ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

G. Felici, V. Spinelli

GENETIC PROCEDURE FOR OVER-TRAINING CONTROL IN LOGIC MINING

R. 19 Settembre 2009

Giovanni Felici – Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email :felici@iasi.cnr.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 address a classification method and a technique for effectively dealing with the frequent problem of model over-training. The classification method is Box Clustering, a technique designed to produce classification rules in logic form stating from data in numerical form. The central concept of this methodology is that of homogeneous boxes, that, as will become clear in the paper, is often prone to the drawback of producing over-trained classifiers. The extension proposed is based on the formulation of an integer optimization problem that states the proper relations between the classification performances on the training and on the test set. This model is applied to refine the solution otained with Box Clustering with a standard agglomerative technique, and it turns out to be intractable for classification problems of realistic size and the use of proper heuristics is adviceable. For its solution we test the use of a genetic algorithm on different well known benchmark data sets, and show that the method is fast and robust, making it possible to derive more compact system of boxes in the data space with improved performance on test data.

1. Introduction

The development of algorithms for the analysis of data in logic or binary form has been a widely investigated research topic. In this area many successful methods make use of integer programming models and algorithms to find explanation of training data in the form of logic formulas that then provide both an understandable interpretation of the known data and a precise method to classify new data. In this framework, one of the most established methods is the Logical Analysis of Data (LAD), originated in Crama et al. and Hammer [6] and significantly developed since then both from a theoretical and from an applied point of view: an overview of the implementation of LAD and of its applications is presented in [5]. Similar methods are proposed bt Felici et al. in [1, 2]. The extension of such logic-based methods to non-logic variables can be obtained by proper transformation of numerical data into logic data, often referred to as *discretization* or *binarization* (see for example [3, 4]). The Box Clustering (BC) Technique can be applied to data sets where the variables are numerical or ordered, and extends the logic approach approach without resorting to transformation of the original data [11].

The paper is organized as follows: Section 2 briefly introduces the terminology used in the paper to describe logic data, logic model, and the geometry for BC. In Section 3 we show how the BC approach is potentially exposed to over-training effects. In Section 4 we define an integer optimization problem to eliminate the over-training effect described; we present a genetic-based procedure to solve that optimization problem in Section 5. Finally, in Section 6 we apply this procedure to obtain BC solutions for some standard benchmark data sets used in machine learning, and conclude that the approach is effective and robust.

2. Box Clustering

In this section we briefly summarize the main concept of logic data analysis, referring for convenience to the terminology of [5]. It is assumed that a number of observations are given in the form of n-vectors of Boolean components, and that with each of these vectors an *outcome* is associated. The outcome is assumed to be binary, and according to its values the observations are usually termed *positive* (or *true*), and *negative* (or *false*). A dataset is interpreted as a partially defined Boolean function (*pdBf*), *i.e.* it is assumed that the *outcomes* are the values which an unknown Boolean function takes in a certain subset of vertices of the unit cube. A *pattern* is a logic function based on these Boolean variables, *i.e.*, a product of Boolean variables, possibly complemented. A positive (or negative) pattern is a pattern that takes the value 0 in every negative (resp., positive) point in the dataset, and takes the value 1 in at least some of the positive (resp., negative) points in the dataset. A positive (or negative) *theory* consists of a set of positive (resp., negative) patterns with the property that in every positive (resp., negative) point of the given dataset, at least one of them takes value 1. The determination of such a Boolean function, that is one of the possible *extensions* of the *pdBf* represented by the dataset, is the main task of the LADmethod of [5]. In analogy with the concept of (positive or negative) patterns we introduce here the concept of boxes for the logical analysis of numerical data, recalling that this definition can be easily extended to non-real but totally ordered variables. We define a *box* as a multi-dimensional range delimited by two points: $I(L,U) = \{X \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i \; i = 1, \ldots, n\}$.

A box is called positive (or negative) if it includes some positive (resp., negative) training points, but does not include any negative (resp., positive) training points. 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 join $A \vee B$ to be the box $[S \cup T]$. Such problem is indeed a particular type of clustering with special structures, a reserch topic that received recent attention (see, for example, the work in [14], where convex cones are considered as basic structures to partition the data space).

