Effective Classification Using a Small Training Set Based on Discretization and Statistical Analysis

Renato Bruni and Gianpiero Bianchi

Abstract—This work deals with the problem of producing a fast and accurate data classification, learning it from a possibly small set of records that are already classified. The proposed approach is based on the framework of the so-called Logical Analysis of Data (LAD), but enriched with information obtained from statistical considerations on the data. A number of discrete optimization problems are solved in the different steps of the procedure, but their computational demand can be controlled. The accuracy of the proposed approach is compared to that of the standard LAD algorithm, of support vector machines and of label propagation algorithm on publicly available datasets of the UCI repository. Encouraging results are obtained and discussed.

Index Terms—Classification algorithms, data mining, machine learning, discrete mathematics, optimization

1 Introduction

Given a set of data grouped into *classes*, the problem of predicting which class new data should receive is called *classification* problem. Many approaches to this problem have been proposed, based on different considerations and data models. Established ones include: Neural Networks, Support Vector Machines, k-Nearest Neighbors, Bayesian approaches, Decision Trees, Logistic Regression, Boolean approaches (see for references [1], [2], [3], [4], [5], [6], [7]). Each approach has several variants, and algorithms can also be designed by mixing approaches. Specific techniques may better fit to specific classification contexts, but one approach that is generally considered quite effective for many practical applications is *Support Vector Machines (SVM)* [8]. SVM are based on finding a separating hyperplane that maximizes the margin between the extreme training data of opposite classes, possibly after a mapping to a higher dimensional space, see also [9], [10].

Roughly speaking, the larger is the training set, the more information it contains, the more accurate the learned classifier can be, even if clearly there are several aspects involved. Unfortunately, in many important applications, labeled data are difficult or expensive to obtain, and a classification methodology able to be accurate using *small* training sets would be very useful. On the contrary, unlabeled data may be relatively easy to collect. Therefore, techniques have been developed for improving a classification by using also a large amount of unlabeled data, that is called *validation set*. Those techniques can be introduced into several of the approaches listed above, obtaining *semi-supervised* classifiers (see [11], [12], [13] and references therein). In the case of SVM they are called Transductive support vector machines (TSVM) [10], and are based on the concepts described above but also force the separating hyperplane to be far away from the unlabeled data. Another major framework in semi-supervised learning techniques is *Label Propagation* (LP), initially proposed in [14], see also [15]. This technique works by constructing a similarity graph over all the records in the input dataset, and by propagating the labels of the labeled records to the unlabeled ones according to the intrinsic data manifold structures collectively revealed by a large number of data records.

However, no single algorithm is currently able to provide the best performance on all datasets, and this seems to be inevitable [16]. Predicting which algorithm will perform best on a specific dataset has become a learning task on its own, belonging to the area called meta-learning [17]. Therefore, techniques based on the aggregation of a set of different (and hopefully complementary) classifiers have been investigated. They are called *Ensemble techniques*, and they include Boosting [18], [19] and Bagging [20]. Roughly speaking, those techniques generate many weak learners and combine their outputs in order to obtain a classification that is both accurate and robust. Those weak learners may be based on several classification approaches.

On the other hand, one interesting Boolean approach to classification is the *Logical Analysis of Data* (LAD) [21], [22], [23], [24], [25], that is inspired by the mental processes that a human being applies when learning from examples. In this approach, data should be encoded into binary form by means of a discretization process called *binarization*. This is done by using the training set for computing specific values for each field, called *cut-points* in the case of numerical

• *R. Bruni is with the Department of Computer, Control and Management Engineering (DIAG), University of Rome "Sapienza," Via Ariosto 25, Roma 00185, Italy. E-mail: bruni@dis.uniroma1.it.*
• *G. Bianchi is with the Department for Integration, Quality, Research and Production Networks Development (DIQR), Italian National Institute of Statistics "Istat," Via Tuscolana 1788, Roma 00173, Italy. E-mail: gianbia@istat.it.*

Manuscript received 24 July 2014; revised 26 Feb. 2015; accepted 14 Mar. 2015. Date of publication 0 . 0000; date of current version 0 . 0000. Recommended for acceptance by T. Li.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2416727

fields, that split each field into *binary attributes*. Discretization is also adopted in other classification methodologies, such as decision trees, and several ways for selecting cut-points exists, such as entropy based ones (see e.g. [6], [26]). The selected binary attributes constitute a *support set*, and are combined for generating logical rules called *patterns*. Patterns are used to classify each unclassified record, on the basis of the sign of a weighted sum of the patterns *activated* by that record. LAD methodology is closely related to decision trees and nearest neighbor methods, and constitutes an extension of those two approaches, as shown in [24].

In this paper, we propose the following original enhancements to the LAD methodology. First, the idea of evaluating the quality of each cut-point for numerical fields and of each binary attribute for categorical fields, and a criterion for doing so. Such quality values are computed by using information extracted from the training set, and are taken into account for improving the selection of the support set. The support set selection can therefore be modeled as a *weighted set covering* problem, and also as a *binary knapsack* problem (see e.g. [27], [28]). In a related work, Boros et al. [29] consider the problem of finding essential attributes in binary data, which again reduces to finding a small support set with a good separation power. They give alternative formulations of such problem and propose three types of heuristics for solving them. An analysis of the smallest support set selection problem within the framework of probably approximately correct learning theory, and algorithms for its solution, is also in [30].

Moreover, the classification of the test set is not given here simply on the basis of the sign of the weighted sum of activated patterns, but by comparing that weighted sum to a suitable *classification threshold*. Indeed, we propose to compute both the values of pattern weights and the value of classification threshold in order to minimize errors, by solving a mixed integer linear programming problem. The objective of minimizing errors is pursued by (*i*) minimizing classification errors on records of the training set and by (*ii*) reproducing in the test set the class distribution of the training set. Pattern weights and classification threshold are in fact parameters for the classification procedure, and, in our opinion, this should allow obtaining the best choice of these parameters for the specific dataset, partially overcoming the parameter tuning or guessing phase that always represents a difficult and questionable step.

The known LAD procedure is recalled in Section 2. We refer here mainly to the "standard" procedure, as described in [23], although other variants have been investigated in literature (e.g. [31]). The original contributions of this work begin with Section 3, which explains motivations and possible criteria for evaluating the quality of cut-points. In particular, we derive procedures for dealing with cut-points on continuous fields having *normal* (Gaussian) distribution, on discrete fields having *binomial* (Bernoulli) distribution, or on general numerical fields having unknown distribution. This latter approach is used also for qualitative, or categorical, fields. The support set selection problem is then reformulated as weighted set covering and as binary knapsack in Section 4. After that, patterns are generated, and computation of pattern weights and classification threshold by using the proposed mixed integer model is described in Section 5. Results of the proposed procedure on publicly available datasets of the UCI repository [32] are analyzed and compared to those of the standard LAD methodology, and also to those of SVM (in its implementation LIBSVM [33], currently deemed to be among the most effective classifiers), TSVM (in its implementation UniverSVM [34]), and LP (in its implementation scikit-learn [35]) in Section 6. Main notation is summarized in the Appendix.

