times [2,2], P_3 = [3,3] \times [1,1] \times [1,1], P_4 = [2,2] \times [0,0] \times [2,2]$. Initialize $C := C_0$.
- Perform consensus adjunction for the pair of patterns P_1 in C_0 and P_3 in C: the candidate for consensus is $P_{1,3} = [1,3] \times [0,1] \times [1,2]$, having $cov(P_{1,3}) = \{v_1, v_3, v_4\} = \Omega^+$; since $P_{1,3}$ is not contained in C, it is added to C.
- Perform consensus adjunction for the pair of patterns P_1 in C_0 and P_4 in C: the candidate for consensus is $P_{1,4} = [1,2] \times [0,0] \times [2,2]$, having $cov(P_{1,4}) = \{v_1, v_4\} \subseteq \Omega^+$; since $P_{1,4}$ is not contained in C, it is added to C.
- Perform consensus adjunction for the pair of patterns P_3 in C_0 and P_4 in C: the candidate for consensus is $P_{3,4} = [2,3] \times [0,1] \times [1,2]$, having $cov(P_{3,4}) = \{v_3, v_4\} \subseteq \Omega_+$; since $P_{3,4}$ is not contained in C, it is added to C.
- The consensus of any other pair of patterns from C and C_0 is contained in C. The algorithm stops and outputs the family of all positive spanned patterns $C = \{ P_1, P_3, P_4, P_{1,3}, P_{1,4}, P_{3,4}\}$.

Generation of all strong spanned patterns. The list L of all strong positive spanned patterns can be easily obtained from the output collection C, by selecting from C the maximal elements with respect to set inclusion. The list L can be also be produced and updated gradually, during the consensus-type procedure: L is initialized with the empty set, and whenever a consensus candidate, say P, is added to C, it is checked whether P is already contained in a pattern in L. If the test fails, then P is added to L, and all patterns in L which are contained in P are deleted from L. The selection of all strong pure spanned patterns can be performed in an additional time of order $O(\beta^2)$; however, we are not able to guarantee yet a total polynomial-time for producing all strong spanned patterns. In fact, the dualization problem of a monotone non-decreasing Boolean function can be reduced in quadratic time to the problem of generating all strong spanned patterns of a certain dataset (see [12]). Thus, the existence of a total polynomial-time algorithm for generating all strong spanned patterns would imply the existence of a total polynomial-time algorithm for the dualization problem mentioned above; until now, the best known algorithms are pseudo-polynomial.

Example (*continued*) If we want to find all the *maximal* patterns in the previous Example, we shall modify the procedure as follows. We create in the initial step a "current" list L of candidates for the collection of maximal patterns. We initialize L with C_0. In the next step, when $P_{1,3}$ is added to C, we add it to L too, and we delete P_1 and P_3 from L. In the following step, when $P_{1,4}$ is added to C, it is also added to L, while P_4 is deleted from L. Finally, when $P_{3,4}$ is added to C, it is also added to L, and no pattern of L is removed at this stage. Thus, the final list of maximal patterns is $L = \{ P_{1,3}, P_{1,4}, P_{3,4}\}$.

5 Implementation and Performance of *SPIC*

In its current implementation, algorithm *SPIC* consists in a sequence of (at most n+1) successive stages. Stage k starts with the list W_{k-1} of pure spanned patterns produced at stage k-1; in the initial stage, W_1 consists simply in the list C_0 of individual positive observations, each being interpreted as the pattern spanned by a singleton. The algorithm produces at stage k the consensus of each of the patterns P in W_{k-1} with each of the patterns P_0 in C_0, whenever P_0 is not included in P. If the resulting consensus candidate is not in W_{k-1}, it is added to the new list W_k. Stage k is completed by merging W_{k} into the sorted list C.

In all real-life applications encountered, accurate classification does not necessarily require the availability of the entire collection of spanned patterns, and can be achieved by using a sufficiently large subset of high prevalence spanned patterns. It is therefore important to apply various filtering mechanisms to restrict the number of patterns produced, and to keep in this way both time and memory requirements at an acceptable level. The final list of spanned patterns is obtained from the list C, based on several selection criteria, which include restrictions on the number of patterns produced, the total time allocation, and the characteristic parameters (e.g., prevalence, homogeneity) of the retained patterns.