When we consider a set of observations, the box clustering problem is to find a system of boxes as a solution of a specific optimization problem taken from a general framework [11]. The constraints of this optimization problem are generally based on the geometrical properties of the boxes, and can be of different types:

- *coverage*: every observation is included in at least one box, and every box includes at least one observation.
- *homogeneity*: all the boxes are homogeneous.
- *spanning*: every box is the *box-closure* for the set of observations inside the same box.
- *overlapping*: we generally consider four levels of overlapping condition. 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 define other two conditions: (c) *mild*, *i.e.* only homogeneous boxes can be overlapped, and (d) *weak* otherwise.
- *consistency*: every box must contain at least one point not included in another box, and this implies that we must have at most $n \leq |S|$ boxes.
- *saturation*: if we consider a subset of the above mentioned constraints C, we say that set of boxes is saturated if it satisfies all constraints in C and we cannot join any couple of boxes without violating one of the constraints in C. This constraint is generally consider a meta-constraint because it is based on a set of other constraints.

In the literature, the *agglomerative approach* is a well-established algorithm framework for the local search of a saturated box system satisfying an input set of constraints [9]. A homogeneous box of positive (or negative) points in BC is equivalent to a positive (resp., negative) pattern in [5] and thus a system of positive boxes in BC is equivalent to a positive (resp., negative) theory in LAD. Besides, a well-known BC problem is *"the maximum box problem"* [8], where the homogeneous box including the highest number of positive (or negative) points is to be found. We can identify this problem in the above described framework by considering (a) homogeneity constraint, (b) box set made by only one box, and (c) the number of points covered by the system of boxes as the objective function to be maximized.

3. Over-training Control

The scheme of a BC-based classifier is shown in Figure 3. It requires four input parameters: (a) a box set BS, (b) the training set Tr, (c) the weight function $w()$, and (d) the point p to classify (*i.e.*, p belongs to the test set Ts).

The weight function $w(B, \mathrm{Tr}, p)$ assigns a weight to each box $B \in BS$. This weight 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 the weight function as the characteristic function shown 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, and only the *spanning* constraint holds, then we can define the weight function as the *manhattan distance* (see Equation 2), a reasonable metric according to the box geometry defined in Section 2.

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

If the weight function $w()$ explicitly depends on Tr it can be defined as a gravitational model, where $w(B,p)$ depends on the mutual position of the point p and the point of Tr contained in B. 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 the box. The core of the *classification* schema is the *choose-best-index()* function. This function choose the index of the box having the best weight (*i.e.*, minimum); we may in general assume that there exists a non-empty subset $\overline{B} \subseteq BS$ such that each box $B \in \overline{B}$ has value equal to the best value. Two exclusive conditions must then be considered:

- 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 a 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. If we consider a dataset $S = Tr \cup Ts$ with n classes, and then apply the classification function to every point in Ts, the resulting confusion matrix $M_c = M_c(Ts) = \{m_{ij}\}$ can be derived, 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. From the definition of M_c we can always consider $i, j = 1, \ldots, n+1$: 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. M_c is univoquely defined by the pair $(classification, Ts)$, and thus it depends on $(BS, Tr, w(), \textit{choose-best-index})$. From the fact that $\sum_{i=1,n} m_{ii}$ is the number of the correctly classified observations of Ts, we can define the error function $Err(M_c) = \sum_{i \neq j} m_{ij}$.

Proposition 3.1. *If the box set* BS *satisfies, at least, the constraints* {*coverage, homogeneous*} *on the point set* S*, then* $Err(M_c(S)) = 0$.

Proof:
Let us consider $p \in S$. 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 $B' \in BS$ then B and B' have the same class. Since every BC classifier, based on the schema shown in Figure 3, must rightly classify p, we can conclude that $Err(M_c(S)) = 0$. □

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

Proof: For every BC classifier, we can check that $Err(M_c(S)) = Err(M_c(Tr)) + Err(M_c(Ts))$. From 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. To avoid this situation, we need a method to *simplify* the logic model and *control* the error function.