2 Classifying with the LAD Methodology

The structure of records, called *record scheme* R, consists of a set of fields f_i, with $i = 1, \dots, m$. A *record instance* r, also simply called *record*, consists of a set of values v_i, one for each field. A record r is *classified* if it is assigned to an element of a set of possible classes C. In many cases, C has only two elements, and we speak of *binary classification*. We will hereinafter consider this case. Note, however, that the proposed procedure, *mutatis mutandis*, could also be used for the case of multiple classes. A positive record instance is denoted by r^+, a negative one by r^-.

For classifying, a *training set* S of classified records is given. Denote by S^+ the set of its positive records and by S^- the set of its negative ones. Sets S^+ and S^- constitute our source of information. A set of records used for evaluating the performance of the learned classifier is called *test set* T. The *real* classification of each record $t \in T$ should be known. We compare the classification of T given by the learned classifier, also called *predicted* classification, to the real classification of T: the differences are the classification errors of our classifier. A positive training record is denoted by s^+, a negative one by s^-. A positive test record is denoted by t^+, a negative one by t^-.

LAD methodology begins with encoding all fields into binary form. This process, called *binarization*, converts each (non-binary) field f_i into a set of binary *attributes* a_i^j, with $j = 1, \dots, n_i$. The total number of binary attributes is $n = \sum_{i=1}^{m} n_i$. Note that the term "attribute" is not used here as a synonym for "field". A binarized record scheme R_b is therefore a set of binary attributes a_i^j, and a binarized record instance r_b is a set of binary values $b_i^j \in \{0, 1\}$ for those attributes.

$$R_b = \{a_1^1, \dots, a_1^{n_1}, \dots, a_m^1, \dots, a_m^{n_m}\},$$
$$r_b = \{b_1^1, \dots, b_1^{n_1}, \dots, b_m^1, \dots, b_m^{n_m}\}.$$

For each qualitative fields f_i, all values can simply be encoded by means of a logarithmic number of binary attributes a_i^j, so that n_i binary attributes can binarize a quantitative field having up to 2^{n_i} different values. For each numerical field f_i, on the contrary, we introduce n_i thresholds called *cut-points* $\alpha_i^1, \dots, \alpha_i^{n_i} \in \mathbb{R}$, and the binarization of a value v_i is obtained by considering whether v_i lies above or below each α_i^j. Cut-points α_i^j should be set at values representing some kind of watershed for the analyzed phenomenon. Generally, α_i^j are placed in the middle of specific couples of data values v_i' and v_i'' :

$$\alpha_i^j = (v_i' + v_i'')/2.$$

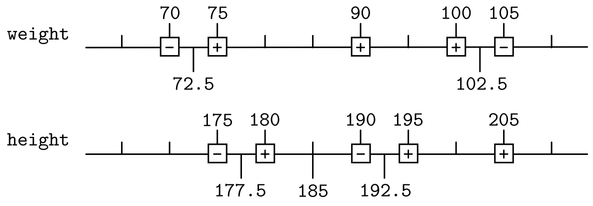
This can be done for each couple v_i' and v_i'' belonging to records from opposite classes that are adjacent on f_i. Cut-points α_i^j are then used for binarizing each numerical field f_i into the binary attributes a_i^j (also called level variables). The values b_i^j of such a_i^j are

$$b_i^j = \begin{cases} 1 & \text{if} \quad v_i \geq \alpha_i^j \\ 0 & \text{if} \quad v_i < \alpha_i^j. \end{cases}$$

Example 1. We have records representing persons having fields `weight` (in Kg.) and `height` (in cm.), and a positive [respectively negative] classifications meaning "is [resp. is not] a professional basketball player". Consider the following training set:

	weight	height	pro.bask.player.?
	90	195	yes
S^+	100	205	yes
	75	180	yes
	105	190	no
S^-	70	175	no

We now plot values belonging to positive [resp. negative] records by using a framed + [resp. −]. Cut-points obtainable from this set S are $\alpha^1_{\texttt{weight}}$=72.5, $\alpha^2_{\texttt{weight}}$= 102.5, $\alpha^1_{\texttt{height}}$=177.5, $\alpha^2_{\texttt{height}}$=185, $\alpha^3_{\texttt{height}}$= 192.5. Corresponding binary attributes obtainable are $a^1_{\texttt{weight}}$, meaning: weight $\geq$ 72.5 Kg., $a^2_{\texttt{weight}}$, meaning: weight $\geq$ 102.5 Kg., $a^1_{\texttt{height}}$, meaning: height $\geq$ 177.5 cm., $a^2_{\texttt{height}}$, meaning: height $\geq$ 185 cm., $a^3_{\texttt{height}}$, meaning: height $\geq$ 192.5.



A set of binary attributes $\{a_i^j\}$ used for binarizing a dataset S is called *support set* U. A support set is exactly separating if no pair of positive and negative records of S have the same binary encoding. A single data-set may have several possible exactly separating support sets. Since the number of binary attributes obtainable in practical problems is often very large, and many of them may be not needed to explain the analyzed phenomenon, we are interested in selecting a small (or even the smallest) exactly separating support set. By using a binary variable x_i^j for each a_i^j, such that

$$x_i^j = \begin{cases} 1 & \text{if } a_i^j \text{ is retained in the support set} \\ 0 & \text{if } a_i^j \text{ is excluded from the support set} \end{cases}$$

the integer programming problem (1) should be solved. For every pair of positive and negative records s^+, s^- we define $I(s_b^+, s_b^-)$ to be the set of couples of indices (i, j) where the binary representations of s^+ and s^- differ, except, under special conditions [23], for the indices that involve monotone values. This problem has a peculiar mathematical form called *set covering* [27], [28]: the objective (sum of all the binary variables) minimizes the cardinality of the support set; the constraints (sums of binary variables ≥ 1) impose retaining at least one binary attribute for each set of them producing different binarizations for any pair of positive and negative records

$$\begin{cases} \min\limits_x \sum_{i=1}^m \sum_{j=1}^{n_i} \; x_i^j \\ \text{s.t.} \sum\limits_{(i,j)\in I(s_b^+, s_b^-)} x_i^j \geq 1 \quad \forall I(s_b^+, s_b^-), \; s^+ \in S^+, \; s^- \in S^- \\ x_i^j \in \{0, 1\} \end{cases} \quad (1)$$

Note that this selection does not aim to improve the classification power, and actually "the smaller the chosen support set, the less information we keep, and, therefore, the less classification power we may have" [23]. Instead, it is necessary for reducing the computational burden of the remaining part of the procedure, which may otherwise become impracticable. Indeed, a non-optimal solution to (1) would not necessarily worsen the classification power [23], [29]. Since different support sets correspond to different alternative binarizations, hence to actually different binarized record, the *support set selection* is a key point.

