ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

G. Felici, B. Simeone, V. Spinelli

CLASSIFICATION TECHNIQUES AND ERROR CONTROL IN LOGIC MINING

R. 21 Settembre 2009

Giovanni Felici – Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email :felici@iasi.cnr.it.

Bruno Simeone – Università degli Studi di Roma "La Sapienza", Piazzale Aldo Moro 4, 00185 Roma. Email: mecoli@uniroma1.it.

Vincenzo Spinelli – ISTAT - Istituto Nazionale di Statistica, Via Tuscolana, 1788, 00173, Rome, Italy (*vispinel@istat.it*)..

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica, CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: `iasi@iasi.rm.cnr.it`
URL: `http://www.iasi.rm.cnr.it`

Abstract

In this paper we consider Box Clustering, a method for supervised classification that partitions the feature space with simple convex sets (boxes). *Box Clustering* produces systems of logic rules obtained from data in numerical form. Such rules explicitely represent the logic relations hidden in the data w.r.t. a target class. The algorithm adopted to solve the *Box Clustering* problem is based on a simple and fast agglomerative method that can be affected by the initial choice of the starting point and by the rules adopted for by the method. In this paper we propose and motivate a randomized approach that generates a large number of candidate models using different data samples, and then chooses the best candidate model according to 2 criteria: *model size*, as expressed by the number of boxes of the model, and *model precision*, as expressed by the error on the test split. We adopt a Pareto-optimal strategy for the choice of the solution, under the hypothesis that such a choice would identify simple models with good predictive power. This procedure has been applied to a wide range of well known data sets to evaluate to what extent our results confirm this hypothesis; its performances are then compared with those of competing methods.

1. Introduction

One of the main problems in the identification of a good interpretative and predictive model is the trade-off between the precision that a model exhibits on training data and its ability to correctly predict data that are not in the training set. Such aspect is particularly important for those models that are loosely constrained in their size and can thus be adapted to the training data in a myopic way (e.g., Neural Networks, Support Vector Machines, decision trees, etc.). For this reason Data Mining techniques should be equipped with methods and tools to deal with the problem of *overfitting*, or *overtraining*, particularly serious when (a) the data is affected with noise, (b) it is not possible to define clearly an objective function for the model, or (c) the problem of finding an optimal solution for such a function is computationally intractable.

The problem of overfitting control can be considered within a formal framework for those methods that are based on a treatable mathematical formulation of the learning problem: for example, in *Support Vector Machines* (SVM) and *Neural Networks* a regularization parameter that weights some measure of model complexity is often used to balance the objective function and contrast any potential overfitting behavior (see [1, 2, 3]).

Such methods, being based on mathematical optimization algorithms, appear t be well suited to those learning problems where the main objective is the correct classification of training and verification data, but where the extraction of usable knowledge from data plays a secondary role w.r.t. to predictive power.

Nevertheless, in certain settings it is important to find good predictive models that are also able to be understood and verified by users, and, last but not least, integrated with other knowledge from different data sources and experience. The most popular method in this class are decision trees (see [4]), that build a classification model based on the iterative partition of the training data. The leaves of this tree represent logic rules on the input variables and can be viewed as clauses of a *Disjunctive Normal Form* (DNF) formula in propositional logic (see [22] for further details on DNF).

Other interesting methods in this class are those that are based on the formulation of combinatorial and integer optimization problems, whose solution can again be put in relation with DNF formulas that classify the training data, such as the *LAD* methods (Logical Analysis of Data) (see [11, 15]), the *Lsquare* method (see [5]), and the *OCAT* method (see [8, 9]). Such methods assume that the data is described straightforwardly by logic variables, and thus may require proper transformations of numerical data into logic data, often referred to as *discretization* or *binarization* (see [10, 13]). This means that the logic based method is applied to the transformed data. What eventually characterizes the logic mining is the nature of the classification model (expressed as a logic formula in the input variables) rather than the nature of the input data, that can, in principle, always be transformed into logic form.

Logic Mining methods are typically based on models and algorithms where it is difficult to formalize the trade-off between precision and generalization capabilities, i.e., to control overfitting. The introduction of regularization effect in the objective function is not straightforward and moreover may generate significant computational problems. In decision trees, for example, overfitting is controlled by pruning techniques that are applied to the tree according to a user-specified confidence parameter; in *LAD*, several methods to control the format of the resulting formula are combined with the use of an objective function that minimizes the dimension of the solution formula; in *Lsquare*, a greedy approach that minimized the cost of a conjunctive clause is one of the techniques by which overfitting control is achieved.

The method under analysis in this paper - *Box Clustering* (*BC*) - can be considered a logic mining method in the sense that the system of boxes that form the classification model can be mapped into a DNF formula, but operates in a slightly different way, as the identification of threshold values for numerical variables (the above mentioned *binarization*) is performed within the classification algorithm itself, and thus extends the logic approach without resorting to any transformation of the original data (see [25]).

BC presents two interesting properties for the control of the error and of the overtraining behavior: a) its complexity can be easily put in relation with the number of boxes that compose the final solution; b) alternative solutions of different complexity can be identified efficiently according to different random choices of the parameters by which the solution algorithm is initialized. In addition, *BC* can deal directly with classification problem with more that two classes (see example 3 in section 3), and manages the missing values inside the model itself (see data sets *ictus*, *annealing*, and *page* in table 1 in subsection 6.1).

It is therefore natural to try and combine these two features to derive a method to choose among those solutions obtained by a randomized application of the BC algorithm, with the specific aim of controlling prediction error and find the right balance between precision and overfitting on the training data. In this sense the proposed approach evaluates the solution space w.r.t. to the specific problem under analysis, and is able to capture the complexity of the latent model and the amount of noise present in the data. The best candidate model is selected according to a bi-criterion problem: *model size*, expressed by the number of boxes of the model, and *prediction accuracy*, expressed by the error on the test split. We propose a choice criterion for the model based on Pareto-optimality under the hypothesis that such a choice would identify simple models with high accuracy.

The paper is organized as follows.

Section 2 defines the main terminology used in this paper. In section 3, we show how the *BC* approach can be used in supervised classification problems. In section 4, we define a bi-criterion procedure to select the *best BC* solution for our classification problem, based on the error distribution on the test set. Here we formulate an hypothesis on the link between the classification performance on the test set (used to select among different classifiers) and the validation one (used to assess the final performances of the classifier chosen). From these considerations, we define, in section 5, an iterative procedure to choose the best *BC* solution for our classification problem. Finally, in section 6, we apply this procedure to obtain *BC* models for benchmark data sets used in machine learning, and check the validity of our hypothesis.