The implemented version of algorithm *SPIC* includes several accelerating heuristics. One of the most important procedures used for this purpose, randomly partitions the original dataset into several subsets, applies the input-consensus algorithm separately to the subsets, and after eliminating redundancies in the union of these subsets, creates a final list of spanned patterns. Another heuristic included in the current implementation of *SPIC* applies a preselection mechanism along the process, eliminating from consideration those patterns whose parameters (prevalence, homogeneity, etc) are not sufficiently high.

In order to verify experimentally the efficiency of algorithm *SPIC*, large collections of spanned patterns have been generated for a series of well-known datasets, frequently used in the data mining literature. The five datasets used, *Breast Cancer* (*bcw*), *Liver Disease* (*bld*), *Diabetes* (*pid*), *Heart Disease* (*hea*), *Congressional Voting* (*vot*), are publicly available at the website of the University of California Irvine (http://www.ics.uci.edu/~mlearn/MLRepository.html), and their basic parameters are presented below and summarized in **Table 1**.

Wisconsin breast cancer (*bcw*). In this dataset 683 observations (obtained after the removal of 16 instances which contain missing attributes) represent malignant or benign breast tissues, each observation being represented by 9 numerical attributes.

Congressional voting records (*vot*). This dataset contains the voting records of the 435 members of the U. S. House of Representatives of the 98th Congress, each being classified as a Democrat or a Republican. The 16 attributes represent the votes of the representatives on 16 issues, encoded as 1, 0 and 0.5, the latter corresponding to the absence of vote.

StatLog heart disease (*hea*). This dataset contains the records of 297 patients, indicating for each of them the presence or absence of heart disease, together with the numerical results of 13 medical tests, 7 of which have numerical expressions, the six other ones having binary outcomes.

BUPA liver disorders (*bld*). In this dataset 345 observations represent male patients, some of whom had liver disorder; each patient is represented by 6 numerical attributes describing alcohol consumption, and results of several blood tests.

PIMA Indian diabetes (*pid*). This dataset contains the records of 455 females of Pima Indian heritage living near Phoenix, Arizona, USA, of which 150 have the disease. Patients are characterized by the results of seven medical tests and physiological measurements.

It should be noted that a detailed analysis of the performance of various classification methods applied to these datasets is presented in [15]; while the average accuracy of various statistical classification methods was very high (in the range 90% - 96%) on the *bcw* and *vot* datasets, it was lower (between 70% and 85%) on the other three datasets.

Dataset	*# Observations*			*# Attributes*		
	Positive	*Negative*	*Total*	*Numerical*	*Categorical*	*Total*
bcw	**239**	**444**	**683**	**9**	**0**	**9**
vot	**267**	**168**	**435**	**0**	**16**	**16**
hea	**137**	**160**	**297**	**7**	**6**	**13**
bld	**200**	**145**	**345**	**6**	**0**	**6**
pid	**150**	**305**	**455**	**6**	**0**	**6**

Table 1. Basic parameters of datasets

Efficiency of *SPIC*. In order to illustrate the efficiency of algorithm *SPIC*, we present in **Table 2** the computing time needed for generating collections of positive and negative spanned patterns for the datasets described in **Table 1**. All the experiments were carried out on a 1.7 GHz processor. We have restricted the search to pure positive and pure negative patterns, and have required that the prevalences of the patterns to be generated exceed 5% (see **Table 2**), or 10% (see **Table 3**). In these tables, the notation N/A means that the total number of maximal pure patterns with the desired prevalence is below the required threshold.

Dataset	*# Spanned Patterns*			
	1000	**5000**	**10000**	**16000**
bcw	**3**	**25**	**47**	**93**
vot	**2**	**7**	**33**	**47**
hea	**8**	**26**	**94**	**197**
bld	**21**	**87**	**241**	**525**
pid	**24**	**94**	**142**	**206**

Table 2. Generation time for spanned patterns with prevalence at least 5%

Dataset	*# Spanned Patterns*			
	1000	**5000**	**10000**	**16000**
bcw	**4**	**28**	**51**	**99**
vot	**6**	**9**	**36**	**55**
hea	**14**	**49**	**101**	**311**
bld	**>1000**	**N/A**	**N/A**	**N/A**
pid	**>1000**	**N/A**	**N/A**	**N/A**

