RUTCOR
RESEARCH
REPORT

Maximum Patterns in Datasets

T. O. Bonates[a] P. L. Hammer[b] A. Kogan[c,d]

RRR 9, 2006

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]RUTCOR – Rutgers University Center for Operations Research, Piscataway, NJ, USA, e-mail: tbonates@rutcor.rutgers.edu
[b]RUTCOR – Rutgers University Center for Operations Research, Piscataway, NJ, USA, e-mail: hammer@rutcor.rutgers.edu
[c]RUTCOR – Rutgers University Center for Operations Research, Piscataway, NJ, USA
[d]Rutgers Business School, Newark-New Brunswick, NJ, USA, e-mail: kogan@rutgers.edu

Rutcor Research Report
RRR 9, 2006

Maximum Patterns in Datasets

T. O. Bonates P. L. Hammer A. Kogan

Abstract. Given a binary dataset of positive and negative observations, a positive (negative) *pattern* is a subcube having a nonempty intersection with the positive (negative) subset of the dataset, and an empty intersection with the negative (positive) subset of the dataset. Patterns are the key building blocks in *Logical Analysis of Data (LAD)*, and are an essential tool in identifying the positive or negative nature of "new" observations covered by them. We develop exact and heuristic algorithms for constructing a pattern of maximum coverage which includes a given point. It is shown that the heuristically constructed patterns can achieve more than 98% of the maximum possible coverage, while requiring only a fraction of the computing time of the exact algorithm. Maximum patterns are shown to be useful for constructing highly accurate LAD classification models. In comparisons with the commonly used machine learning algorithms implemented in the publicly available Weka software package, the implementation of LAD using maximum patterns is shown to be a highly competitive classification method.

Acknowledgements: T. Bonates gratefully acknowledges the partial support of a DIMACS Graduate Student Award. We acknowledge the assistance provided by Dash Optimization by allowing the use of its integer and linear programming solver Xpress-MP within its Academic Partnership Program.

1 Introduction

Patterns are the key building blocks in *Logical Analysis of Data (LAD)* (*e.g.*, see [1], [2], [3], [5] and [9]), and have been shown in numerous studies to provide important indications about the positive or negative nature of the points "covered" by them. The collection of patterns used in the implementation of LAD (see e.g. [5]) is generated by a combinatorial enumeration process, which can be quite expensive computationally. The number and type of patterns generated in this process are controlled by several parameters, which impose limitations on the "degree" of the patterns produced, their "coverage", and their "fuzziness". The choice of the most appropriate parameter values is quite involved and time-consuming, being based on numerous computational experiments. Moreover, even with the best choice of control parameter values the size of the pattern collection produced is very large and requires in most cases the application of a "filtering" procedure, which selects small subsets of patterns to form highly accurate predictive models.

In this study we shall address the complexities of the pattern generation process by introducing the concept of "maximum" patterns, the number of which is naturally bounded by the number of observations in the dataset. We propose a discrete optimization based exact algorithm, and highly efficient heuristics, for generating maximum patterns, and show that the accuracy of LAD models based on these patterns is highly competitive with that of the original LAD models, as well as with those of the best commonly used classification methods.

In Section 2 we introduce the basic notation used throughout the text. In Section 3 we develop an integer programming formulation for the construction of exact maximum patterns, and in Section 4 we propose three heuristics for constructing approximately maximum patterns. Sections 5 and 6 describe natural extensions of the concept of patterns to datasets with missing attribute values and possible misclassification noise, showing how to modify the pattern generation algorithms to handle these cases. Section 7 presents computational results concerning the generation of exact and heuristic patterns. Section 8 evaluates the accuracy of LAD models built using one or more of the algorithms described in Sections 3 and 4, and shows how their accuracies compare to those of some commonly used classification algorithms. In Section 9 we discuss the comparative accuracy of the LAD models using maximum patterns, and argue in favor of the use of a combination of two of the proposed heuristic algorithms as an efficient pattern generation procedure for constructing accurate LAD models.

2 Notation

In the case of a binary dataset $\Omega = \Omega^+ \cup \Omega^- \subset \{0,1\}^n$, with $\Omega^+ \cap \Omega^- = \emptyset$, a *pattern* is simply a homogeneous subcube, *i.e.* a subcube having (i) a nonempty intersection with one of the sets Ω^+ or Ω^-, and (ii) an empty intersection with the other set (Ω^- or Ω^+, respectively). We recall that a subcube consists of those points of the n-cube for which a subset of the variables is fixed to 0 or 1, while the remaining variables take all the possible 0,1 values.

We shall refer to Ω^+ as the set of *positive* points of the dataset Ω, and to Ω^- as the set of *negative* points of the dataset. A pattern P disjoint from Ω^- is called a *positive* pattern, while a pattern P' disjoint from Ω^+ is called a *negative* pattern.