2. Brief Introduction to Box Clustering

In Logic Mining it is assumed that a number of observations are given in the form of n-dimensional vectors; with each of these vectors a binary outcome is associated; according to its values the observations are usually termed "*positive*" (or "*true*"), and "*negative*" (or "*false*") (see [15, 16, 17]). Here we assume that the components are not constrained in type (e.g., they can be binary or numeric[1]) and are represented in $\mathbb{R}^n$.

We define a *box* as a multi-dimensional range delimited by two points in $\mathbb{R}^n$, (L, U):

$$I(L,U) = \prod_{i=1}^{n} [l_i, u_i] = \{X \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i \; i = 1, \ldots, n\}$$

[1]We can further extend this approach to N-value variables and manage the presence of missing values by including them inside the *BC* model itself.

where L and U are defined the *lower* and *upper* bound of I. A box is called positive (or negative) if it includes only positive (respectively, negative) observations. Positive and negative boxes will also be called homogeneous boxes. For any finite set S of points, we define its *box-closure* $[S]$ as the intersection of all boxes containing S. Given two boxes $A = [S]$ and $B = [T]$, we define their union $A \vee B$ to be the box $[S \cup T]$. The BC model can be easily extended from 2-Class to N-Class supervised classification problem, and, from now on, we consider the general case: $N \geq 2$.

A decision tree can be seen as a set of logical formulas, defined by its leaves, and each logical formula can be considered a special box as we can see example 2.

Let us considered a decision tree defined on 3 numerical variables: $\overline{x} = (x_1, x_2, x_3)$. If a leaf of the tree is defined by the logical formula

$$F(\overline{x}) \equiv (x_1 < 1.2) \wedge (x_2 > 0.0) \wedge (x_3 \geq 0) \wedge (x_3 \leq 1)$$

then we can considered this equivalent box

$$B =]-\infty, 1.2] \times]3, +\infty[\times [0, 1].$$

It is easy to check that: $F(\overline{x})$ is true $\Leftrightarrow \overline{x} \in B$.
This example also shows that a decision tree can be always seen as a set of logical formulas in DNF format.
When considering a set of observations, the BC problem amounts to finding a finite set of boxes as a solution of a specific optimization problem (see [25]). Such problem may be based on different types of constraints, generally based on the geometrical properties of the boxes:

- *coverage*: every observation is included in at least a box, and every box includes at least one observation.
- *homogeneity*: all the boxes are homogeneous, i.e. all the points inside a single box are from the same class. This implies that we must have at least N boxes, where N is the number of classes, if *coverage* and *homogeneity* do hold.
- *spanning*: every box is the *box-closure* for the set of observations inside the same box.
- *overlapping*: four types of overlapping conditions may be considered. If *homogeneity* does not hold, then we have: (a) *none*, i.e. no overlapping configuration is allowed, and (b) *strong* otherwise. If *homogeneity* does hold, then we can define: (c) *mild*, i.e. only homogeneous boxes can be overlapped, and (d) *weak* otherwise.
- *consistency*: every box must contain at least an observation not included in another box, and this implies that we must have at most n boxes, where $n \leq |S|$.
- *saturation*: if we consider a subset C of the above mentioned constraints, we say that a set of boxes $\hat{B}$ is saturated if and only if it satisfies C and we cannot join any pair of boxes in $\hat{B}$ without violating any constraints in C.

It is worth to notice the link between spanning sets and set coverage of the parameter space. In literature we can find *"the maximum patterns"* approach to get a set coverage of the parameter space by the minimum number of patterns (see [12]). In our approach, we find a saturated system of boxes that is a spanning and a coverage for the training set, and this means that

every training point is inside a box (*coverage*) and all the boxes have the minimum dimension to include the training points inside them (*spanning*). But the two constraints do not necessarily imply that the system of boxes covers all the points of the feature space.
If the constraints *coverage* and *homogeneity* hold then the BC approach is an exact method as it performs an exact separation of training data (see proposition 3.1 in section 3). This fact motivates the analysis of methods to select, among different BC models, those that appear less biased by overfitting.
The identification of a set of boxes that satisfies a set of constraints is not an easy problem to formulate and solve; a restricted version known as *"the maximum box problem"*, where one wants to find the homogeneous box including the highest number of positive (or negative) points, has been studied in [20]. For a fixed number of points, the (weighted) maximum box problem is in class $\mathcal{P}$, but it can be shown that the maximum box problem is in class $\mathcal{NPC}$ for any number of points, by using a polynomial reduction of the maximum clique problem on the graph $G = (V, E)$, known to be $\mathcal{NP}$-complete, to the maximum box problem. From the practical standpoint we use an agglomerative approach based on greedy or random choice, and this is a well-known algorithmic framework in BC, for the search of a saturated box system satisfying an input set of constraints (see [23, 24, 25]).
The simplest version of the algorithm is mainly based on two steps:

- *starting box set*: all the points in the training set are consider as boxes, and in this case we call them *singletons*. These boxes are consider the current box set.
- *main loop*: in each step of the loop, we search for two boxes that can be joined in the current box set. If these boxes do not exist then we exit the loop otherwise we make a new box by joining the two boxes, and update the box set.

The maximum number of iterations for the search loop is not greater than the number of points in the training set, and the last box set is a saturated box set.

3. *BC*-based Classifier

A BC-based classifier is a function (shortly referred to as *classification* in the following), with three input parameters: (a) a set of boxes BS, (b) the weight/distance function $w()$, and (c) the point p to classify. This function first assigns the weight $w_i = w(B_i, p)$ to each $B_i \in BS$, then chooses the index of the box having the best, i.e. the minimum, weight; generally, there is a subset $\overline{B} \subseteq BS$ such that each box $B \in \overline{B}$ has the best value. There can be two exclusive conditions:

- the boxes in $\overline{B}$ belong to *only one class*: the function randomly chooses the index of one of these boxes.
- the boxes in $\overline{B}$ belong to *more than one class*: the most elementary classification functions return an undefined result; the most sophisticated ones try to find out group of homogeneous boxes in $\overline{B}$ having *the best global weight*. If this search has no result then the function returns an undefined result.