Table 3. Generation time for spanned patterns with prevalence at least 10%

Sensitivity of *SPIC* to discretization. We shall show next that the input consensus algorithm has another important quality, namely, it has a very low sensitivity to the increase of the discretization grid resolution. Originally, *LAD* was proposed [8] as a method for the analysis of binary (0-1) data. In [6] it was shown that the applicability of *LAD* can be extended to the analysis of problems with numerical data, by replacing each numerical variable by several binary ones. Also in [6] it was shown that the choice of a support set using a minimum number of 0-1 variables is an *NP*-hard problem.

In [4] it was shown that "discretizing" a numerical variable (i.e., replacing it with a new variable taking only the discrete values 0, 1,..., k, for some appropriately defined positive integer k) instead of binarizing it, has computational advantages. The basic idea of discretization is quite simple. For the variable x taking real values, the interval [$\min_\Omega x$, $\max_\Omega x$] is partitioned into the subintervals $[x_0, x_1)$, $[x_1, x_2)$,…, $[x_k, x_{k+1})$, where $x_0 = \min_\Omega x$, $x_{k+1} = \max_\Omega x$, and the original numerical variable x is replaced by a discretized variable x', by considering $x' = h$ if $x \in [x_h, x_{h+1})$.

A partitioning of $\mathbf{R}^n$ is *feasible* if each of the corresponding n dimensional subintervals is "homogeneous", i.e. does not contain both positive and negative observations. In order to arrive to homogeneous intervals in the discretization process, it is frequently necessary to increase the number of cutpoints defining a finer partition. At the same time, it is natural to try to keep the number of cutpoints reasonably small, in order to avoid both overfitting and computational difficulties. This is not always an easy task, since the determination of a feasible partitioning with a minimum number of subintervals is *NP*-hard, due to the result [6] mentioned above.

It is natural to assume that the complexity of finding patterns in a discretized space increases along with its refinement; for example, in [4] it was shown that the complexity of finding all the prime patterns in a dataset is proportional to the number of cutpoints used for discretization. A major advantage of the proposed *SPIC* algorithm for generating spanned patterns over other known algorithms ([6], [4] etc) is that its complexity does *not* increase with the number of cutpoints. However, an increase in the number of cutpoints may slightly increase the computational time, due to the larger data structures to be maintained, and the increase of allocated memory.

We show in **Table 4** the time needed for generating 1000 spanned patterns in the five datasets discussed above, when the number of subintervals in the partitioning of each of the numerical variables increases. It can be seen in the table that a 10 fold increase in the number of cutpoints (from 10 to 100) for each one of the components (which implies an increase of between 6^{90} and 16^{90} in the number of subintervals), increases the average running time required by the algorithm *SPIC* by only 47%.

Dataset	*# Attributes*	*# Cutpoints / Component*		
		10	**20**	**100**
bcw	**9**	**3**	**4**	**5**
vot	**16**	**9**	**N/A**	**N/A**
hea	**13**	**15**	**20**	**23**
bld	**6**	**21**	**29**	**33**
pid	**6**	**82**	**88**	**92**

Table 4. Generation time (sec) for 1000 spanned patterns

6 Robustness of Spanned Patterns

In *LAD*, patterns are used as inference rules in the modeling process, and therefore, it is very desirable that their qualities (e.g. prevalence, homogeneity) do not degrade when applied to test data. In this section we show that the spanned patterns -- generated by algorithm *SPIC* -- are highly robust, i.e., the deterioration of their qualities on test sets is relatively small. In order to illustrate this fact, a series of computational experiments were carried out on the five datasets described in Section 5, for analyzing these changes in prevalence and homogeneity.

The validation process consisted in “two-folding” experiments, in which the dataset was partitioned randomly into a "training set" containing 50% of the observations, and a "test set" containing the other 50%. The observations in the training set were used for finding lists of pure spanned patterns, which were validated on the test set. Afterwards, the experience was repeated by using the old test set as training set, and validating the results on the old training set, used as a test set. This procedure was applied for each of the datasets five times, providing in this way 10 validation experiments. All the results reported in this section represent averages for the 10 experiments.