A positive α-pattern for $\alpha \in \{0,1\}^n$, is a pattern *covering* (i.e., containing) α. A *maximum positive α-pattern* P is a positive α-pattern, for which its *coverage* (i.e., the cardinality of $|P \cap \Omega^+|$) is maximum. A maximum negative α-pattern is defined in a similar way.

The basic assumption of *LAD* is that a binary point covered by some positive patterns, but not covered by any negative pattern is positive, and similarly, a binary point covered by some negative patterns, but not covered by any positive pattern is negative.

Previous experience with *LAD* ([5], [9]) has shown that patterns with higher coverage provide better indication of the positive or negative character of new observations than those with lower coverage. This observation motivates the focus of this study on maximum patterns, their construction, and their use in classification.

3 Construction of Exact Maximum Patterns (EMPs)

Given the sets Ω^+, Ω^-, we shall be concerned here with ways of finding a maximum positive α-pattern, $\alpha \in \{0,1\}^n \setminus \Omega^-$. The determination of a maximum negative α-pattern can be done in a symmetric way. In view of the perfect symmetry of positive and negative α-patterns we shall describe the proposed methodology only for the positive case.

In order to formulate the maximum α-pattern problem as an integer program, we shall introduce a binary decision variable y_j which describes whether or not the value of the j-th variable of the pattern is fixed to α_j. With this notation, the condition that the α-pattern should not include any negative point requires that for every point γ of Ω^-, the variable y_j should take the value 1 for at least one of those j's for which $\gamma_j \neq \alpha_j$, i.e.

$$\sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^{n} y_j \geq 1, \quad \text{for every } \gamma \in \Omega^-. \tag{1}$$

On the other hand, a positive point β will be covered by the α-pattern if and only if $y_j = 0$, for all those indices j for which $\beta_j \neq \alpha_j$. Therefore, the number of positive points covered by the α-pattern will be given by

$$\sum_{\beta \in \Omega^+} \prod_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^{n} \overline{y}_j, \tag{2}$$

where $\overline{y}_j = 1 - y_j$, for every $j = 1, \ldots, n$. In conclusion, the maximum α-pattern problem can be formulated as the following nonlinear integer program

$$
\begin{aligned}
\textit{maximize} \quad & \sum_{\beta \in \Omega^+} \prod_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^{n} \overline{y}_j \\
\textit{subject to} \quad & \sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^{n} y_j \geq 1, \quad \text{for every } \gamma \in \Omega^- \\
& y_j \in \{0,1\}, \quad \text{for every } j = 1, \ldots, n.
\end{aligned} \tag{3}
$$

This problem is a generalized set covering problem. Indeed, in the special case when Ω^+ consists of the n points $\beta^{(i)}$, where $\beta^{(i)}$ differs from α only in variable i, then the objective function becomes simply $n - \sum_{j=1}^{n} y_j$, which is equivalent to minimizing $\sum_{j=1}^{n} y_j$, i.e. to a standard set covering problem. In view of this remark it is clear that this problem is $\mathcal{NP}$-hard and hence no polynomial algorithm is available for its solution. Moreover, it has been shown [8] that the set covering problem is not approximable within $c \log n$, for any positive c.

Since numerous software packages are available for solving integer linear programs, it is useful to rewrite (3) in this form. This can be achieved by introducing a new binary variable z_β to replace each term of the objective function of (3). Clearly, in the case of 0/1 variables, the relation

$$
z_\beta = \prod_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^{n} \overline{y}_j
$$

is equivalent with the system of inequalities (4) and (5):

$$
w(\beta) z_\beta \leq \sum_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^{n} \overline{y}_j, \text{ for every } \beta \in \Omega^+, \tag{4}
$$

$$
\sum_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^{n} \overline{y}_j \leq z_\beta + w(\beta) - 1, \text{ for every } \beta \in \Omega^+, \tag{5}
$$

where

$$
w(\beta) = |\{j : \beta_j \neq \alpha_j\}|. \tag{6}
$$

Therefore, the nonlinear integer program (3) is equivalent to the linear integer program (EMP):

$$
\begin{aligned}
&\textit{maximize} && \sum_{\beta\in\Omega^+\setminus\{\alpha\}} z_\beta \\
&\textit{subject to} && \\
&\sum_{\substack{j=1\\ \gamma_j\neq\alpha_j}}^{n} y_j && \geq \quad 1, \text{ for every } \gamma\in\Omega^- \qquad (7)\\
w(\beta)z_\beta + &\sum_{\substack{j=1\\ \beta_j\neq\alpha_j}}^{n} y_j && \leq \quad w(\beta), \text{ for every } \beta\in\Omega^+\setminus\{\alpha\},\\
z_\beta + &\sum_{\substack{j=1\\ \beta_j\neq\alpha_j}}^{n} y_j && \geq \quad 1, \text{ for every } \beta\in\Omega^+\setminus\{\alpha\},\\
& && y_j\in\{0,1\}, \text{ for every } j=1,\ldots,n,\\
& && z_\beta\in\{0,1\}, \text{ for every } \beta\in\Omega^+\setminus\{\alpha\}.
\end{aligned}
$$

While the integer linear program (7) has the same feasible solutions as (3), the omission of the constraints (5) from (7) results in a new integer linear program, which may have a larger set of feasible solutions, but has exactly the same set of optimal solutions as (7), and therefore as (3). However, from the computational efficiency point of view this simplification is not beneficial and was not used in our experiments.