The output of the function is the estimated class of the point p. The weight $w(B, p)$ is the measure of "*the attraction intensity*" of B with respect to p. If BS is a *set coverage* of the observation space, then we can naturally define $w()$ as the characteristic function shown (*yes-no*) in equation 1.

$$w(B,p) = \begin{cases} 0 & p \in B \\ 1 & otherwise \end{cases} \tag{1}$$

If BS is not a *set coverage* of the observation space, e.g. the *spanning* constraint holds, then we can define $w()$ as the *manhattan distance*, (see equation 2), naturally compatible with the box geometry defined in section 2.

$$w(B,p) = \begin{cases} 0 & p \in B \\ d_1(B,p) & otherwise \end{cases} \tag{2}$$

It is worth to notice that $d_1(B,p) = \min_{q \in B} d_1(q,p)$, and it can be proven that $d_1()$ depends only by the size and position of B and p in $\mathbb{R}^n$, as shown in figure 1: let us consider a generic box B and four points, P_1, P_2, P_3, and P_4 , placed in generic positions outside B. The dashed lines are the piecewise paths which lengths give us the values $d_1(B,P_i)$, $i = 1,2,3,4$.

Consider the usual split of the available data into a *the training set* and a *the test set*, and the complete data set $S = Tr \cup Ts$. We also assume that the target variables divides S into n classes.

With regard to Tr, then we define $w()$ in a slightly different way:

$$d_1(B, Tr, p) = mean_{q \in Tr \cap B} d_1(q,p).$$

This new version of d_1 can be defined as a gravitational model, for $w()$ depends on the mutual position between the point p and all the points of Tr which are inside B.

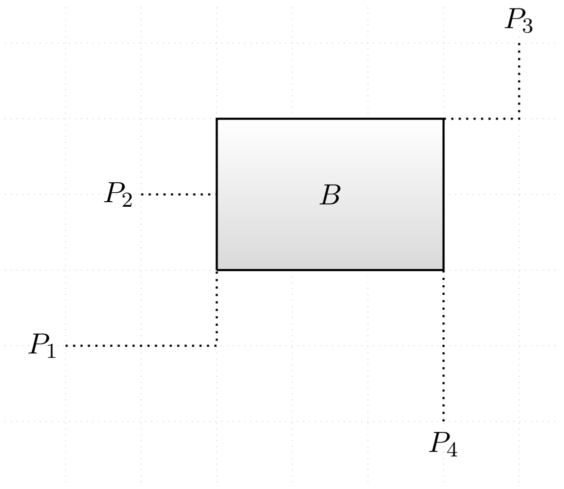

Figure 1: The nearby weight function

These three classes for the weight function have a similar behavior for every box: they are positive definite functions and get their minimum value inside each box.

We then apply the classification function to every point in Ts and define the confusion matrix $M_c = M_c(Ts) = \{m_{ij}\}$ where m_{ij} is defined as the number of points having the i-th class in

Ts and classified in the j-th class by the BC classifier [2]. M_c is univoquely defined by the pair $(classification(), Ts)$, and thus depends on $(BS, w(), Tr)$. It comes from the definition that $Ok(M_c) = \sum_{i=1,n} m_{ii}$ is the number of the correctly classified observations of Ts, and, for this reason, we can define the error function $Err(M_c) = \sum_{i \neq j} m_{ij}$.

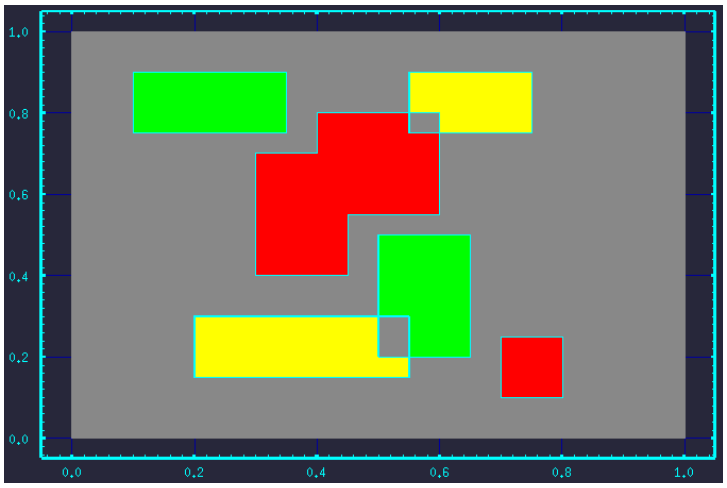

Figure 2: BC Classifier by characteristic function

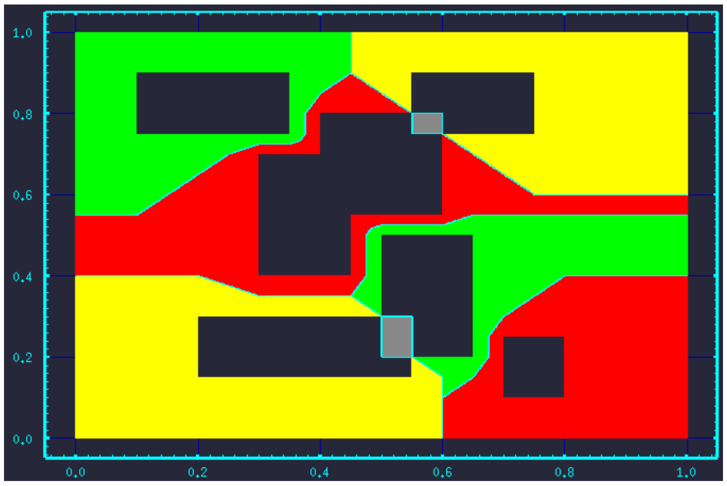

Figure 3: BC Classifier by Manhattan function

Let us consider a simple BS made by 7 boxes in $[0,1]^2$; in BS there are 2 *green class* boxes, 3 *red class* boxes, and 2 *yellow class* boxes. There are 3 pairs of overlapping boxes: *red-red*, *yellow-red*, and *yellow-green* pairs. If we consider the *gray* color for the indefinite cases, we can classify a uniform point grid in $[0,1]^2$ by using the different weight functions, i.e. equations 1 and 2. In figure 2, we can see the result we get for the function *yes-no*: every point outside

[2]From the definition of M_c we can always consider $i, j = 1, \dots, n+1$: when $i, j = n+1$ we can manage some extreme cases: $m_{n+1,j}$ is the number of the points having a class in Ts not defined in Tr, and $m_{i,n+1}$ is the number of the points that the BC classifier can not assign a class.