Example 2. Continuing Example 1, by solving to optimality the above set covering problem (1), we have the alternative support sets $U_1 = \{a^2_{\texttt{weight}}, a^1_{\texttt{height}}\}$ and $U_2 = \{a^1_{\texttt{weight}}\, a^2_{\texttt{weight}}\}$. Moreover, an approximate solution is $U_3 = \{a^1_{\texttt{weight}}, a^2_{\texttt{weight}}, a^1_{\texttt{height}},\}$. The corresponding alternative binarizations of S are:

	U_1		U_2		U_3		
	$b^2_{\text{we.}}$	$b^1_{\text{he.}}$	$b^1_{\text{we.}}$	$b^2_{\text{we.}}$	$b^1_{\text{we.}}$	$b^2_{\text{we.}}$	$b^1_{\text{he.}}$
	0	1	1	0	1	0	1
S^+	0	1	1	0	1	0	1
	0	1	1	0	1	0	1
	1	1	1	1	1	1	1
S^-	0	0	0	0	0	0	0

The selected support set U is then used to create patterns. A *pattern* P is a conjunction ($\wedge$) of literals, which are binary attributes $a_i^j \in U$ or negated binary attributes $\neg a_i^j$. Given a binarized record r_b, that is a set of binary values $\{b_i^j\}$ for the above binary attributes, each literal of P receives a value: $b_i^j \in \{0,1\}$ for literal a_i^j; $(1 - b_i^j) \in \{0,1\}$ for literal $\neg a_i^j$. We have that $P = 1$ if all literals of P are 1, $P = 0$ otherwise. We say that a pattern P *covers* a record r if the set of values $r_b = \{b_i^j\}$ makes $P = 1$. A *positive* pattern P^+ is a pattern covering at least one positive record r^+ but no negative ones. A negative pattern P^- is defined symmetrically. Patterns can be viewed as rules governing the analyzed phenomenon. We denote as $P(r)$ the value of pattern P applied to record r:

$$P(r) = \begin{cases} 1 & \text{if} \quad P \text{ covers } r \\ 0 & \text{if} \quad P \text{ does not cover } r. \end{cases}$$

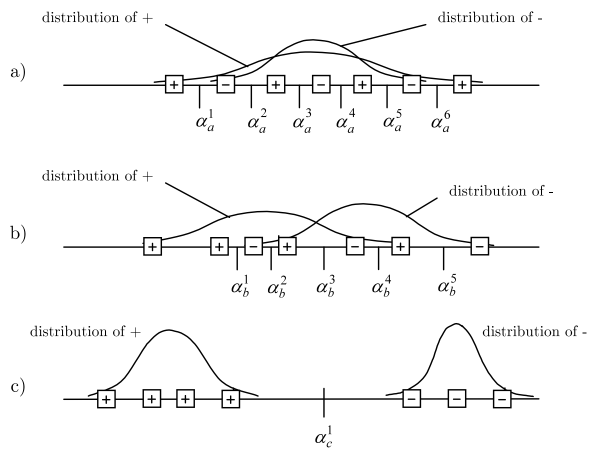

a)
distribution of +
distribution of -
α_a^1 α_a^2 α_a^3 α_a^4 α_a^5 α_a^6
b)
distribution of +
distribution of -
α_b^1 α_b^2 α_b^3 α_b^4 α_b^5
c)
distribution of +
distribution of -
α_c^1

Fig. 1. Examples of cut-points in different conditions.

Example 3. By continuing Example 2, a positive pattern obtained using the support set U_1 is $P_1 = \neg a^2_{\texttt{weight}} \wedge a^1_{\texttt{height}}$. This means `weight` $<$ `102.5` Kg. and `height` $\geq$ `177.5` cm. Recall P_1 is a pattern if $P_1(s^+) = 1$ for at least some $s^+ \in S$ and $P_1(s^-) = 0$ for all $s^- \in S$. Indeed, we have $P_1(s^+) = 1$ for all $s^+ \in S$ and $P_1(s^-) = 0$ for all $s^- \in S$. Another pattern, obtained using support set U_3, is $P_2 = a^1_{\texttt{weight}} \wedge \neg a^2_{\texttt{weight}} \wedge a^1_{\texttt{height}}$. P_2 appears to be even more appropriate than P_1, since it means "one is a professional basketball player if has a medium weight (`weight` $\geq$ `72.5` Kg. and `weight` $<$ `102.5` Kg.) and height above a certain value (`height` $\geq$ `177.5` cm.)". $P_2(s^+) = 1$ for all $s^+ \in S$, $P_2(s^-) = 0$ for all $s^- \in S$.

Positive patterns can be generated by means of two types of approaches: top-down, i.e. by removing one by one literals from the conjunction of literals covering a single positive record until no negative records are covered, or bottom-up, i.e. by conjoining one by one single literals until obtaining a conjunction covering only positive records. Negative patterns can be generated symmetrically. Also the number of generated patterns may be too large, so a pattern selection step can be performed. This is done in [23] by solving another set covering problem, whose solution gives the set of the indices H^+ of selected positive patterns and that of the indices H^- of selected negative patterns, with $H = H^+ \cup H^-$. Weights w_h are now assigned to all patterns in H, with $w_h \geq 0$ for $h \in H^+$ and $w_h \leq 0$ for $h \in H^-$, by using criteria described in [23]. We skip detail here since we will discuss this again and propose a new approach in Section 5. Finally, each new record r is classified according to the positive or negative value of the following weighted sum, called *discriminant* and denoted by $\Delta(r)$

$$\Delta(r) = \sum_{h \in H^+} w_h P_h(r) + \sum_{h \in H^-} w_h P_h(r) = \sum_{h \in H} w_h P_h(r).$$

3 Evaluation of Binary Attributes

We remarked that selecting a small support set is computationally necessary, but that excluding attributes means losing information. Therefore, we propose to evaluate the *quality* (the separating power) of each attribute and to perform such a selection taking into account this evaluation. In the following Fig. 1, we give three examples of numerical fields (a,b,c). In each case, we draw (in the area above the horizontal line) "qualitative" distributions densities of a large number of values from positive and negative records, and report (on the same line) a smaller sample of those values. Very intuitively, cut-points obtainable in case a) are the worst ones (they do not appear very useful for separating the two classes), while the cut-point of case c) is the best one (it has a good "separating power"). Moreover, the different cut-points of case b) do not have the same quality.

To estimate this, we analyze how α_i^j divides the two classes, *even if* the real classification step will use patterns. Different estimators could of course be designed, however results show that the proposed technique is able to improve accuracy with respect to the standard LAD procedure.