4 Heuristics

The number of binary variables appearing in problem (7) is $n + |\Omega^+| - 1$, which in case of large datasets results in very large integer linear programs. In view of the computational difficulty of handling such large integer linear programs, it is important to develop appropriate heuristics to deal with instances for which current integer linear programming software packages fail to find exact solutions to problem (7). In order to achieve this objective, two heuristic approaches will be presented below.

Section 4.1 will describe an approach based on replacing the original objective function of (3) by its best linear approximation in L_2, while Sections 4.2 and 4.3 will present two implementations of a greedy combinatorial heuristic in which the coverage of a positive α-pattern is successively increased in order to find a maximal (rather than maximum) positive α-pattern. Computational experiments with the proposed heuristics will be presented in Section 7.

4.1 Best Linear Approximation (BLA)

We shall describe in this section a model based on the nonlinear set covering model (3) in which we shall replace the objective function by its best linear approximation in L_2 in order to reduce it to the usual format of a weighted (linear) set covering problem. The determination of the L_2-best linear approximation will be based on the results of [10].

The objective function of (3) is a real-valued function in binary variables, i.e. a *pseudo-Boolean function*. If we represent a pseudo-Boolean function $f(u_1, u_2, \ldots, u_m)$ as

$$f(u_1, u_2, \ldots, u_m) = \sum_{j=1}^{s} \left(c_j \prod_{i \in S_j} u_i \right),$$

where c_j are real numbers, and S_j are subsets of $\{1, 2, \ldots, m\}$, then the L_2-best linear approximation $L(f)$ of f is known ([10]) to be given by

$$L(f(u_1, u_2, \ldots, u_m)) = \sum_{j=1}^{s} c_j \; L(\prod_{i \in S_j} u_i).$$

Further, it was shown in [10] that

$$L(\prod_{i \in S_j} u_i) = -\frac{|S_j| - 1}{2^{|S_j|}} + \frac{1}{2^{|S_j|-1}} \left(\sum_{i \in S_j} u_i \right).$$

Therefore, the L_2-best linear approximation of f is given by

$$L(f(u_1, u_2, \ldots, u_m)) = \sum_{j=1}^{s} c_j \left(-\frac{|S_j| - 1}{2^{|S_j|}} + \frac{1}{2^{|S_j|-1}} \left(\sum_{i \in S_j} u_i \right) \right),$$

and it can be seen that its computation does not present any difficulties.

Applying the above formula to the objective function of (3) we find that its L_2-best linear approximation is given by

$$\sum_{\beta \in \Omega^+} \frac{w(\beta) + 1}{2^{w(\beta)}} - \sum_{j=1}^{n} \left(\sum_{\substack{\beta \in \Omega^+ \\ \beta_j \neq \alpha_j}} \frac{1}{2^{w(\beta)-1}} \right) y_j.$$

Since the coefficient of every variable y_j is nonpositive, we shall approximate problem (3) by the following weighted (linear) set covering problem:

$$
\begin{aligned}
&\textit{minimize} \\
&\qquad \sum_{j=1}^{n} \left(\sum_{\substack{\beta \in \Omega^+ \\ \beta_j \neq \alpha_j}} \frac{1}{2^{w(\beta)-1}} \right) y_j \\
&\textit{subject to} \\
&\qquad \sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^{n} y_j \geq 1, \quad \text{for every } \gamma \in \Omega^- \\
&\qquad y_j \in \{0,1\}, \quad \text{for every } j = 1, \ldots, n.
\end{aligned} \tag{8}
$$

Clearly, the optimal solution of (8) will define a positive α-pattern whose size will provide a lower bound to the size of a maximum positive α-pattern. The computational experiments to be presented in Section 7 will show that this bound provides a good approximation of the maximum size of a positive α-pattern.

We recall that problem (8) is NP-complete and, therefore, in principle, its solution can become computationally intractable. However, many commercially available integer linear programming solvers are capable of solving fairly large size weighted set covering problems in an acceptable amount of time. Moreover, since most standard machine learning datasets involve only a relatively small number of attributes, the corresponding problems (8) can be solved fairly quickly.