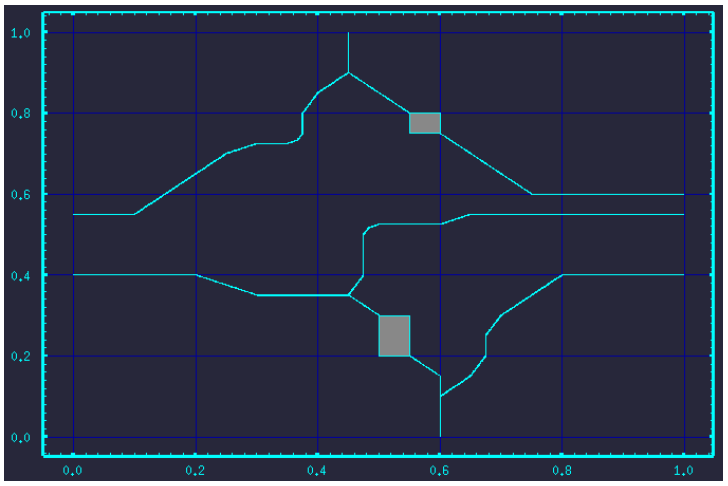

Figure 4: *BC* Class Edges for Manhattan Classifier

the boxes are gray, and so is for the overlapping regions. In figure 3 we have the result for the function *nearby*: the gray regions are very narrow outside the boxes, and they can be easier seen in figure 4. The edges are straight lines for the function *yes-no* in this 2-dimensional problem (hyperplanes in $\mathbb{R}^n$), but this is not true for the function *nearby*. Moreover, it is worth to consider the number of gray points we have in the two methods. In figures 2 and 4 we can see the different presence rate of the gray points in the sample grid. The function *yes-no* has many gray points, and this is coherent with the fact that BS is not a set coverage of $[0,1]^2$. The *nearby* function returns a very small number of gray points and this is an important parameter to evaluate the classification power of this BC classifier.
From the definitions of the constraints {coverage, homogeneous} and $Err(M_c)$, we can check that proposition 3.1 holds.

Proposition 3.1. *If the box set* BS *satisfies, at least, the constraints {coverage, homogeneous} on the point set* Tr*, then* $Err(M_c(\mathrm{Tr})) = 0$ *holds for every* BC *classification technique, i.e. yesno, nearby, and gravitational.*

Proof: Let us consider $p \in Tr$. If BS is a coverage then there exists $B \in BS$ such that $p \in B$; but BS is homogeneous, and this means that p and B have the same class. If there is another $B' \in BS$ such that $p \in B'$ then B and B' must have the same class. Since every BC classifier must rightly classify p, we can conclude that $Err(M_c(Tr)) = 0$. □

Proposition 3.2. *If* $S = \mathrm{Tr} \cup \mathrm{Ts}$ *and* BS *is coverage and homogeneous for* Tr*, then* $Err(M_c(S)) = Err(M_c(\mathrm{Ts}))$.

Proof: For every BC classifier, we can check that

$$Err(M_c(S)) = Err(M_c(Tr)) + Err(M_c(Ts)).$$

From the proposition 3.1, we know that $Err(M_c(Tr)) = 0$, and this ends the proof. □

The above considerations lead to conclude that every BC-classifier, based on BS with few elementary properties, is exposed to the risk of over-training, a well known situation where the learned model is adapted to noise or non-informative training data with excessive precision resulting in complex models and poor recognition performances on test data.

4. Best Choice in BC solution group

Let us consider a BC approach for the supervised classification problem on S, and $\hat{BS} = \{BS_i\}_{i=1,n}$ a set of feasible solutions. Solutions in $\hat{BS}$ are created by the agglomerative algorithm with (a) different starting parameter values, (b) cross-validation or (c) repeated percentage split on S.

We tackle the problem of choosing the *best* solution in $\hat{BS}$ considering two alternative objective functions:

$$\begin{cases} \min & |BS| \\ s.t. & BS \in \hat{BS} \end{cases} \tag{3}$$

$$\begin{cases} \min & Err(M_c) \\ s.t. & BS \in \hat{BS} \end{cases} \tag{4}$$

Problem 3 puts the focus on the simplest BC model we can consider for S; on the contrary, the problem 4 puts the focus on the reliability of the BC model in classification; the extent to which these two objectives are in a trade-off relation is strongly depending on the data under analysis.

While the evaluation of the complexity of the model is directly measured by objective function of problem 3, the use of simple test accuracy for the evaluation of predictive power (problem 4) is not a straight-forward choice. The literature proposes alternative and possibly more meaningful methods to evaluate the performance of a classifier, such as, among others, the area under the ROC (*Receiver Operating Characteristic*) curve, or simply AUC, that is widely used to measure model performance in binary classification problems (see [26]). Huang and Ling (see [27]) also show theoretically and empirically that AUC is a better measure for model evaluation than accuracy, i.e. problem 4. Nevertheless, the literature does not converge on a widely accepted performance method similar to ROC analysis for an N-class classifier ($N > 2$), and for this reason we make the choice to use to accuracy as a performance evaluator. BC is by definition a N-class classifier, and many of the experimental tests presented later are based on dataset with $N > 2$ classes.

Propositions 3.1 and 3.2, in section 3, suggest us not to solve just one problem to get a robust and reliable BC solution. For this reason we consider a bi-criterion problem:

$$f(BS, Ts) = \begin{pmatrix} |BS| \\ Err(M_c) \end{pmatrix} \tag{5}$$

$$\begin{cases} \min & f(BS, Ts) \\ s.t. & BS \in \hat{BS} \end{cases} \tag{6}$$

Optimization with regard to multiple objective functions aims at a simultaneous improvement of the objectives. The goals in problem 6, may easily conflict so that an optimal solution in the conventional sense does not exist. Instead we aim at e.g. Pareto optimality, i.e., we find the Pareto set in the plane π_{NE} as shown in figure 5, from which we can choose a promising BC solution.

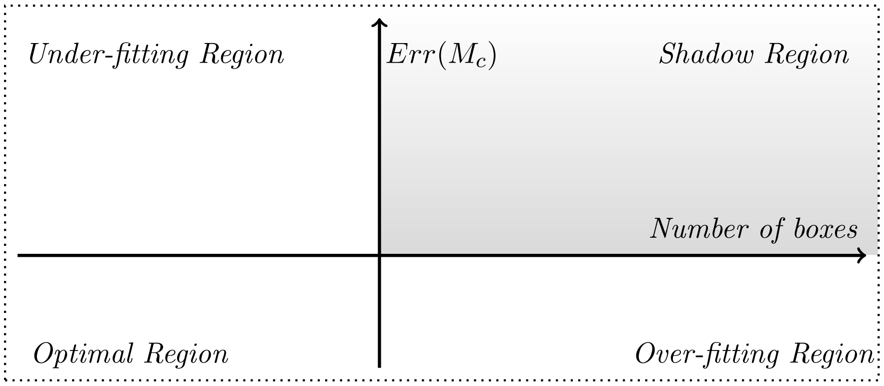

Figure 5: The BCG-like matrix in Logic Mining

In the plane π_{NE} we can place each BS_i according its number of boxes and number of errors. We can classify the four quadrants of this plane as following:

- *Shadow region*: the worst BSs are in the upper right quadrant. They have many boxes and high error rate.
- *Over-fitting Region*: the BSs in the lower right quadrant have many boxes but low error rate; *model complexity* dominates *model generalization*.
- *Under-fitting Region*: the BSs in the upper left quadrant have few boxes but high error rate; *model generalization* dominates *model complexity*.
- *Optimal Region* or *Opt*(π_{NE}): the BSs in the lower left quadrant have few boxes and low error rate; for this reason they are the natural candidates to be considered as the best in our BS group. In general, we want to choose our best solution among the non dominated solutions in these region.