4. The Optimization Problem

Given the box set B and the point set S, for which Proposition 3.2 holds, we can consider the confusion matrix $M_0 = M_c(BS, Ts) = \{m^0_{ij}\}$ and, for every $\emptyset \neq B' \subseteq B$, $M = M_c(B', Ts) = \{m_{ij}\}$, and $l \in [0,1]^n$. We can then define a l-similarity property for the matrix M with respect to M_0:

$$M \equiv_l M_0 \Leftrightarrow \begin{cases} m_{ii} \geq (1 - l_i) m^0_{ii} & i = 1, \ldots, n+1. \\ m_{n+1,i} = 0 \end{cases}$$

The condition $M \equiv_l M_0$ requires that the classification results of B' in Ts are similar class by class to the results of B. If $\emptyset \neq B' \subset B$ then Proposition 3.2 could not hold for B', and this means that $Err(M_c(B', Tr)) \geq 0$. For the above considerations, it is meaningful to consider the objective function

$$\Delta(B', Tr, Ts) = |Err(M_c(B', Tr)) - Err(M_c(B', Ts))|.$$

As the dimensions of $|Tr|$ and $|Ts|$ may be different, we turn to a size-indendent measure for the errors on a generic set P:

$$Err'(M_c(B', P)) = \frac{Err(M_c(B', P))}{|P|}$$

changing accordingly the definition of Δ. From the consideration that B' could be *less over-trained* than B, it is appropriate to define an optimization problem to search the best B' having this property.

$$\begin{cases} \min & \Delta(B', Tr, Ts) \\ s.t. & \emptyset \neq B' \subseteq B \\ & M_c(B', Ts) \equiv_l M_0 \end{cases} \tag{3}$$

Here we require that a new, smaller set of boxes B' is selected, for which the structure of the confusion matrix on the test set is similar (according to l) to that of the complete set B and where the errors on training and test are as similar as possible.

If we consider the characteristic boolean vector $x = (x_1, \ldots, x_m)$ of B, *i.e.* $|B| = m$ and $b_i \in B \Leftrightarrow x_i = 1$ then Problem 3 can be equivalently defined:

$$\begin{cases} \min & \Delta(x, Tr, Ts) \\ s.t. & \sum_{i=1,m} x_i \geq 1 \\ & M_c(x, Tr, Ts) \equiv_l M_0 \\ & x_i \in \{0, 1\} \end{cases} \tag{4}$$

The condition $\emptyset \neq B'$ is changed into $\sum_{i=1,m} x_i \geq 1$, but if we consider $l_i > 0$ for every i then $\sum_i x_i \geq n \geq 1$, and the constraint always holds. Δ is a positive and bounded function by definition, and in particular $\Delta(x, P_1, P_2) \in [0, 1]$ for every (P_1, P_2). We can then represent the objective function with a barrier formulation as the following one:

$$f(x, P_1, P_2, l) = \begin{cases} \Delta(x, P_1, P_2) & if M_c \equiv_l M_0 \\ 1 & otherwise \end{cases} \quad (5)$$

With the above transformations, we formulate an un-constrainted version of the optimization problem for the over-training control in BC:

$$\begin{cases} \min & f(x, Tr, Ts, l) \\ s.t. & x_i \in \{0, 1\} \qquad i = 1, \ldots, m \end{cases} \quad (6)$$

Problem 6 can be formulated as an Integer Linear Problem with $n(2m + 1)$ binary variables and $m(2n + 1) + 2 + n_c$ constraints such that $|Tr| + |Ts| = n$ and n_c is the number of classes. This means that it depends quadratically on the size of the BC problem. If we consider a small BC problem, $m = 1000$, $n = 100$, and $n_c = 2$, we must define a problem with $200,100$ binary variables and $201,004$ constraints. Clearly such problem is hard to solve even for small values of m and n; for this reason, we resort to a heuristic solution approach, and in particular to a genetic method that appears to be well suited for the purposes of this application.

5. Genetic-based Procedure

In this section we describe the straight-forward application of a genetic search engine to the optimization problem described in the previous section. For this implementation we have chosen the *PGAPack* software system [17, 18] that presents several interesting features, mentioned below.

A scheme of the algorithm is given in Figure 6; the genetic-based procedure for BCconsiders k independent runs of 2-fold cross validations for the chosen combination of parameter values, and is based on four functions: (a) *cross-split*(), (b) *local-search*(), (c) *genetic*(), and (d) $f()$ (the fitness function). The *data-split* function randomly splits the input data set S into 2 parts (Tr, Ts) of fixed size (the size being determined by the pair $(|S|, k)$), where Tr is the training set, used to build the BC model, and Ts is the test set, used for validation. The *local-search* function is the implementation of the well-known agglomerative algorithm for BC [8]. The *genetic* function is the implementation of the genetic-based engine to solve the Problem 6, where the fitness function is the objective function of problem 6. The choice *PGAPack* ([17, 18]) is mainly based on three features: (a) itis highly configurable system, (b) there eists a scalar and a parallel version of the engine, and (c) its interface is very simple and easy-to-use. The algorithm in Figure 5 is characterized by steps 6 and 7; these 2 steps, in a standard applcation ov would be replaced by $d = \mathrm{Err}(BS, Tr, Ts)$. The genetic function, *i.e.* step 6 in Figure 5, is expanded in Figure 5, where we can see the calls to the *PGAPack* functions (see [18] for further details).