4.2 Enlarging Patterns to Maximized Prime Patterns (MPP)

Given a point α, the associated *minterm* $\mathcal{P}_\alpha$ is defined as $\bigwedge_{i=1}^{n} x_i^{\alpha_i}$, where

$$
x_i^{\alpha_i} = \begin{cases} x_i, & \text{if } \alpha_i = 1; \\ \overline{x}_i, & \text{if } \alpha_i = 0. \end{cases}
$$

Clearly, if $\alpha \in \Omega^+$ then P_α is a positive α-pattern, covering only the point α. Each pattern can be represented as a Boolean conjunction $\bigwedge_{j \in S} x_j^{\beta_j}$, $S \subseteq \{1, \ldots, n\}$, of a subset of literals (i.e. complemented and non-complemented variables). Minterms are those positive patterns for which $S = \{1, \ldots, n\}$.

A positive pattern $\bigwedge_{j \in S} x_j^{\beta_j}$ is called *prime* if the conjunction $\bigwedge_{j \in S'} x_j^{\beta_j}$ is not a positive pattern for any proper subset $S' \subset S$. It is known from experience that many of the prime patterns have very large coverages. Because of this it makes sense to transform a minterm $\mathcal{P}_\alpha$ into a positive prime α-pattern.

If a positive pattern $\mathcal{P}$ is not a prime pattern then we can apply to it the iterative algorithm described in [9], which transforms a given pattern into a prime pattern. Starting from $\mathcal{P}$ we shall obtain a prime pattern $P = \bigwedge_{j \in S} x_j^{\alpha_j}$, for some $S \subset \{1, \ldots, n\}$, by successively removing literals from $\mathcal{P}$ so as to maximize the coverage of the resulting pattern.

In order for the algorithm to construct a prime pattern of large coverage a heuristic criterion is used to choose the literal to be removed at each iteration. The removal of a literal is considered to be advantageous if the resulting pattern is "closer" to the set of positive points not covered by it than to the set of negative points.

In order to specify the heuristic, let us define the *disagreement* between a point β and a pattern P to be the number of literals of P whose values are zero on β. The disagreement between a set of points and a pattern is simply the sum of the disagreements between the pattern and every point in the set. Let us denote by $d_+(P)$ the disagreement between P and the set of positive points not covered by it. Similarly, let us denote by $d_-(P)$ the disagreement between the pattern and the negative points. Our computational experiments suggest that the ratio $\frac{d_+(P)}{d_-(P)}$ provides a good criterion for choosing the literal to be removed at each step. We describe in Figure 1 a pseudocode for this algorithm. Obviously this algorithm can be restated for the construction of negative α-patterns.

1. Input: $\Omega^+, \Omega^-, \mathcal{P}(S) = \bigwedge_{i \in S} x_i^{\alpha_i}$ – positive pattern.
2. For every $k \in S$ let $\mathcal{P}_k(S) = \bigwedge_{i \in S \setminus \{k\}} x_i^{\alpha_i}$.
3. If there is no $k \in S$ such that $\mathcal{P}_k(S)$ is a positive pattern
then stop and output $\mathcal{P}(S)$.
4. Choose a $k \in S$ such that $\mathcal{P}_k(S)$ is a positive pattern and $\frac{d_+(\mathcal{P}_k(S))}{d_-(\mathcal{P}_k(S))}$ is minimized.
5. Set $S := S \setminus \{k\}$, $\mathcal{P}(S) = \bigwedge_{i \in S} x_i^{\alpha_i}$, and go to Step 2.

Figure 1: Algorithm for Enlarging Patterns to Maximized Prime Patterns (MPP).

This algorithm can be used to enlarge a minterm $\mathcal{P}_\alpha$ to a prime α-pattern, or more generally, to enlarge patterns generated by other algorithms to prime ones.

One can easily see that the time complexity of this algorithm is $O(|\Omega|n^2)$. Therefore, constructing a heuristic maximum prime pattern for every point in the dataset takes $O(|\Omega|^2 n^2)$ time.

4.3 Enlarging Patterns to Maximized Strong Patterns (MSP)

Let $\alpha \in \Omega^+$ and $\mathcal{P}$ be a positive α-pattern. We denote by $Cov(\mathcal{P})$ the set of points covered by $\mathcal{P}$, and denote by $Lit(\mathcal{P})$ the set of literals defining $\mathcal{P}$, i.e. $\mathcal{P} = \bigwedge_{i \in Lit(\mathcal{P})} x_i^{\alpha_j}$.

A positive pattern $\mathcal{P}$ is called *strong* if there is no positive pattern $\mathcal{P}'$ such that $Cov(\mathcal{P}') \supset Cov(\mathcal{P})$. It is known (see e.g. [9]) from experience that many strong patterns have very large coverage and their use in LAD leads to a superior performance. If $\mathcal{P}$ is not a strong pattern, we can apply the iterative algorithm described in [9] to transform it to a strong pattern $\mathcal{P}'$, such that $Cov(\mathcal{P}') \supset Cov(\mathcal{P})$.