Given a single cut-point α_i^j and a record r, denote by $+$ the fact that r is actually positive, and by $-$ the opposite situation. Moreover, denote by $\mathit{class} + (\alpha_i^j)$ the fact that r is classified as positive by α_i^j, i.e. stays on the positive side of cut-point α_i^j, and by $\mathit{class} - (\alpha_i^j)$ the fact that r is in the opposite situation. Given a generic set of records N, let A_+ be the set of the records which are $\mathit{class} + (\alpha_i^j)$, and A_- be the set of records which are $\mathit{class} - (\alpha_i^j)$. Denote instead by N^+ and N^- the (possibly unknown) real positive and negative sets. Errors occur when a negative record is classified as positive, and vice versa. False positive errors are $N_- \cap A_+$; false negative ones are $N_+ \cap A_-$. The relative *confusion matrix* is given in Table 1 below. Clearly, in the general case of k classes, each matrix would be relative to the two classes separated by α_i^j.

Since the described support set selection problem is a non-trivial decision problem, it seems reasonable to model it as a binary linear programming problem. For doing so, we need to use a criterion for evaluating the quality of each binary attribute such that the overall quality value of a set of binary attributes can be given by the sum of their individual quality values. We obtain this as follows. A basic measure of the accuracy of the positive classification obtained from α_i^j can be the probability of producing a true positive divided by the probability of producing a false positive

$$o^+(\alpha_i^j) = \frac{Pr(+ \cap \mathit{class} + (\alpha_i^j))}{Pr(- \cap \mathit{class} + (\alpha_i^j))}.$$

A similar measure can evaluate the accuracy of the negative classification obtained from α_i^j

$$o^-(\alpha_i^j) = \frac{Pr(- \cap \mathit{class} - (\alpha_i^j))}{Pr(+ \cap \mathit{class} - (\alpha_i^j))}.$$

Clearly, $o^+(\alpha_i^j) \in [0, +\infty)$ and $o^-(\alpha_i^j) \in [0, +\infty)$. The higher the value, the better positive [resp. negative] classification

TABLE 1
Confusion Matrix for α_i^j

		Predicted by α_i^j	
		$+$	$-$
Real	$+$	$N_+ \cap A_+$	$N_+ \cap A_-$
	$-$	$N_- \cap A_+$	$N_- \cap A_-$

α_i^j provides. In order to have a complete evaluation of α_i^j, we consider the product $o^+(\alpha_i^j) \times o^-(\alpha_i^j) \in [0, +\infty)$. Moreover, rather than the numerical value of such evaluation, we are interested in the relative differences among the values obtained for the different cut-points. Therefore, we can sum 1 to such product, obtaining a value in $[1, +\infty)$.

$$1 + \frac{Pr(+ \cap \textit{class} + (\alpha_i^j))}{Pr(- \cap \textit{class} + (\alpha_i^j))} \cdot \frac{Pr(- \cap \textit{class} - (\alpha_i^j))}{Pr(+ \cap \textit{class} - (\alpha_i^j))}.$$

Denote now by A the set of couples of indices (i, j) of a generic set of cut-points: $\{\alpha_i^j : (i, j) \in A\}$. The overall accuracy of a classification using the cut-points in A is related to the product of the individual evaluations:

$$\prod_{(i,j)\in A} \left[1 + \frac{Pr(+ \cap \textit{class} + (\alpha_i^j))}{Pr(- \cap \textit{class} + (\alpha_i^j))} \cdot \frac{Pr(- \cap \textit{class} - (\alpha_i^j))}{Pr(+ \cap \textit{class} - (\alpha_i^j))}\right].$$

As noted above, more than the numerical values, we are interested in producing, for each set of cut-points, values that can be compared. Therefore, we can apply a scale conversion and take the logarithm of the above value. This allows to convert it in a sum, as requested above, obtaining:

$$\sum_{(i,j)\in A} \ln\left[1 + \frac{Pr(+ \cap \textit{class} + (\alpha_i^j))}{Pr(- \cap \textit{class} + (\alpha_i^j))} \cdot \frac{Pr(- \cap \textit{class} - (\alpha_i^j))}{Pr(+ \cap \textit{class} - (\alpha_i^j))}\right].$$

In conclusion, the quality q_i^j of a single cut-point α_i^j can be evaluated as follows (so that the quality of a set of cut-points results in the sum of their individual quality values)

$$q_i^j = \ln\left[1 + \frac{Pr(+ \cap \textit{class} + (\alpha_i^j))}{Pr(- \cap \textit{class} + (\alpha_i^j))} \cdot \frac{Pr(- \cap \textit{class} - (\alpha_i^j))}{Pr(+ \cap \textit{class} - (\alpha_i^j))}\right].$$

Clearly, $q_i^j \in [0, +\infty)$. Computing the above probabilities by counting instances (and denoting by $|\cdot|$ the cardinality of a set), we have:

$$q_i^j = \ln\left[1 + \frac{\frac{|N_+ \cap A_+|}{|N^+|}}{\frac{|N_- \cap A_+|}{|N^-|}} \cdot \frac{\frac{|N_- \cap A_-|}{|N^-|}}{\frac{|N_+ \cap A_-|}{|N^+|}}\right] = \ln\left[1 + \frac{|N_+ \cap A_+|}{|N_- \cap A_+|} \cdot \frac{|N_- \cap A_-|}{|N_+ \cap A_-|}\right].$$

In the general case of k classes, the above cardinalities are those of the sets appearing in the confusion matrix for α_i^j.

However, this evaluation needs the correct classification $\{N^+, N^-\}$ of the dataset N. We obviously prefer an *a priori* quality evaluation, i.e. computable by knowing only the correct classification of the training set S. We can do this by using a non-parametric method for fields having unknown distribution, and a parametric one for fields having known distribution.

In the case of fields having unknown distribution, q_i^j is simply obtained by considering the training set S instead of the generic N, while for each cut-point α_i^j sets A_+ and A_- are clearly known (they respectively are the sets or records that are $\textit{class} + (\alpha_i^j)$ and $\textit{class} - (\alpha_i^j)$). Now, the quality of each attribute a_i^j over a numerical field f_i is that of its corresponding cut-point α_i^j, that is the defined q_i^j.

In the case of fields where the hypothesis of a known distribution is satisfactory, their positive and negative density functions can be computed using the training set S, and the above quantities $|N_+ \cap A_+|$, etc. can be evaluated by using such density functions. In other words, we just know data from the training set S, but we may infer where other data will be, and compute how useful α_i^j would be for all of them. In particular, for any continuous-valued field f_i, we make the hypothesis of a *normal* (Gaussian) distribution. Such distribution can indeed model the majority of real-world values, as a consequence of the central limit theorem [36]. Denote now by m_{i+} the *mean value* that positive records have for f_i and by σ_{i+} their (population) *standard deviation* (defined as $\sqrt{\frac{\sum_{s\in S^+}(v_i^s - m_{i+})^2}{|S^+|}}$), denote by m_{i-} and σ_{i-} the same quantities for the negative records, and suppose w.l.o. g. that cut-point α_i^j represents a transition from $-$ to $+$. By computing the above parameters from the training set S, our evaluation of quality q_i^j becomes:

$$q_i^j = \ln\left[1 + \frac{\int_{\alpha_i^j}^{+\infty} \frac{1}{\sqrt{2\pi(\sigma_{i+})^2}}\, e^{-\frac{(t-m_{i+})^2}{2(\sigma_{i+})^2}} dt}{\int_{\alpha_i^j}^{+\infty} \frac{1}{\sqrt{2\pi(\sigma_{i-})^2}}\, e^{-\frac{(t-m_{i-})^2}{2(\sigma_{i-})^2}} dt} \cdot \frac{\int_{-\infty}^{\alpha_i^j} \frac{1}{\sqrt{2\pi(\sigma_{i-})^2}}\, e^{-\frac{(t-m_{i-})^2}{2(\sigma_{i-})^2}} dt}{\int_{-\infty}^{\alpha_i^j} \frac{1}{\sqrt{2\pi(\sigma_{i+})^2}}\, e^{-\frac{(t-m_{i+})^2}{2(\sigma_{i+})^2}} dt}\right].$$