We consider the point $(\bar{N}, \bar{E})$ the origin of the axes, where $\bar{N}$ and $\bar{E}$ are the mean value of the number of boxes and the mean value of the error rate over all samples, respectively. Let us consider P^* the lower left corner of π_{NE}, and the value $d^* = \min_{i=1,n} d(P_i, P^*)$. Furthermore, let us consider the set $\widehat{P} = \{P_i | d^* = d(P_i, P^*)\}$:

- $|\widehat{P}| = 1$: this means that there is only one $P_i \in \widehat{P}$; this point is the best BS we can choose $\hat{BS}$.
- $|\widehat{P}| > 1$: we have more than one BS at the minimum distance, and we choose the unique BS having the minimum error rate (*model complexity* is preferred to *model generalization*). In figure 6 we have an example of this method: $\{P_1, P_2, P_3, P_4\}$ is the Pareto set of $\hat{BS} = \{P_1, P_2, P_3, P_4, \ldots, P_i, P_j, \ldots\}$, and P_2, P_3 are the BSs with the minimum distance to P^*. We prefer P_2 to P_3, because it has a lower error rate. Finally, P_4 is the *solution* for the problem 3, while P_1 is the *solution* for the problem 4.

In figure 7, we have the full schema of this multi-criterion approach for $\hat{BS}$.
The solution BS_{i^*} is not generally the optimal solution of the problems 3 and 4. The point $\hat{P} = (N_{i^*}, E_{i^*})$ can not be in the *Shadow Region* by definition. If we consider the equivalent

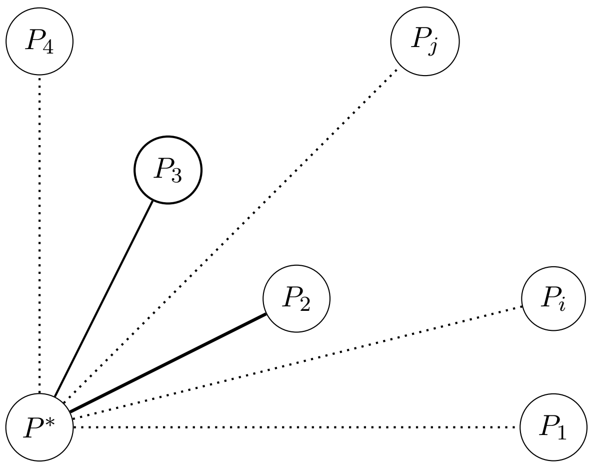

Figure 6: Choice in Pareto set

```
function pareto_choice($N, E, n$)
{
    $P^* = (\min_i N_i, \min_i E_i)$;
    $d^* = +\infty$;
    $for(i = 1; i \leq n; i++)\{$
        $P_i = (N_i, E_i)$;
        $d_i = d(P^*, P_i)$;
    $\}$
    $d^* = \min_i d_i$;
    $E^* = +\infty$;
    $i^* = +\infty$;
    $for(i = 1; i \leq n; i++)\{$
        $if(d_i = d^*)\{$
            $if(E_i < E^*)\{$
                $E^* = E_i$;
                $i^* = i$;
            $\}$
        $\}$
    $\}$
    $return(i^*)$;
}
```

Figure 7: Pareto Choice schema for BC

definitions for the 2-dimensional plane based on the validation set (i.e. $\pi_{NE'}$, $Opt(\pi_{NE'})$, and $\hat{P}'$), we can now informally formulate our hypothesis:

Hypothesis 1. *For n large $\hat{P} \in Opt(\pi_{NE}) \Rightarrow \hat{P}' \in Opt(\pi_{NE'})$.*

The hypothesis may also be proved under some probabilistic conditions on the distribution of the data, but we are more interested in its practical verification: probabilistic information is normally not available for real data and therefore its practical relevance is difficult to assess. On the other hand, it is of interest to check if the hypothesis is verified for reasonable values of n in real data sets with different sizes and nature.

5. Bi-Criterion Procedure for BC-based Classifier

The considerations of Section 4 can be used to design an algorithm for BC to control the validation set error. This algorithm is based on five basic functions: (a) *data-split*, (b) *local-search*, (c) *pareto-choice*, (d) *normalization*, and (e) BC-*Classifier*.
The *data-split* function returns a percentage split of the input data set S. It uses a percentage of the total records as training data and the remaining percentage as test data. If 80% is shown as the split, then 80% of the data set is being used for training, while the remaining 20% is used for testing. It must be recalled that a portion of the data should be stored apart before the splitting into training and testing to act as validation set. The *local-search* function is the implementation of the well-known agglomerative algorithm for BC (see [20, 24, 25]), as described at the end of Section 2, based on a random choice engine. The *pareto-choice* function has been fully explained in paragraph 4. The input vectors, i.e. N and E, widely vary in size as a result of the units selected for representation: $N_i \leq |S|$ but $E_i \leq 1$. To avoid a higher influence of N respect to E, when we consider the distance between two points, we need to normalize these vectors.
The BC-*Classifier* function is the general form of the function explained in paragraph 3; in this version, it returns the number of errors for the test set. In figure 8 we have the full description of the algorithm.

```
function error-control($S, P$)
{
  $for(i = 1; i \leq k; i++)\{$
    $(Tr, Ts)$ = data-split($S, P$);
    $BS_i$ = local-search($Tr$, P);
    $N_i = |BS_i|$;
    $E_i$ = $BC$-Classifier($BS_i, Tr, Ts, P$);
  $\}$
  $E'$ = normalization($E, k$);
  $N'$ = normalization($N, k$);
  $i^*$ = pareto-choice($N', E', k$);
  $return(BS_{i^*})$;
}
```

Figure 8: General Schema for BC

The algorithm is based on a repeated percentage split strategy and a multi-criterion choice of one *BS* out of the k available. We summarize in the input parameter P the structure of specific parameters that we have used in the experiments. In particular, we have used four parameters for each run of the procedure: (a) $k = 100$, (b) 80% percentage split, (c) *BC* random engine, and (d) overlapping *none*. The other several low-level parameters needed in the different steps and not relevant within the scope of the experiments have been here omitted for brevity.