Let S be a non-empty subset of Ω^+, and let $[S]$ be the convex hull of the points in S, i.e. the smallest subcube containing S. A pattern $\mathcal{P}$ is called *spanned* if $\mathcal{P} = [Cov(\mathcal{P})]$. As shown in [9] this definition is equivalent to saying that if I is the set of those indices i for

which the corresponding components of all points of $Cov(\mathcal{P})$ have the same value, say β_i, then

$$\mathcal{P} = \bigwedge_{i \in I} x_i^{\beta_i}.$$

The general algorithm described in [9] can be adapted so as to produce patterns of large coverage. For this purpose we use a heuristic criterion to choose the next point to be included in the coverage of the current pattern $\mathcal{P}$. The criterion selects a point $\beta \in \Omega^+ \setminus Cov(\mathcal{P})$ such that $[Cov(\mathcal{P}) \cup \{\beta\}]$ is a positive pattern, and $|Lit([Cov(\mathcal{P}) \cup \{\beta\}])|$ is maximized. We describe in Figure 2 a pseudocode for this algorithm.

1. Input: $\Omega^+, \Omega^-, \alpha \in \Omega^+, \mathcal{P}$ – positive α-pattern, $S = Cov(\mathcal{P})$.
2. $\mathcal{P}(S) := [S]$.
3. For every $\beta \in \Omega^+ \setminus S$ let $\mathcal{P}_\beta(S) = [S \cup \{\beta\}]$.
4. If there is no point $\beta \in \Omega^+ \setminus S$ such that
$\mathcal{P}_\beta(S)$ is a positive pattern
then stop and output $\mathcal{P}(S)$.
5. Choose $\beta \in \Omega^+ \setminus S$ such that $\mathcal{P}_\beta(S)$ is a positive pattern and $|Lit(\mathcal{P}_\beta(S))|$ is maximized.
6. Set $S := Cov(\mathcal{P}_\beta(S))$ and go to Step 2.

Figure 2: Algorithm for Enlarging Patterns to Maximized Strong Patterns (MSP).

It is obvious from the definition that the pattern generated by this algorithm is not only strong, but also spanned. Furthermore, note that, since $\mathcal{P} = [Cov(\mathcal{P})]$ (according to Theorem 4.5 in [9]), after $\mathcal{P}_\beta(S)$ is computed in Step 3, no additional computation is required in Step 2 of the following iteration.

A straightforward way to generate a strong α-pattern is to apply this algorithm to the minterm $\mathcal{P}_\alpha$. Another way of using this algorithm is to apply it to the patterns generated by any other pattern generating algorithms, e.g. the two heuristics described above, and thus possibly achieving an increase in the coverage of the patterns produced by them.

It is clear that the algorithm can be restated to construct negative α-patterns. One can easily see that the time complexity of this algorithm for constructing a heuristic maximum strong positive α-pattern is $O(|\Omega^+|^2|\Omega^-|n)$. Therefore, constructing a heuristic maximum strong pattern for every point in the dataset takes $O(|\Omega|^2|\Omega^+||\Omega^-|n)$ time.

4.4 Combined Heuristic Algorithms (CHAs)

Computational experiments show that in the case of large problems it takes a substantially shorter time to run *all* the heuristics described above than to solve problem (7) using a standard integer programming package. Thus, whenever the running time of solving problem (7) becomes prohibitively long, a computationally affordable alternative is to run *all* the heuristics, and combine their results, as described below.

A so-called "combined heuristic algorithm" (CHA) consists in: (i) choosing a subset of heuristics to use, (ii) running the chosen heuristics for every point in the dataset, and (iii) selecting for each $\alpha \in \Omega^+$ (respectively, Ω^-) a positive (respectively, negative) α-pattern of largest coverage from the collection of all patterns constructed in step (ii).

5 Fuzzy Patterns

The concept of patterns discussed in the previous sections is based on the ideal assumption that the historical information is perfectly correct. More specifically, it is assumed that, on the one hand, all the attribute values are measured precisely, and on the other hand, all the classifications of observations are recorded correctly. In most real-life situations these ideal assumptions do not hold. This fact can be seen in the evaluation of patterns on a testing set, showing that positive α-patterns of large coverage (on the training set) cover on the testing set not only a significant number of positive observations but also a (usually small) number of negative observations.

In view of this fact, it is reasonable to allow large patterns to cover a "small" number of observations of the opposite class. It is to be expected that such a relaxation of constraints defining positive (negative) patterns should lead to a significant increase in the coverage of positive (negative) observations. We shall refer to those patterns that do not cover any observations of the opposite class as "pure", while the other ones shall be called "fuzzy".

The fuzziness of a pattern will be measured by a parameter which determines how many observations of the opposite class are covered by that pattern. A family of positive (negative) patterns is said to have *fuzziness* φ if the percentage of negative (positive) observations covered by each pattern in the family does not exceed φ.