The function *evaluate*() is the implementation of the objective function used for the genetic engine. In function *genetic*(), steps 2-3 define the element of the search space, *i.e.* a binary vector of length *nb*. Step 4 define the type of optimization problem, *i.e.* the minimization of the objective function. Step 5 is the creation of the *PGAPack* data structure, while step 6 sets the upper bound for the size of the population in each step. Steps 7-8 start the genetic engine, while step 9 cleans all the *PGAPack* data structures used during the execution. The current version of our genetic-BC system uses the scalar version of the *PGAPack* system; its parallel version may be deployed in solving larger problems, *i.e.* more than 10000 observations. The genetic engine has an heavy impact on the final results, as shown later. We have used four parameter sets for each run of the procedure, as shown in Table 5. Such parameters are the main ones that

8.

```
    function overtraining(S, k, P)
    {
 1    Δ* = 1;
 2    Best BS = ∅;
 3    for(i = 0; i < k; i++){
 4      (Tr, Ts) = data-split(S, k, i);
 5      BS = local-search(Tr, P);
 6      BS^ = genetic(BS, Tr, Ts, P);
 7      d = f(BS^, Tr, Ts);
 8      if(d < Δ*){
 9        Best BS = BS^;
10        Δ* = d;
11      }
12    }
    }
```

Figure 2: Genetic schema for BC

```
    function genetic(BS, Tr, Ts, P)
    {
1     PGAContext * ctx;
2     nb = |BS|;
3     ty = PGA_DATATYPE_BINARY;
4     op = PGA_MINIMIZE;
5     ctx = PGACreate(&argc, argv, ty, nb, op);
6     PGASetPopSize(ctx, P.pop);
      /* P.pop = 1000 */
7     PGASetUp(ctx);
8     PGARun(ctx, evaluate);
9     PGADestroy(ctx);
    }
```

Figure 3: Genetic function genetic()

defibe the BCmethod, related with the type of constraints imposed on the system of boxes, on the way the boxes are clustered, and on the method to assign a single element to one of the boxes in the solution. Even if there are other low-level parameters, they are not relevant within the scope of the experiments, and they have been omitted for brevity.

Table 1: Summary for parameter sets.

	Overlapping	Local search	Classification
P_1	mild	random	nearby
P_2	none		
P_3	mild	greedy	
P_4	none		

6. Experiments

In order to empirically evaluate the efficiency of the procedure described in Section 5 and its use in data analysis, we have applied it to five frequently-used data sets, taken from the repository of the University of California, Irvine (UCI), see [12]: Pima Indian diabetes (*pima*), Congressional voting records (*voting*), Wisconsin Prognostic Breast Cancer (*wpbc*), Image Segmentation data (*segment*), and Waveform Database Generator (*wave*). One further data set, *ictus*, is an unpublished medical archive about the ictus disease. The key parameters of these six data sets are listed in Table 2: columns 2-4 indicate, for each data set, the number of records, of features, and of classes, respectively. Table 2 also indicates the presence/absence of missing values (column 5).

Table 2: Summary for considered data sets.

Data set	Observations	Attributes	Classes	Missing	l-similarity	k-fold
wpbc	194	32	2	n	5%	2
voting	342	16	2	y	5%	2
ictus	698	37	2	y	10%	2
pima	768	8	2	n	10%	2
segment	2086	18	7	n	10%	3
wave	5000	21	3	n	10%	5