6. Examples

In this section we present some experimental results obtained with the proposed method. After a brief description of the data sets considered, we present the *BC* results and, in subsection 6.3, a synthetic comparison with *decision trees* applied to the same problems.

6.1. The Data Sets

In order to empirically evaluate the efficiency of the procedure described in section 5 and its effectiveness in data analysis, we have applied it to seven frequently-used data sets, taken from the repository of the University of California, Irvine (UCI) (see [28]): Thyroid Domain (*thyroid*), Isolated Letter Speech Recognition (*isolet*), Blocks Classification (*page*), Multi-Spectral Scanner Image (*satimage*), Annealing Data (*annealing*), and Image Segmentation Data (*segment*). One further data set, *ictus*, is an unpublished medical archive about the ictus disease.

Table 1: Summary for considered data sets.

Data set	Training	Validation	%	Features	Classes	Missing
ictus	698	0		37	2	*y*
	500	198	28.4%			
annealing	798	0		18	6	*y*
	690	100	12.6%			
page	5406	0		10	5	*n*
	4000	1308	24.6%			
segment	210	2100	90.9%	19	7	*n*
thyroid	3772	3428	47.6%	21	3	*n*
satimage	4435	2000	31.1%	36	6	*n*
isolet	6238	1559	20.0%	617	26	*n*

These data sets have different values for size, number of features, and number of classes to classify. The main parameters of these six data sets are listed in table 1: columns 2-3 of the table report the number of records in training and validation sets that we have considered in each data set. For the data sets *ictus*, *annealing*, and *page* we do not have an explicit validation set, i.e. 0 in column 3, and we have randomly split the training set in two parts as shown in the second row of the table for these data sets. Column 4 contains the percentage rate of the validation set. Similarly, column 5-6 of the table correspond to the number of features and classes in each data set. The last column indicates the presence or the absence of missing values.

6.2. Experiment results with *BC*

For every dataset, we now consider the results in the plane π_{NE}, as defined in paragraph 4, and display them in figures 9-12. The point P^* is represented as black circle in each figure.

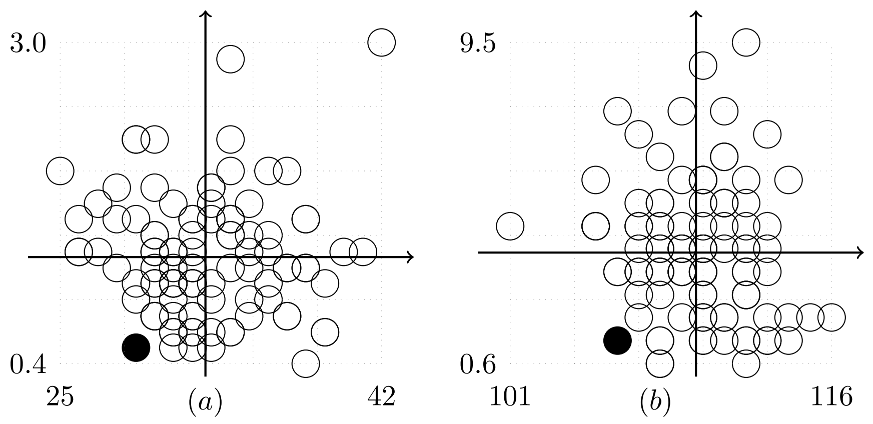

Figure 9: Results for (a) *thyroid* and (b) *annealing*

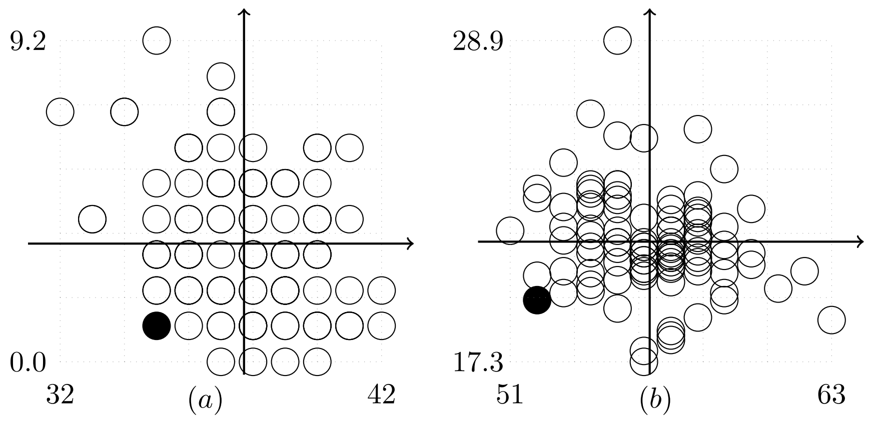

Figure 10: Results for (a) *ictus*, and (b) *isolet*

The table 2 give us a summary of the minimal, maximal, and average values for the parameters N , E, and E'. The columns 2-4 are related to N, i.e. the number of boxes, while the columns 5-7 are related to E, i.e. the test errors. The columns 8-10 provide the same information on the validation errors. The distribution for N and E are evaluated inside the function *error-control* for $k = 100$, where we build 100 training and test sets (repeated percentage split). Even if there is one validation set for each data set and generally we do not know the distribution of E', we have modified the behavior of the function *error-control* to evaluate it, but only for a better comprehension of our approach. This means that we have also applied all the box sets to the validation set.