It can be seen in **Table 5** that for the “separable” datasets *bcw* and *vot* (which are known to admit clean separations into positive and negative observations), the average drops in prevalence and homogeneity for spanned patterns are very low (at most 5%). While for the “inseparable” datasets *hea*, *bld*, and *pid* (which are known not to admit clean separations), the average drops in prevalence and homogeneity are somewhat higher, they still remain in a reasonable range (between 12% and 20%). Overall, the average drop in prevalence is 12.10%, and the average drop in homogeneity is only 10.60%.

Pattern / *DataSet*	*Percentage of Decrease*	
	Prevalence	***Homogeneity***
bcw	**5**	**4**
vot	**4**	**5**
hea	**17**	**12**
bld	**20**	**19**
pid	**15**	**14**
average	**12.10**	**10.60**
stdev	**7.39**	**6.55**

Table 5. Spanned patterns: decrease of prevalence and homogeneity from training to test sets

In order to check whether high robustness is a specific quality of spanned patterns, we repeated the above experiment for prime patterns. More precisely, for each of the 5 datasets we generated 1000 spanned patterns, determined the subsets of maximal spanned patterns, and using then a simple heuristic, we associated to each maximal spanned pattern *P* the collection of all its "reduced" patterns, i.e., prime patterns having the same coverage as *P*.

For comparison, **Table 6** shows the average drop in prevalence and homogeneity for the reduced patterns of low and high degree. We remark that for the "inseparable" datasets *hea*, *bld*, and *pid* the average drop in prevalence is significantly higher for the reduced patterns (22.88%) than for the spanned ones (17.33%), and the drop is even higher for the low degree reduced patterns (25.00%). The "separable" datasets *bcw* and *vot* perform much better than the "inseparable" ones, the average drop in prevalence and homogeneity being only 5.83% and 4.50%, respectively.

Overall, the reduced patterns have an average drop in prevalence and homogeneity of 15.57%, which is higher than the corresponding average drop for the spanned patterns (11.35%).

Pattern / ***Dataset***	***Percentage of Decrease***					
	Prevalence			***Homogeneity***		
	deg ≤ 3	***deg = 4***	***deg ≥5***	***deg ≤ 3***	***deg = 4***	***deg ≥5***
bcw	**7**	**2**	**4**	**4**	**1**	**1**
vot	**8**	**8**	**6**	**6**	**7**	**8**
hea	**22**	**20**	**19**	**13**	**14**	**13**
bld	**25**	**24**	**22**	**28**	**27**	**26**
pid	**28**	**24**	**22**	**27**	**25**	**25**
average	**18**	**16**	**15**	**15**	**15**	**15**
stdev	**10**	**10**	**9**	**12**	**11**	**11**

Table 6. Decrease in prevalence and homogeneity of reduced patterns from training to test sets

Finally, **Table 7** shows the percentage of reduced patterns of low degree (i.e. at most 3) and of high degree (i.e. at least 4) obtained in this way. Generally, spanned patterns have high degrees (equal in average to about 75% of the number of attributes in the dataset). From **Table 7** we remark that while for the "separable" datasets the percentage of low degree reduced patterns is high (more than 90% for *bcw* and *vot*), for the "inseparable" datasets the percentage of low degree reduced patterns is considerably smaller (up to 26% for *hea, bld*, and *pid*).

Dataset	***# Attributes***	***% of Maximal Spanned Patterns***	***% of Reduced Patterns***		
			deg ≤ 3	***deg = 4***	***deg ≥ 5***
bcw	**9**	**25**	**96**	**4**	**0**
vot	**16**	**9**	**90**	**6**	**4**
hea	**13**	**65**	**26**	**34**	**40**
bld	**6**	**77**	**19**	**40**	**41**
pid	**6**	**67**	**9**	**23**	**68**

Table 7. Percentage of maximal and reduced patterns extracted from 1000 spanned patterns

7 Conclusions

The main conclusions of this study concern:

A. *The importance of spanned patterns in data analysis*

(i) The class of spanned patterns is remarkably robust, i.e., the decrease in the prevalence and homogeneity of spanned patterns in test sets compared to training sets is low, thus justifying their use in LAD models.