According to the dimension of each data set, we have consider different values for the k parameter in the cross validation procedure and the l similarity parameter in the genetic function. The columns 6 and 7 in Table 2 shows our run-time choices for these two parameters. A representative examples of the results obtained is given in n Figure 4, where we show some sequences of solutions found by the genetic engine for *pima*: the black lines are the values of $Err'(M_c(B', Tr))$, while the gray ones are the values of $Err'(M_c(B', Ts))$. At the very first generations of the genetic population, the two groups of lines are very distant (as predicted by Proposition 3.2), but they converge quite rapidly. Similar behaviors are observed for the other data sets. In Figures 5-6 we

can see the performances of the function *genetic*() by comparing the input BS and the output one. The horizontal axes represents the number of boxes, while the vertical axes the values of the fitness function f (see Problem 6). The black circles are the BC solutions coming out from the *local-search* function, while the gray ones are the genetic solutions. Many duplicated and/or very similar solutions have appeared during the search, and they have been omitted from the plots. The solid (or dashed) box is the *box-closure* of the black (respectively, gray) points, and their mutual positions give us some hints for the interpretation of the results:

- The main effect of the genetic engine is to shift the dashed box to the left of the black one, *i.e.* the genetic solutions must have less boxes than the *local-search* one. This aspect can be clearly seen in Figure 6.
- The side effect of the genetic engine is to shift the dashed box to the top of the black one, and this means that $Err(B', Tr \cup Ts) \geq Err(B, Ts)$, see Figure 5.

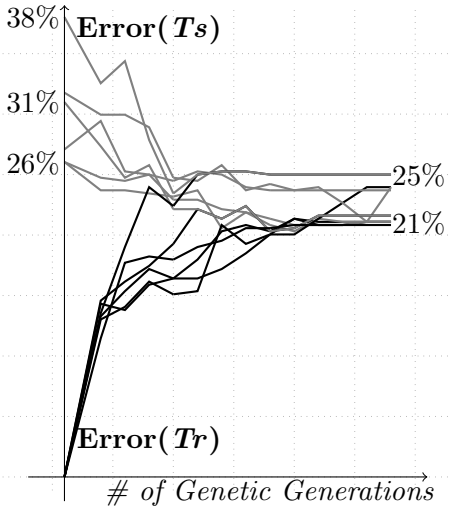

Figure 4: Genetic tracing for *pima*

6.1. Classification performance

Even if the focus of this work is to analyze the behavior of the *genetic* function, several considerations can be dranw from the classification performances of the *overtraining* procedure. We thus consider the mean and standard deviation of the errors over the different experiments. In Table 3 we have summed up the best results for all the data sets. Evidently, it is not possible to define the single best result, but we have focused on the non-dominated points in the planes defined by the number of boxes and the percentage of errors, choosing as best the points that belong to the Pareto set. Columns 2 and 3 are the results of the *local-search* function, while columns 4 and 5 are the results of the *genetic* one. In Table 4, we have a statistical analysis of the *result differences* for each input data set $S = (Tr, Ts)$: for every run-time parameter sets P_i we have chosen, in a hierarchical way, the BS having the minimum error rate and the minimum number of boxes; on these four couples of values, we have evaluate the mean and the standard

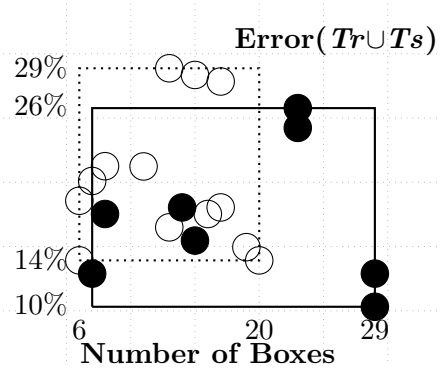

Figure 5: Results for *wpbc*

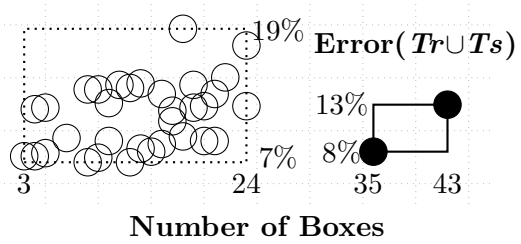

Figure 6: Results for *ictus*

deviation of both the number of boxes and the error rate. The results in Table 4 show us that each P_i (Table 5), has an heavy influence on $|BS|$; in other words, $\sigma(|BS|)$ is generally very large, but the error rate is *very stable*, *i.e.*, $\mu(Err)$ is almost constant and $\sigma(Err)$ is narrow. In general we have witnessed a robust behavior of the genetic-based procedure for all the different data sets considered. The magnitude of the effect of the procedure varies among the different benchmarks, due mainly to the different amount of noise present in the data.