In case of a discrete-valued field f_i, on the contrary, we make the hypothesis of *binomial* (Bernoulli) distribution. This should indeed describe many discrete real-world quantities [36]. Denote now by m_{i+} and M_{i+} the *minimum* and the *maximum* values on field i for positive records, and by m_{i-} and M_{i-} the same quantities for the negative records. Denote also by $n_{i+} = M_{i+} - m_{i+}$ the *number* of possible positive values for f_i, and by p_+ the characteristic positive *probability of success* (also called Bernoulli probability parameter, estimated as $|S^+|/n_{i+}$). Denote by $n_{i-} = M_{i-} - m_{i-}$ and by p_- the same quantities for negative records. Suppose, again, that α_i^j is a transition from $-$ to $+$. By computing the above parameters from S, our evaluation of quality q_i^j becomes now:

$$q_i^j = \ln\left[1 + \frac{\sum_{t=\alpha_i^j - m_{i+}}^{n_{i+}} \binom{n_{i+}}{t} (p_{i+})^t (1 - p_{i+})^{n_{i+}-t}}{\sum_{t=\alpha_i^j - m_{i+}}^{n_{i+}} \binom{n_{i-}}{t} (p_{i-})^t (1 - p_{i-})^{n_{i-}-t}} \cdot \frac{\sum_{t=0}^{\alpha_i^j - m_{i-} - 1} \binom{n_{i-}}{t} (p_{i-})^t (1 - p_{i-})^{n_{i-}-t}}{\sum_{t=0}^{\alpha_i^j - m_{i-} - 1} \binom{n_{i+}}{t} (p_{i+})^t (1 - p_{i+})^{n_{i+}-t}}\right].$$

Moreover, we modify the above q_i^j in order to reduce possible *overfitting* and to avoid selecting attributes in an unbalanced manner (e.g. all from the same fields). We penalize each attribute a_i^j corresponding to a cut-point α_i^j originated by a few isolated points of one class laying near many points of the opposite class. More precisely, we set two thresholds ν_1 and ν_2 and put $q_i^j := q_i^j/2$ for each a_i^j such that: *i*) a number of training records $\leq \nu_1$ lie on one side of α_i^j, and *ii*) a number of training records $\geq \nu_2$ (of the opposite class) lie on the other side of α_i^j. We use $\nu_1 = 5$ and $\nu_2 = 50$. We also penalize the binary attributes over a field f_i from which other binary attributes have already been selected. Clearly, this can be applied only during a sequential solution (see Section 4) of the support set selection problem. More precisely, each time an attribute from f_i is selected, we put $q_i^j := q_i^j/2$ for every still unselected attributes of f_i. Finally, for fields having a considerable overlapping between the two classes, cut-points cannot be generated when inverting the class, because almost every region of the field contains both classes. On the contrary, they are generated when inverting the *class predominance*, i.e., when passing from a region with positive predominance to one with negative predominance and *vice versa*. By using the fraction of negative records in the training $\frac{|S^-|}{|S|}$, a region has positive predominance when its percentage of negative records is $\leq g\frac{|S^-|}{|S|}\%$. Value g was set at 70.

4 Reformulations of the Support Set Selection Problem

When the quality value of each attribute have been computed, the exactly separating support set selection problem can be modeled as follows. We would like to minimize a weighted sum (and not only the number) of selected attributes, where the weights are the reciprocal $1/q_i^j$ of the quality q_i^j, while selecting at least an attribute for each of the above defined sets $I(r_b^+, r_b^-)$. Note that $1/q_i^j$ can be viewed as a measure of the *uselessness* of a_i^j. By using the binary variables x_i^j already introduced in Section 2, the following *weighted* set covering problem should be solved, using the non-negative weights $1/q_i^j$

$$\begin{cases} \min\limits_x \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n_i} \frac{1}{q_i^j}\ x_i^j \\ \text{s.t.} \sum\limits_{(i,j)\in I(r_b^+, r_b^-)} x_i^j \geq 1 \quad \forall I(s_b^+, s_b^-),\ s^+ \in S^+,\ s^- \in S^- \\ x_i^j \in \{0,1\}. \end{cases} \tag{2}$$

This formulation takes now into account the individual qualities of the attributes. One may observe that this would discard attributes that have a poor isolated effect but may have important effect when combined with other attributes during the pattern generation step. However, a selection is necessary for the computational viability of the entire procedure, and the proposed approach aims at discarding the attributes that appear more suitable to be discarded.

Moreover, such weighted set covering formulation (2) has strong computational advantages on the non-weighted one (1). Although still NP-hard [27], solution algorithms become considerably faster when the model variables receive different weight coefficients in the objective function. Depending on the size of the model and on available computational time, such weighted set covering problem may be either solved to optimality or by searching for an approximate solution. In the former case, it is guaranteed that the pattern generation step is performed by using a set of attributes U which is a minimal set for which no positive and negative records have the same binary encoding. In the latter case, if the approximate solution is feasible but non-optimal, it is not guaranteed that U is minimal, i.e. it may exist also a proper subset $U' \subset U$ such that no positive and negative records have the same binary encoding. This could have the effect of increasing the computational burden of the pattern generation step, but not of worsening the classification accuracy. If, on the contrary, the approximate solution is (slightly) infeasible, U is such that (few) positive and negative records have the same binary encoding. This could have the effect of accelerating the pattern generation step, but of decreasing the classification accuracy.