(ii) It is known from [] that LAD models based on spanned patterns have fewer classification errors on test sets than those based on prime patterns, although the number of unclassified observations may be somewhat higher in the spanned pattern-based models.

(iii) Efficient methods are known for the enumeration of low degree prime patterns, but these methods are difficult to apply for the generation of higher degree prime patterns. Since the prevalence of low degree prime patterns in "inseparable" (i.e. low-quality) datasets is very low, LAD models for such datasets must be built on spanned (rather than prime) patterns.

B. *The efficiency of the proposed spanned pattern enumeration algorithms*

(iv) SPAN and SPIC provide the first systematic enumeration methods of spanned patterns.

(v) The running time of both SPAN and SPIC is linear in the output. Moreover, SPIC runs in total polynomial time.

(vi) The reported computational experiments confirm the high efficiency of SPIC.

(vii) The computational experiments also show the remarkable stability of SPIC with respect to the number of cutpoints used for discretizing continuous data.

References

[1] Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P.L., and Simeone, B. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics* (in print).

[2] Alexe, G., Alexe, S., Hammer, P.L., and Kogan, A. Comprehensive vs. Comprehensible Classifiers in Logical Analysis of Data. Rutgers University, *RUTCOR Research Report*, RRR 9-2002.

[3] Alexe, S., and Hammer, P. L. Accelerated Algorithm for Pattern Detection in Logical Analysis of Data. Rutgers University, *RUTCOR Research Report*, RRR 59-2001.

[4] Alexe, S., Blackstone, E., Hammer, P. L., Ishwaran, H., Lauer, M. S., and Pothier Snader, C. E. Coronary Risk Prediction by Logical Analysis of Data. *Annals of Operations Research* 2003 (in print).

[5] Blake, A. *Canonical expressions in Boolean algebra*. Ph.D. Thesis. University of Chicago, 1937.

[6] Boros, E., Hammer, P.L., Ibaraki, T., and Kogan, A. Logical Analysis of Numerical Data. *Mathematical Programming* **79** (1997), 163-190.

[7] Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I. An Implementation of Logical Analysis of Data. *IEEE Transactions on Knowledge and Data Engineering*. **12**, No. **2** (2000), 292-306.
[8] Crama, Y., Hammer, P.L., and Ibaraki, T. Cause-Effect Relationships and Partially Defined Boolean Functions. *Annals of Operations Research* **16** (1988), 299-326.
[9] Eckstein, J., Hammer, P.L., Liu, Y., Nediak, M., Simeone, B. The Maximum Box Problem and its Application to Data Analysis. *Computational Optimization and Applications* (in print).
[10] Foldes, S., and Hammer, P.L. Disjunctive and Conjunctive Normal Forms of Pseudo-Boolean Functions. *Discrete Applied Mathematics* **107** (2000), 1-26.
[11] Hammer, P.L. Partially Defined Boolean Functions and Cause-Effect Relationships. *International Conference on Multi-Attribute Decision Making Via OR-Based Expert Systems*, University of Passau, Passau, Germany, (1986).
[12] Hammer P.L. , Kogan A., Simeone B., and Szedmak S. Pareto-optimal patterns in Logical Analysis of Data. *RUTCOR Research Report*, Rutgers University, 7-2001, January 2001, *Discrete Applied Mathematics* (in print).
[13] Johnson, D.S., Yannakakis, M., and Papadimitriou, C.H. On generating all maximal independent sets. *Information Processing Letters* **27** No. **3** (1988), 119--123.
[14] Lauer, M. S., Alexe, S., Snader, C. E. P., Blackstone, E. H., Ishwaran, H., and Hammer, P. L. Use of the "Logical Analysis of Data" Method for Assessing Long-Term Mortality Risk After Exercise Electrocardiography. *Circulation* **106** (2002), 685-690.
[15] Lim, T.-S., Loh, W.-Y., and Shin, Y.-S. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. *Machine Learning*, **40** (2000) 203-229.
[16] Malgrange, Y. *Recherche des sous-matrices premières d'une matrice à coefficients binaires. Applications à certains problèmes de graphe*. In Deuxième Congrès de l'AFCALTI, October 1961, Gauthier-Villars, 1962, 231-242.
[17] Quine, W. A way to simplify truth functions. *American Mathematical Monthly*, **62** (1955), 627-631.