7. Conclusion

In this paper we have proposed a genetic based algorithm to refine the solutions obtained using Box Clustering for supervised classification of non-logic data with logic based models, with the main purpose of eliminating the frequent over-training behavior of such classification tool. The over-training control problem has been formulated as an optimization problem where we constraint the structure of the confusion matrix in a desider way and, moreover, we require the minimal distance between the error on training set and the error on test set. The straight forward formulation of this optimization problem grows in size very rapidle with the dimensions of the data analyzed, making it untractable for reasonable applications. For this problem a standard genetic algorithm has proved to find solutions of good quality. The application of the genetic procedure amounts to the selection of a subset of the box set identified by the standard Box Clustering algorithm; such subset is typically smaller that the original one and exhibits improved recognition performances on testing data.

Table 3: Best Results for data sets

	Local Step		Genetic Step	
Data set	$\|BS\|$	$f()$	$\|\widehat{BS}\|$	$f()$
wpbc	8	17.53%	6	18.56%
voting	63	19.88%	20	16.67%
	63	19.88%	10	17.54%
	64	21.64%	23	18.13%
ictus	36	8.02%	3	7.59%
	43	12.46%	13	7.02%
pima	43	15.62%	22	20.44%
	39	19.01%	12	22.92%
segment	20	4.55%	14	4.84%
	31	5.47%	11	7.57%
wave	97	14.94%	11	17.80%
	93	16.26%	30	16.30%
	267	13.04%	109	13.98%

Table 4: Average Results for data sets

		Number of boxes		Total error	
Data set	k-fold	μ	σ	μ	σ
wpbc	1	13.00	5.15	19.33%	5.98%
	2	13.00	4.30	20.11%	4.53%
voting	1	24.50	7.50	20.47%	1.79%
	2	22.50	3.04	19.66%	3.77%
ictus	1	4.75	1.48	8.41%	0.83%
	2	10.50	1.50	7.52%	0.65%
pima	1	38.75	16.18	20.05%	0.60%
	2	38.75	9.91	19.92%	2.01%
segment	1	24.00	7.45	6.92%	1.35%
	2	23.75	10.28	6.87%	0.53%
	3	21.75	4.97	8.12%	1.15%
wave	1	53.50	37.45	18.98%	3.49%
	2	55.50	34.77	18.59%	3.67%
	3	64.50	47.55	17.78%	4.63%
	4	52.25	38.31	17.50%	1.99%
	5	63.25	43.08	18.55%	3.26%

References

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

[2] 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.

[3] J. Mugan and K. Truemper. Discretization of rational data, in *Mathematical Methods for Knowledge Discovery and Data Mining*, G. Felici and C. Vercellis eds., IGI Global - Information Science Reference, 1-23, 2008.

[4] P.L. Hammer, I.I. Lozina. Boolean Separators and Approximate Boolean Classifiers. *RUTCOR Research Report, RRR 14-2006*,

[5] 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.

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

[7] 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.

[8] J. Eckstein, P. 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.

[9] 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.

[10] 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.

[11] 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.

[12] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, University of California, Irvine, Department of Information and Computer Sciences, 1998.

[13] P.L. Hammer. Optimization models for logical analysis of Data. *Workshop on Data Mining and Mathematical Programming*, Centre de Recherches mathématiques Montréal, Québec, Canada, October 10-13, 2006.

[14] P.M. Pardalos. Supervised and Unsupervised Consistent Biclustering. *Workshop on Data Mining and Mathematical Programming*, Centre de Recherches mathématiques Montréal, Québec, Canada, October 10-13, 2006.

[15] P.M. Pardalos and P. Hansen. Data Mining and Mathematical Programming, CRM vol 45, American Mathematical Society, 2008.

[16] G. Felici, P. Bertolazzi, M.R. Guarracino, A. Chinchuluun, and P.M. Pardalos. Logic formulas based knowledge discovery and its application to the classification of biological data, BIOMAT 2008, Rubem P. Mondaini Editor, World Scientific, pp. 265 279 (2009). ISBN: 978-981-4271-81-3.

[17] PGAPack Parallel Genetic Algorithm Library. Argonne National Laboratory, Mathematics and Computer Science Division.

[18] D. Levine. Users Guide to the PGAPack Parallel Genetic Algorithm Library. Argonne National Laboratory, Mathematics and Computer Science Division. ANL-95/18.