In the cases when the above model still remains computationally demanding, e.g. for large datasets, or when there are very tight time requirements, e.g. real time applications, the support set selection problem can be modeled differently. We could evaluate the computational burden added to the whole classification procedure by retaining each single attribute a_i^j, and call it its *size* s_i^j. When no specific evaluations can be done, those sizes could be set all at 1. Moreover, we can establish a maximum affordable computational burden b, for instance on the basis of the time available for performing the classification, or of the available computing hardware, etc. Note that such requirement may be independent from the minimum size of an exactly separating support set: the available resources are limited, and, if they allow obtaining an exactly separating support set, the better, but this cannot be imposed. By using the same binary variables x_i^j, the support set selection problem can now be modeled as *binary knapsack* problem:

$$\begin{cases} \max\limits_x \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n_i} q_i^j\ x_i^j \\ \text{s.t.} \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n_i} s_i^j\ x_i^j \leq b \\ x_i^j \in \{0,1\}. \end{cases} \tag{3}$$

Solving the above model is again NP-hard [27], so it may in general be as hard as (2). However, in the case when all sizes s_i^j are 1, it becomes polynomially solvable by just sorting the q_i^j values and by taking the best b of them. Note that, in this case, attributes can be selected sequentially, and the weights be modified after each single attribute selection, in order to incorporate penalty techniques such as the one described in the end of previous Section. The above selections are performed independently on positive and negative attribute, so as to find the set U^+ of selected positive attributes and the set U^- of selected negative ones. In the general case of k classes, k selections problems are to be solved.

5 Pattern Generation and Use

A pattern P is a logic function of attributes a_i^j, typically a conjunction of literals, which are binary attributes $a_i^j \in U$ or negated binary attributes $\neg a_i^j$. Given a binarized record r_b, that is a set of binary values $\{b_i^j\}$, each literal of a generic pattern P receives a value, and so P itself receives a value, denoted by $P(r) \in \{0,1\}$ (see also Section 2). We say that a pattern P *covers* a record r if $P(r) = 1$, and that pattern P is *activated* by r. In the standard LAD procedure, a *positive* pattern P^+ has to cover at least one positive record r^+ but no negative ones, and a *negative* pattern P^- is defined symmetrically. This, however, can lead to improper pattern generation in the case of noisy or otherwise difficult datasets. In our procedure, patterns are built in a bottom-up fashion, as described below. For obtaining a positive pattern, we generate every possible logic conjunction grouping up to p literals, using one after another all literals obtainable from U^+. When a conjunction $\bar{P}$ verifies the following *coverage conditions*:

- $\bar{P}$ covers at least η^c positive records of S
- $\bar{P}$ covers at most η^e negative records of S,

we save $\bar{P}$ as a pattern and never repeat $\bar{P}$ as part of other conjunctions. A negative pattern is generated symmetrically. This simple generalization of the original covering condition can generate patterns being more robust, since patterns not covering any element of the opposite class may be rare in the mentioned cases. In the general case of k classes, patterns for each class are needed, and the second coverage condition counts the records belonging to all the other classes. Thresholds η^c and η^e may be tuned on the specific dataset, with η^c proportional to data density and η^e proportional to the noise contained in the data. However, reasonable values for η^c are 1 or 2, and, in general, $2\eta^e \leq \eta^c$.

In order to produce a complete classifier, each test record should be covered by at least one pattern. However, generating bottom-up patterns could leave uncovered some regions of the data space. Therefore, an additional pattern generation step is required, in a top-down fashion: patterns describing single training records covering the still uncovered regions of the data space are taken, and then simplified, by iteratively removing literals from them in all possible ways, until they satisfy other two coverage thresholds η^c_a and η^e_a. Their meaning is respectively analogous to η^c and η^e, but the requirements should in general be more relaxed.

Now, unclassified records can be classified by examining which patterns cover them. Clearly, a record activating only positive patterns should be classified as positive, and vice versa. A positive pattern is indeed a (partial) compact description of positive records. However, in most of the cases, unclassified records activate both positive and negative patterns. Some kind of "voting" criterion is needed. LAD methodology uses a weighted sum of the activated patterns, also called discriminant Δ. The weight given to pattern P_h in this sum is denoted by w_h, with $h \in H$. The discriminant must be compared to a *classification threshold* δ for classifying record r:

$$\sum_{h\in H} w_h P_h(r) = \Delta(r) > \delta \quad \Leftrightarrow \quad r \in R^+$$
$$\sum_{h\in H} w_h P_h(s) = \Delta(r) \leq \delta \quad \Leftrightarrow \quad r \in R^-.$$

Using patterns can also be seen as *Boosting* [18], [19]: learning weak classifiers (the patterns) and combining them by means of weights in order to obtain a strong classifier. Evaluating the mentioned weights, i.e. the "power" of each pattern in the classification process, can be done using different criteria. A first criterion can be based on the coverage values of each pattern, as in the original LAD [23]. For instance (squared pattern coverage): if u_h is the number of positive records covered by a positive pattern P_h, its weight is set to $w_h = u_h^2$, and symmetrically for a negative one. However, in the case of patterns covering overlapping sets of records, criteria based on coverage could be misleading.

A more ambitious criterion is assigning weights and classification threshold in order to minimize classification errors. Since the only classification errors that can be detected at this stage are those on the training set, we try to minimize them. We assume, in absence of further information, that this would produce a similar effect on the test set, being such data of the same nature of the training set. For doing so, denote by $c(r)$ the value 1 if r is a positive record, 0 otherwise (the real classification). Clearly, $c(s)$ is known for each training record $s \in S$. On the other hand, applying the learned classifier on the training set S produces a predicted classification for each $s \in S$. By comparing real and predicted classification of a training record $s \in S$, we obtain $e_s \in \{0, 1\}$, that we call classification error for the training record s

$$e_s = \begin{cases} 1 & \text{if} \quad \Delta(s^+) \leq \delta \quad \text{or} \quad \Delta(s^-) > \delta \\ 0 & \text{otherwise.} \end{cases}$$

Values e_s clearly depend on all elements of the procedure: cut-point selection, pattern generation, pattern weights, classification threshold, so they are not easily expressible. However, when knowing whether each pattern P_h, with $h \in H$, covers or not each record $s \in S$, the above e_s are simple functions of pattern weights $\{w_h\}$ and classification threshold δ. Therefore, given the set of generated pattern $\{P_h\}$, we compute the coverages $P_h(s) \in \{0, 1\}$ for any h in the set H of all patterns and $s \in S$, obtaining a $|H| \times |S|$ matrix *PS* having binary elements d_{hs}:

$$PS = [d_{hs}] \quad \text{with } d_{hs} = P_h(s).$$

The same can be done for each pattern P_h, with $h \in H$ and each test record $t \in T$, obtaining a $|H| \times |T|$ matrix *PT* having binary elements d_{ht}:

$$PT = [d_{ht}] \quad \text{with } d_{ht} = P_h(t).$$

On the other hand, for each test record $t \in T$, we only know (at this stage) the classification c_t given by the learned classifier, again a function of $\{w_h\}$ and δ

$$c_t = \begin{cases} 1 & \text{if} \quad \sum_{h\in H} w_h P_h(t) > \delta \\ 0 & \text{if} \quad \sum_{h\in H} w_h P_h(t) \leq \delta. \end{cases}$$