The construction of fuzzy (positive) patterns can be accomplished by a simple modification of the constraints of model (7). The only constraints that have to be modified are those which have to hold for every $\gamma \in \Omega^-$; more precisely, for these constraints we shall require that

$$\sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^{n} y_j \geq 1 - s_\gamma, \quad \text{for every } \gamma \in \Omega^-$$

$$s_\gamma \in \{0,1\}, \text{ for every } \gamma \in \Omega^-,$$

and

$$\sum_{\gamma \in \Omega^-} s_\gamma \leq \varphi |\Omega^-|.$$

Exactly the same modification carries over to the best linear approximation-based heuristic, formulated as problem (8).

It is fairly straightforward to generalize the two combinatorial heuristic algorithms – Algorithm for Enlarging Patterns to Prime Patterns, and Algorithm for Enlarging Patterns to Strong Patterns – to the case of fuzzy patterns. The only modification to be made in the formulation of the algorithms is to replace everywhere "positive pattern" by "positive pattern with fuzziness at most φ".

Note that the utilization of fuzzy patterns allows to relax the assumption that Ω^+ and Ω^- are disjoint, by the weaker assumption that their intersection is "relatively small".

6 Robust Patterns

A common occurrence in real world datasets is the absence of the values of some of the attributes in certain observation points. This can be due either to missing information, or to measurement imprecisions leading to missing values when numerical data are binarized. Some of the standard approaches to dealing with such datasets consist either in the removal from the dataset of the observations or of the attributes with missing values, or in the filling in of the missing values with estimates obtained in a variety of ways (e.g. by using average values). While both approaches transform the original dataset with missing values into a fully specified one, the former approach discards possibly valuable information from the dataset, while the latter one introduces poorly justified modifications in the data. Since the reduction of datasets with missing values to completely specified ones is not satisfactory, we propose instead to work directly with datasets with missing values by appropriately extending the concept of patterns.

The concept of *robust patterns* (see [5]) extends that of patterns to the case of datasets with missing attribute values, always assuming a worst-case scenario. The worst-case assumption concerns the way of defining whether a pattern covers an observation with missing attribute values. More specifically, on the one hand, a positive observation with a missing attribute value in any of the variables appearing in a robust positive pattern will be considered not to be covered by that pattern (since the actual attribute value may conflict with the literal in the pattern). On the other hand, a negative observation with missing attribute values will be considered covered by a robust positive pattern if there is a combination of missing attribute values for which the corresponding completed observation is covered by that pattern. For example, the coverage of the positive pattern $x_1\overline{x}_2$ does not include the positive point $(1, -, 0)$, but does include the negative point $(-, 0, 1)$.

7 Computational Evaluation of Maximum Pattern Generation Algorithms

Our computational experiments were carried out on five publicly available datasets (Breast Cancer, Pima Indian Diabetes, Heart Disease, Liver Disease and Congressional Voting) from the UC Irvine Repository [4]. The basic parameters of these datasets are presented in Table

1.

Dataset		# of Observations		# of Attributes	
		Positive	Negative	Original	Binarized
Breast Cancer	(BCW)	241	458	9	20
Bupa	(BLD)	200	145	6	31
Diabetes	(DIA)	268	500	8	27
Heart	(HEA)	139	164	13	17
Voting	(VOT)	264	165	16	16

Table 1: Parameters of datasets.

We report in Table 2 the quality of the patterns obtained by the three heuristics described in Section 4, as well as the combinations CHA of MPP, MSP and BLA (referred to simply as CHA), and of MPP and MSP (referred to as MPSP). The quality of heuristically generated patterns is expressed as the average percentage of the number of points covered by these patterns compared to the number of points covered by the optimum patterns constructed by solving exactly the corresponding integer programming problem (7). In our experiments we used the Xpress-MP solver of Dash Optimization [6] for the solution of integer programming problems.

Dataset	MPP	MSP	BLA	CHA	MPSP
BCW	97.1%	89.6%	96.5%	99.8%	98.0%
BLD	66.2%	67.3%	80.4%	98.0%	78.6%
DIA	82.6%	69.8%	80.9%	93.9%	90.4%
HEA	90.1%	72.7%	91.6%	98.7%	93.3%
VOT	99.0%	78.0%	96.2%	99.9%	99.6%
Average	87.0%	75.5%	89.1%	98.1%	92.0%

Table 2: Relative sizes of heuristically generated patterns.

We report in Table 3 the average running times (in seconds) used to find the optimum patterns by solving exactly the integer programming problem (7). The table also presents for all 5 datasets considered here the average running times of the proposed heuristic procedures, as a percentage of the running time of the exact algorithm. The computer used to run the maximum pattern generation algorithms was an *Intel Pentium*® 4, 3.4GHz, 2GB of RAM. The maximum pattern generation algorithms were implemented using the *MinGW* compiler[1]. Problems (7) and (8) were solved using version 16.10.02 of the XpressMP solver.

It can be seen that the heuristic algorithms produced patterns whose average coverages ranged from 75% to 98% of those of the optimum patterns. The average time needed by the heuristics ranged from 6% to 40% of the time needed by the exact algorithm.