The table 2 must be compared to the table 3, where we show the performances of P^* when applied to the validation set and the difference with the test error for the same point. The

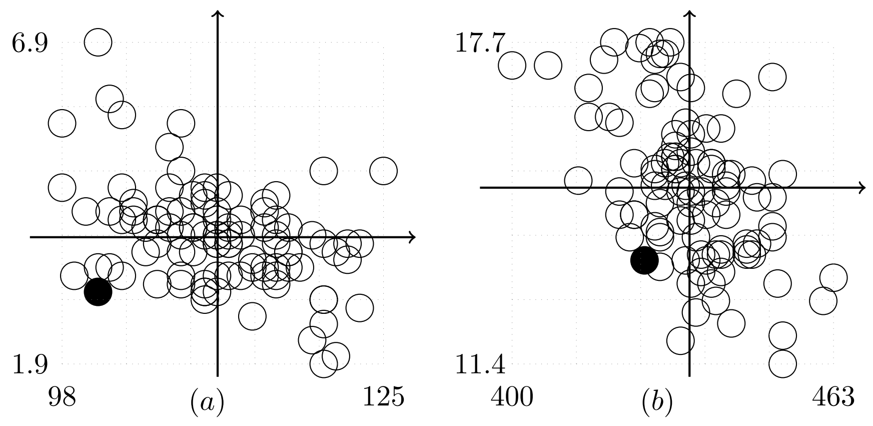

Figure 11: Results for (a) *page* and (b) *satimage*

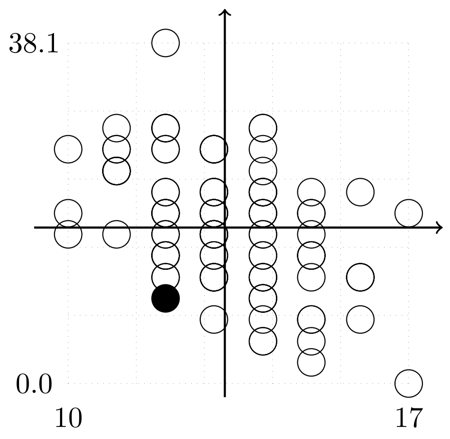

Figure 12: Results for *segment*

columns 2 and 3 give us the coordinates of the points P^* in π_{NE}, while the columns 2 and 4 those of the points P'^* in $\pi_{NE'}$. Column 5 shows an interesting measure for the performance of our choice algorithm: it gives us the percentile of the validation error rate of P'^* respect all the n points we could consider in our tests[3]. We can check that for two data sets, i.e. *thyroid* and *annealing*, P'^* is really a very good choice. For other two data sets, i.e. *page* and *satimage*, it has better performances than 70.0% of all points, while, for the other three data sets, i.e. *ictus*, *isolet* and *segment*, it is better than 60.0% of all points. Finally, we must underline that even if the hypothesis 1 in section 4 is always satisfied in our tests (column 6 in table 3), there is one case, i.e. *ictus*, where P'^* is on the border of $Opt(\pi_{NE'})$.

Table 2: Results in π_{NE} and $\pi_{NE'}$.

	Efficacy (#)			Efficiency (%)					
	Number of Boxes			Err(Test Set)			Err(Validation Set)		
Data set	Min	Max	μ	Min	Max	μ	Min	Max	μ
thyroid	25	42	32.7	0.40	3.05	1.28	1.25	2.39	1.84
annealing	101	116	109.7	0.63	9.49	3.70	1.00	4.00	2.12
ictus	32	42	37.7	0.00	9.18	3.39	2.40	5.29	3.85
isolet	51	63	56.2	17.31	28.93	21.65	19.44	29.06	22.52
page	98	125	111.1	1.88	6.88	3.85	2.75	4.71	3.64
satimage	400	463	434.8	11.39	17.40	14.85	14.20	16.70	15.38
segment	10	17	13.2	0.00	38.10	17.48	9.95	15.71	12.56

Table 3: Best Choices in π_{NE} and $\pi_{NE'}$.

Data set	N^*	E^*	$Err(V)$	$Percentile(Err(V))$	$\hat{P}' \in Opt(\pi_{NE'})$
thyroid	29	0.53	1.34	P_3	y
annealing	106	1.27	1.00	P_1	y
ictus	35	1.02	3.85	P_{35}	y
isolet	52	19.55	22.13	P_{38}	y
page	101	3.00	3.33	P_{16}	y
satimage	426	13.42	15.05	P_{27}	y
segment	12	9.52	12.10	P_{35}	y

6.3. Comparison with Decision Trees

A decision tree can always seen as a set of boxes (as seen in example 2 of section 2). For this reason, we have decided to use the performances of standard decision trees (a definitely well established method in the literature) as the benchmark for the *BC* results we present in subsection 6.2. We use the Weka suite (see [29]) to analyze the data sets by the decision trees, and the results are in table 4. These results have been obtained by using the method *weka.classifiers.trees.J48*,

[3]In this context, the known distribution of E' allows us to evaluate some statistical parameters we use in these tables, i.e. mean, min, max, and percentile.

based on *C4.5 approach* (see [30]). We have used the default configuration in Weka, but have controlled the confidence parameter for building the classification models. In table 4, there are the best results for 10-*cross validation* (columns 3-4), 20-*cross validation* (columns 5-6), and *percentage split (*80*%)* (columns 7-8). The results for the data set *isolet* are not available because its dimension do not allow a complete set of trials by the Weka system.

Table 4: Decision Tree Results.

		$k = 10$		$k = 20$		Percentage	
Data Set	Confidence	Leaves	Error(%)	Leaves	Error(%)	Leaves	Error(%)
thyroid	0, 20	16	0, 4	16	0, 3	16	0, 6
annealing	0, 20	49	6, 3	49	7, 4	49	7, 3
ictus	0, 30	12	12, 0	12	11, 0	12	11, 2
isolet	−−	−−	−−	−−	−−	−−	−−
page	0, 10	29	2, 7	29	2, 7	29	2, 9
satimage	0, 05	121	12, 9	121	12, 9	121	13, 2
segment	0, 20	39	3, 1	39	3, 1	39	2, 8

The comparison of tables 3 and 4 helps us in assessing that the proposed *BC* approach provides good results with respect to decision trees - its more natural competitor. Its performance are better for the majority of the experiments, while the differences in solution sizes are not of significant size. In particular, we can highlight that:

- The *BC* models are slightly more complex than the decision trees when we consider *annealing* and *ictus*, but they are more efficient, i.e. the difference of errors for the two approaches is greater than 5.0%.
- The *BC* models are more complex than the decision trees when we consider *thyroid* and *page*, but the efficiency is very similar, i.e. the difference of errors for the two approaches is less than 1.0%.
- The decision trees are better than *BC* models when applied to *satimage*.
- The *BC* models are simpler than the decision trees when we consider *segment*, but less efficient.

7. Conclusion

In this paper we consider *Box Clustering*, a method designed to classify data described by variables in qualitative and numeric forms by a set of *boxes* that are equivalent to logic formulas on qualitative or discretized numerical variables. We adopt a standard agglomerative approach based on random choice to solve the *BC* problem, and propose a method for the choice of the most interesting solution among those obtained by a sufficiently large number of different runs. Such method is based on the property of pareto-optimality in the plane defined by the model complexity and the model training error (π_{NE}). Despite the evidence that the error obtained on test data is a good predictor for the error that the model will obtain on new data, we also claim that a particular non-dominated solution in π_{NE} will maintain its desirable solution also when the validation error is considered. Such method of choice takes into account the complexity of the model and can play an important role in containing the overtraining behavior of the model. We try to verify our hypothesis by several heterogeneous data sets from the literature, and verify how it is experimentally confirmed for all of them. Moreover, the comparison of BC with a widely used *decision tree* technique w.r.t. to model size and test set accuracy provides positive feedback for the validation of the proposed method.