Moreover, we want to learn from the training set the *class distribution*, that is the fraction of positive $\frac{|S^+|}{|S|}$ (or of negative $\frac{|S^-|}{|S|}$) records contained in the training set (clearly, given one of the two, the other is also fixed). We therefore introduce a value, called *tolerance* and denoted by γ, measuring the "difference" from the class distribution of the training set and that of the test set. Hence, in our optimization model, we have a bi-objective: minimizing the number of errors on the training set and minimizing the tolerance γ. By introducing a scalarization parameter $G > 0$, our objective becomes:

$$\min_{e,c,w,\delta,\gamma} \sum_{s \in S} e_s + G\gamma.$$

A reasonable choice for G is $|S|/10$, so that the second term of the objective cannot override the first one (whose theoretical maximum is $|S|$, but with typical values between $0.01|S|$ and $0.4|S|$). We now describe the constraints. We need to impose that the classification error e_s is 1 for each record $s \in S$ such that the classification that s would receive using $\{w_h\}$ and δ does not match its real class $c(s)$:

$$\sum_{h \in H} w_h d_{hs} - \delta \leq M(c(s) + e_s) \quad \forall s \in S \tag{4}$$

$$\sum_{h \in H} w_h d_{hs} - \delta > -M(1 - c(s) + e_s) \; \forall s \in S. \tag{5}$$

M is a positive constant greater than any possible value of the first member (see also [37]). The mathematical behavior of those constraints is the following. When $\sum_{h \in H} w_h d_{hs} - \delta > 0$, record s is predicted positive. In this case, the second member of (4) must be $\geq$ than a positive number, so it must be positive, while the second member of (5) must be $<$ than the same positive number, so it can be either 0 or negative: if $c(s) = 0$ (= the prediction is an error), e_s is forced to be 1 by the (4), while it is free for the (5); if $c(s) = 1$ (= the prediction is not an error), e_s is free for both constraints.

On the other hand, when $\sum_{h \in H} w_h d_{hs} - \delta \leq 0$, record s is predicted negative. In this case, the second member of (4) must be $\geq$ than a number ≤ 0, so it can be either 0 or positive, while the second member of (5) must be $<$ than the same number, so it must be negative: if $c(s) = 1$ (= the prediction is an error), e_s is forced to be 1 by the (5), while it is free for the (4); if $c(s) = 0$ (= the prediction is not an error), e_s is free for both constraints. Note that, when e_s is free for both constraints, the minimization of the objective will make it 0. In order to have a closed feasible region, (5) is converted into $\geq$ by introducing a small $\epsilon > 0$

$$\sum_{h \in H} w_h d_{hs} - \delta \geq -M(1 - c(s) + e_s) + \epsilon \qquad \forall s \in S.$$

In order to evaluate the class distribution that $\{w_h\}$ and δ would produce in T, we need to compute the predicted classification of its records. We therefore need constraints connecting values $\{w_h\}$ and δ to the class c_t that would be predicted for each record $t \in T$. The machinery is similar to that of the above analyzed constraints, but note that we do not use at all the real class of the test records, that must obviously remain unknown during the classification process:

$$\sum_{h \in H} w_h d_{ht} - \delta \leq M c_t \qquad \forall t \in T \tag{6}$$

$$\sum_{h \in H} w_h d_{ht} - \delta > -M(1 - c_t) \quad \forall t \in T. \tag{7}$$

Constraint (7) is converted into $\geq$ by using a small $\epsilon > 0$

$$\sum_{h \in H} w_h d_{ht} - \delta \geq -M(1 - c_t) + \epsilon \qquad \forall t \in T.$$

Finally, we need constraints imposing that $\{w_h\}$ and δ reproduce in T the class distribution of S, so $|T^+|$ should be as similar as possible to $|S^+| \cdot \frac{|T|}{|S|}$, and connecting the difference to the introduced γ:

$$\sum_{t \in T} c_t \leq \sum_{s \in S} c(s) \cdot \frac{|T|}{|S|} + |T|\gamma + \rho \tag{8}$$

$$\sum_{t \in T} c_t \geq \sum_{s \in S} c(s) \cdot \frac{|T|}{|S|} - |T|\gamma - \rho. \tag{9}$$

Note that, when we need to classify just one or a few records, obtaining the same class distribution of S could be impossible. For example, if we need to classify two records, and the fraction of positive $\frac{|S^+|}{|S|}$ is 0.2, targeting at that class distribution is clearly useless. Hence, (8-9) should have no effect when T is very small. This is obtained by using value ρ, that, when set for instance at 3, relaxes constraints (8-9) of 3 units. For large $|T|$ this relaxation is negligible (so we do not consider it in the tests of Section 6), while for small $|T|$ the problem gradually reduces to minimizing only the classification error on S.

The overall mixed integer linear model for finding optimal pattern weights w_h and classification threshold δ is now the following:

$$\begin{cases} \min\limits_{e,c,w,\delta,\gamma} \sum\limits_{s \in S} e_s + G\gamma & \\ \sum\limits_{h \in H} w_h d_{hs} - \delta \leq M(c(s) + e_s) & \forall s \in S \\ \sum\limits_{h \in H} w_h d_{hs} - \delta \geq -M(1 - c(s) + e_s) + \epsilon & \forall s \in S \\ \sum\limits_{h \in H} w_h d_{ht} - \delta \leq M c_t & \forall t \in T \\ \sum\limits_{h \in H} w_h d_{ht} - \delta \geq -M(1 - c_t) + \epsilon & \forall t \in T \\ \sum\limits_{t \in T} c_t \leq \sum_{s \in S} c(s) \cdot \frac{|T|}{|S|} + |T|\gamma + \rho & \\ \sum\limits_{t \in T} c_t \geq \sum_{s \in S} c(s) \cdot \frac{|T|}{|S|} - |T|\gamma - \rho & \\ -W \leq w_h \leq W & \forall h \in H \\ e_s \in \{0, 1\} & \forall s \in S \\ c_t \in \{0, 1\} & \forall t \in T \\ w_h \in \mathbb{R} & \forall h \in H \\ \delta \in \mathbb{R} & \\ \gamma \in \mathbb{R}_+. & \end{cases} \tag{10}$$

Weights are bounded by a value W, in order to avoid giving excessive importance to any single pattern, since that could cause overfitting. We briefly remark that, in the case of k classes, there is a set H^c of patterns for each c-th class; they still need weights w_h and threshold δ such that r can be assigned to class c when $\sum_{h \in H^c} w_h P_h(r) > \delta$, and this could still be obtained with an extension of model (10). Clearly, the final class assigned to r would in that case be the one having the largest value of the above sum.