[1]http://www.mingw.com/

Dataset	EMP (sec)	Running time as % of EMP				
		MPP	MSP	BLA	CHA	MPSP
BCW	0.25	47.7%	44.9%	12.5%	97.6%	94.5%
BLD	2.13	5.9%	1.3%	1.5%	9.3%	7.9%
DIA	4.62	3.8%	5.2%	1.0%	10.6%	9.7%
HEA	0.42	3.6%	6.6%	5.4%	19.5%	16.3%
VOT	0.21	22.9%	29.6%	9.6%	62.6%	50.1%
Average		16.8%	17.5%	6.0%	39.9%	35.7%

Table 3: Running time of heuristic and exact maximum pattern generation algorithms.

It can also be seen that in view of the very high coverages of the patterns produced by CHA, the computational expense of solving the exact model is not justified. Note, however, that the running time of CHA usually exceeds slightly the sum of the running times of the three individual heuristics.

8 Application to Classification

One of the most important applications of maximum patterns (or their heuristically generated approximations) is in the Logical Analysis of Data (LAD). Patterns are the building blocks of LAD models, and the accuracy of these models depends on the type of patterns used for their construction. It is therefore important to empirically evaluate the effect of using (exact or approximate) maximum patterns on the accuracy of LAD classification models. We present below the results of such evaluation carried out on the five datasets described in the previous section.

Given a dataset $\Omega = \Omega^+ \cup \Omega^-$, a collection of positive and negative patterns $\Pi^+ \cup \Pi^-$ is called a *LAD model*, if every $\omega^+ \in \Omega^+$ is covered by at least one pattern in Π^+, and every $\omega^- \in \Omega^-$ is covered by at least one pattern in Π^-. Given a LAD model and a "new" observation $\omega \notin \Omega$, the "classification" of ω is determined by the sign of the so-called *discriminant* Δ. If $\pi^+(\omega)$ and $\pi^-(\omega)$ represent respectively the number of those positive and negative patterns in Π^+ and Π^- that cover ω, then the value of the discriminant Δ on ω is

$$\Delta(\omega) = \frac{\pi^+(\omega)}{|\Pi^+|} - \frac{\pi^-(\omega)}{|\Pi^-|}.$$

LAD classifies ω as positive (respectively, negative) if $\Delta(\omega) > 0$ (respectively, $\Delta(\omega) < 0$). If $\Delta(\omega) = 0$, LAD declares ω "unclassified"; it should be remarked that this situation occurs very infrequently.

We evaluate the accuracy of LAD models using the so-called *k-fold cross-validation* method. This is a re-sampling technique which randomly partitions the dataset into k parts of approximately equal sizes, preserving the ratio of positive and negative observations. Then, one of the parts is put aside to be used as a testing set, while the other $k-1$

parts are used as the training set to infer a LAD model, whose accuracy is then evaluated on the testing set. This evaluation is repeated k times using always another one of the k parts as the testing set, and finally the average accuracy over the k experiments is calculated. This k-folding evaluation is repeated r times, each time randomly generating a different k-partition, and the accuracy is averaged over these r experiments.

The accuracy of a LAD model on the testing set is calculated using the formula:

$$\text{Accuracy } = \frac{1}{2}\left[a + e + \frac{1}{2}(c + f)\right], \tag{9}$$

where a, b, c (d, e, f) represent the percentages of positive (negative) observations which are predicted by the LAD model to be positive, negative, or unclassified, respectively (see Table 4).

	Predictions		
	Positive	Negative	?
% of Positive Observations	a	b	c
% of Negative Observations	d	e	f

Table 4: Classification matrix.

Note that $a + b + c = d + e + f = 100\%$. The reason for assigning a 50% accuracy to unclassified observations is due to the possibility of easily achieving this accuracy by assigning randomly and with equal probability either of the two classes to every unclassified observation.

Each of the LAD models used in our experiments is built on a collection of maximum patterns, constructed by one of the exact or heuristic algorithms described in the previous sections. The only exception to this statement concerns the CAP model (*Combined Approximate Patterns*), which uses the union of the two pattern collections constructed by the heuristic algorithms MPP and MSP which separately enlarge each minterm of Ω to a maximized prime, respectively strong pattern. In Table 5 the combination CHA utilizes all three heuristics BLA, MPP and MSP.

Tables 5 and 6 below report the accuracy of various classification algorithms obtained by averaging the results of 10-fold cross-validation experiments over 20 runs.

Dataset	LAD	EMP	BLA	MPP	MSP	CHA	CAP
BCW	97.2%	96.6%	95.9%	95.4%	95.7%	96.8%	96.5%
BLD	72.3%	68.9%	71.6%	70.9%	69.0%	70.8%	72.3%
DIA	72.3%	71.9%	74.9%	73.7%	74.6%	74.7%	74.4%
HEA	83.8%	82.6%	81.9%	81.2%	81.9%	82.3%	83.7%
VOT	96.6%	96.6%	96.8%	95.8%	96.5%	96.5%	96.2%

Table 5: Classification accuracy of LAD using different types of patterns.