References

[1] M.A. Davenport, R. G. Baraniuk, and C. D. Scott. *Learning minimum volume sets with support vector machines.* IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Maynooth, Ireland, 2006.

[2] M.A. Davenport, R. G. Baraniuk, and C. D. Scott. *Controlling false alarms with support vector machines.* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, 2006.

[3] Neyman-Pearson SVMs: *www.ece.rice.edu/m̃d/np_svm.php/.*

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International, Belmont, Ca, 1984.

[5] K. Truemper. *Lsquare System for Learning Logic.* University of Texas at Dallas, Computer Science Program, April 1999.

[6] G. Felici and K. Truemper. *A Minsat Approach for Learning in Logic Domains.* INFORMS Journal on Computing, 13 (3), 2001, 1-17.

[7] G. Felici, F-S. Sun, and K. Truemper. *Learning Logic Formulas and Related Error Distributions*, in Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, G. Felici and E. Trintaphyllou eds., Springer Science.

[8] E. Triantaphyllou. *The OCAT approach for data mining and knowledge discovery.* Working Paper, IMSE Department, Louisiana State University, Baton Rouge, LA 70803-6409, USA, 2001.

[9] E. Triantaphyllou. *The One Clause At a Time (OCAT) Approach to Data Mining and Knowledge Discovery*, in Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, G. Felici and E. Trintaphyllou eds., Springer, Heidelberg, Germany, Chapter 2, pp. 45-87, 2005.

[10] S. Bartnikowsi, M. Granberry, and J. Mugan. *Transformation of rational data and set data to logic data*, in Data Mining & Knowledge Discovery Based on Rule Induction Techniques. Massive Computing, Springer Science, 12 (5) : 253–278, November 2006.

[11] G. Alexe, P.L. Hammer, P.L. Kogan. *Comprehensive vs. comprehensible classifiers in Logical Analysis of Data.* RUTCOR Research Report, RRR 9-2002. Also available at *http://rutcor.rutgers.edu/pub/rrr/reports2002/40_2002.pdf.*

[12] T.O. Bonates, P.L. Hammer, P.L. Kogan. *Maximum Patterns in Datasets.* RUTCOR Research Report, RRR 9-2006. Also available at *http://rutcor.rutgers.edu/pub/rrr/reports2006/9_2006.pdf.*

[13] P.L. Hammer, I.I. Lozina. *Boolean Separators and Approximate Boolean Classifiers.* RUTCOR Research Report, RRR 14-2006. Also available at *http://rutcor.rutgers.edu/pub/rrr/reports2006/14_2006.pdf.*

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

[15] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. *An implementation of logical analysis of data.* IEEE Transactions on Knowledge and Data Engineering, 12 (2): 292–306, November 2000.

[16] E. Boros, T. Ibaraki, L. Shi, M. Yagiura. *Generating all good patterns in polynomial expected time.* lecture at the 6th International Symposium on Artificial Intelligence and Mathematics, Ft. Lauderdale, Florida, January 2000.

[17] P.L. Hammer, A. Kogan, B. Simeone, S. Szedmak. *Pareto-optimal patterns in logical analysis of data.* RUTCOR Research Report, RRR 7-2001. Also available at *http://rutcor.rutgers.edu/pub/rrr/reports2001/07.pdf.*

[18] Y. Crama, P.L. Hammer, T. Ibaraki. *Cause-effect relationships and partially defined Boolean functions.* Annals of Operationals Research, 16 : 299–325, 1988.

[19] P.L. Hammer. *Partially defined Boolean functions and cause-effect relationships.* Lecture at the International Conference on Multi-Attribute Decision Making Via Or-Based Expert Systems, University of Passau, Germany, April 1986.

[20] J. Eckstein, P.L. Hammer, Y. Liu, M. Nediak, and B. Simeone. *The maximum box problem and its application to data analysis.* Computational Optimization and Application, 23: 285–298, 2002.

[21] O. Ekin, P.L. Hammer, A. Kogan. *Convexity and Logical Analysis of Data.* RUTCOR Research Report, RRR 5-1998. Also available at *http://rutcor.rutgers.edu/pub/rrr/reports1998/05.ps.*

[22] S. Foldes, P.L. Hammer. Disjunctive and Conjunctive Normal Forms of Pseudo-Boolean Functions. *RUTCOR Research Report, RRR 1-2000*, Also available at *http://rutcor.rutgers.edu/pub/rrr/reports2000/01_2000.ps.*

[23] P.L. Hammer, Y. Liu, S. Szedmk, and B. Simeone. *Saturated systems of homogeneous boxes and the logical analysis of numerical data.* Discrete Applied Mathematics, Volume 144, 1-2: 103–109, 2004.

[24] B. Simeone, G. Felici, and V. Spinelli. *A graph coloring approach for box clustering techniques in logic mining.* Book of Abstract of Euro XXII - 22nd European Conference on Operational Research, page 193. The Association of European Operational Research Societies, July 2007.

[25] B. Simeone and V. Spinelli. *The optimization problem framework for box clustering approach in logic mining.* Book of Abstract of Euro XXII - 22nd European Conference on Operational Research, page 193. The Association of European Operational Research Societies, July 2007.

[26] S. Wu and P. Flach. *A scored AUC Metric for Classifier Evaluation and Selection.* Second Workshop on ROC Analysis in ML, Bonn, Germany, August 11, 2005.

[27] J. Huang and C.X. Ling. *Using AUC and Accuracy in Evaluating Learning Algorithms.* IEEE Transactions on Knowledge and Data Engineering vol. 17, no. 3, pp. 299-310, 2005.

[28] C.L. Blake and C.J. Merz. *UCI repository of machine learning databases.* University of California, Irvine, Department of Information and Computer Sciences, 1998. Also available at *URL http://www.ics.uci.edu/ mlearn/MLRepository.html.*

[29] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* 2nd Edition, Morgan Kaufmann, San Francisco, 2005. *URL http://www.cs.waikato.ac.nz/ml/.*

[30] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA.

[31] B. Henderson. *The experience curve reviewed: IV the growth share matrix or product portfolio*, 1973. Also available at *URL http://www.bcg.com/publications/files/*, *Experience_Curve_IV_Growth_Share_Matrix_1973.pdf.*