Note that none of the algorithms for constructing LAD models shown in Table 5 clearly dominates the others in terms of accuracy. The results in Table 5 indicate that the computational expense of producing patterns achieving the exact maximum coverage does not translate into an (even marginally) superior classification performance of the resulting EMP model of LAD, since the LAD models using heuristically generated patterns are always of comparable quality (being even better on some datasets). Furthermore, in spite of the fact that CHA chooses for every point the best of the three patterns generated by BLA, MPP and MSP, the CHA model does not always perform better than its individual components. Overall it looks like the CAP model of LAD provides a reasonable compromise between achieving superior classification performance and keeping the computational expense reasonably low. The accuracies of the CAP models are comparable to the highest accuracies of the models constructed by the other algorithms, as well as to the accuracies reported in [5] of the original LAD implementation.[2]

Based on the above, we shall use the CAP model of LAD (termed CAP-LAD in the comparisons presented below) as LAD's reference implementation. A significant advantage of this reference implementation of LAD is its non-reliance on various control parameters (utilized in the original implementation of LAD), the only one still used being the fuzziness parameter φ. This drastically reduces the amount of fine tuning and therefore simplifies the usage of the LAD methodology.

In Table 6 we present the cross-validated accuracies of CAP-LAD and of four commonly used machine learning algorithms as implemented in the publicly available Weka software package [11]: C4.5 Decision Trees (C4.5), Nearest Neighbors (Near.Nghbr.), Simple Linear Logistic Regression (S.Log.) and Support Vector Machines (SVM). The comparison of these accuracies shows that CAP-LAD is a very competitive classification method, whose accuracy mostly exceeds the accuracies of all the other examined methods.

Dataset	Weka Methods						CAP-LAD
	C4.5	Near.Nghbr.	S.Log.	SVM	Average	Maximum	
BCW	94.9%	96.4%	95.6%	96.6%	95.9%	96.6%	96.5%
BLD	63.8%	62.5%	66.3%	69.2%	65.5%	69.2%	72.3%
DIA	70.6%	64.8%	71.6%	71.6%	69.6%	71.6%	74.4%
HEA	79.1%	77.7%	82.6%	83.2%	80.6%	83.2%	83.7%
VOT	97.3%	94.0%	97.0%	97.1%	96.4%	97.3%	96.2%

Table 6: Classification accuracy of Weka and CAP-LAD.

[2]Since BLD is absent in [5], for the sake of completeness its accuracy has been evaluated in the context of this study.

9 Conclusions

In this paper we investigated the problem of constructing maximum coverage patterns containing any given observation point in the dataset. Both exact and heuristic methods for the construction of such maximum patterns are presented in the paper, and it is shown that the heuristically constructed ones can achieve more than 98% of the maximum possible coverage, while requiring only a fraction of the computing time of the exact method.

Maximum patterns are shown to be useful for constructing highly accurate LAD classification models. One of the advantages of such models is that in contrast with the original LAD models, the maximum pattern based ones do not require extensive calibration (determination of good values of several control parameters), and can therefore be easily used by non-experts.

It is also interesting that the LAD models built using the exact patterns of maximum coverage do not exhibit any superior classification performance as compared to the models built using approximate maximum patterns – in agreement with similar phenomena observed in different machine learning contexts [7].

In comparisons with the commonly used machine learning algorithms implemented in the publicly available Weka software package, the proposed reference implementation of LAD (termed CAP-LAD) is shown to be a highly competitive classification algorithm.

References

[1] Alexe G., P.L. Hammer. Spanned Patterns in the Logical Analysis of Data. *Discrete Applied Mathematics (to appear).*

[2] Alexe G., S. Alexe, P. L. Hammer. Pattern-Based Clustering and Attribute Analysis. *RUTCOR Research Report 10-2003 (to appear in Soft Computing)*, 2003.

[3] Alexe S., P.L. Hammer. Accelerated Algorithm for Pattern Detection in Logical Analysis of Data. *Discrete Applied Mathematics (to appear).*

[4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

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

[6] Dash Optimization, Inc. Xpress-MP Release 2003C: Xpress-Mosel and Xpress-Optimizer, 2003.

[7] Dietterich T. Overfitting and under-computing in machine learning. *ACM Computing Surveys*, 27:326–327, 1995.

[8] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[9] P. Hammer, A. Kogan, B. Simeone, and S. Szedmak. Pareto-Optimal Patterns in Logical Analysis of Data, 2001.

[10] P. L. Hammer, R. Holzman. Approximations of Pseudo-Boolean Functions; Applications to Game Theory. *ZOR – Methods and Models of Operations Research*, 39:3–21, 1992.

[11] Witten I.H., E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco, 2005.