6 Implementation and Computational Results

Tests are carried out on an Intel Pentium 4 PC with 3 GHz processor and 3.24 Gb RAM. The proposed procedure has been implemented in C++ using MS Visual Studio 2008. The quality values q_i^j are numerically approximated by using C functions described in [38]. The support set selection problem, when modeled as knapsack (3) with all $s_i^j = 1$, is solved by simply ordering by quality values the binary attributes. When modeled differently, as in (1) or (2), is solved by means of IBM Cplex [39], a state-of-the-art implementation of branch-and-cut (e.g. [27], [28]). The same solver is used to solve problem (10), possibly relaxing numerical precision. Data sets used in the experiments are "Ionosphere", "Spambase", "Pima Indians Diabetes", "Statlog Heart", "Mushroom", "Adult", "Madelon" and "MiniBooNE", publicly available from the UCI Repository of machine learning problems [32]. They were chosen in order to have a test bed containing different types of datasets (with many or few records, many or few fields, easy or difficult, etc.), so as to analyze the classifiers in different conditions.

Ionosphere has 351 instances, each with 34 fields: 32 are real-valued and 2 are binary. Real-valued fields were considered with normal distribution, one binary field was considered binomial, the other is always 0. They are data collected by a radar system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. Good radar returns are those showing evidence of some type of structure in the ionosphere.

Spambase has 4,601 instances, each with 57 fields: 55 are real-valued and 2 are integer; however they are the frequencies of particular words or characters in an email, so all 57 were considered having normal distribution. Records of this dataset correspond to received emails, and the class denotes whether the e-mail was considered spam or not.

Pima Indians Diabetes has 768 instances, each with eight fields: two are real-valued and six are integer. However, since three integer fields have a number of possible values high enough, five were considered normal and three were considered binomial. Fields are medical informations about females patients of Pima Indian heritage living near Phoenix, Arizona, the class is whether the patient shows signs of diabetes.

Statlog Heart has 270 instances, each with 13 fields: seven are real-valued and six are categorical or binary. The first seven were considered having normal distribution. The last six could not be considered having binomial distribution, so they were treated as those with normal distribution but generating cut-points when inverting the class predominance due to the few number of possible values (see Section 3). Fields are several medical informations about patients, the class is whether the patient has or not a heart disease.

Mushroom has 8,124 instances, each with 22 fields, all categorical with very few possible values (no more than 12, some just 2), so they were treated as those with binomial distribution but generating cut-points when inverting the class predominance (see Section 3). The records describe mushrooms in terms of physical characteristics, from the Audobon Society Field Guide, and the classification is poisonous or edible.

Adult has 48,842 instances, each with 14 fields: six are real-valued and were considered having normal distribution, the other eight are categorical and were treated as fields with unknown distribution. They are a set of reasonably clean person records extracted from the 1994 US Census database. The class is whether that person earns more than 50,000 USD per year or not.

Madelon has 4,400 instances, each with 500 fields, and is a difficult artificial dataset. All fields are numerical and were considered with normal distribution. It contains data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled 0 or 1. Fifteen linear combinations of those fields were added to form a set of 20 (redundant) informative fields. Moreover, 480 distractor fields were added, having no predictive power, and the order of fields and records was randomized.

MiniBooNE is a very large dataset with 130,065 instances, each with 50 fields. All fields are numerical and were considered with normal distribution. The data are obtained from the MiniBooNE experiment to distinguish electron neutrinos (signal) from muon neutrinos (background). These data-sets have been classified using the following procedures:

- The proposed one, called Statistical and Logical Analysis of Data (SLAD), that determines the binarization by solving the knapsack version (3) of the Support set selection problem, then generates patterns, determines pattern weights and classification threshold by solving (10), and classifies by comparing discriminant and threshold.
- The standard Logical Analysis of Data procedure, obtained from the former by not assigning values to binary attributes and solving an unweighted set covering problem (1) for the Support set selection, and using pattern weights w_h based on squared pattern coverage (see Section 5) and classification threshold $\delta = 0$.
- A simplified version of SLAD, called Reduced Logical Analysis of Data (RLAD), solving the knapsack version (3) of the Support set selection problem and simply using the binary attributes to perform the classification. In other words, each pattern is made of only one literal, and it determines pattern weights and classification threshold by solving (10).
- The publicly available LIBSVM 3.17 (Library for support vector machines [33]), a very good C++ implementation of the support vector machines methodology [5], [9], developed by Chih-Chung Chang and Chih-Jen Lin, possibly working on dataset previously scaled to a restricted range by means of svm-scale [33] (a preprocessing for improving accuracy).
- The publicly available UniverSVM 1.22 (support vector machines with large scale transduction[34], [40]), an updated C++ implementation of Transductive support vector machines [10], developed by Fabian Sinz and Matteo Roffilli.

TABLE 2
Ionosphere (351 Records, 34 Fields)

Algorithm	Training 5% (18/351)			
	Accur.	NMI	Time	Parameters
LAD	76.58%	0.21	0.60	std, $\eta^c = 1, \eta^e = 0$
RLAD	85.10%	0.40	0.04	$b = 35$
SLAD	85.14%	0.40	0.04	$b = 10, p = 3$ $G = 1.8, W = 10^4$
LIBSVM	82.66%	0.36	0.58	-s 0 -t 2 -g 0.25 -c 1.6818 -e 0.001
UniverSVM	80.90%	0.35	0.70	-s 0 -t 2 -g 0.25 -c 1.6818 -e 0.001
LabelProp	79.80%	0.27	0.92	gamma 1.5 m_iter 30
Algorithm	Training 10% (36/351)			
	Accur.	NMI	Time	Parameters
LAD	71.13 %	0.12	0.75	std, $\eta^c = 1, \eta^e = 0$
RLAD	89.10 %	0.50	0.10	$b = 46$
SLAD	89.49 %	0.52	0.18	$b = 11, p = 3$ $G = 3.6, W = 10^4$
LIBSVM	88.48 %	0.52	0.63	-s 0 -t 2 -g 0.1486 -c 1 -e 0.001
UniverSVM	87.91%	0.51	0.76	-s 0 -t 2 -g 0.1486 -c 1 -e 0.001
LabelProp	84.20%	0.36	1.04	gamma 0.8 m_iter 10

TABLE 3
Spambase (4601 Records, 57 Fields)

Algorithm	Training 5% (230/4,601)			
	Accur.	NMI	Time	Parameters
LAD	83.10%	0.39	2.14	std, $\eta^c = 1, \eta^e = 0$
RLAD	61.90%	0.08	0.18	$b = 240$
SLAD	90.76%	0.54	0.92	$b = 100, p = 3$ $G = 23, W = 10^4$
LIBSVM	81.83%	0.32	1.05	-s 0 -t 2 -g 0.5 -c 8.0 -e 0.001
UniverSVM	87.06%	0.42	6.90	-s 0 -t 2 -g 0.5 -c 8.0 -e 0.001
LabelProp	70.78%	0.09	1.64	gamma 0.01 m_iter 10
Algorithm	Training 10% (460/4,601)			
	Accur.	NMI	Time	Parameters
LAD	84.32 %	0.39	2.21	std, $\eta^c = 1, \eta^e = 0$
RLAD	62.12 %	0.09	0.50	$b = 370$
SLAD	91.15 %	0.56	1.61	$b = 200, p = 3$ $G = 46, W = 10^4$
LIBSVM	85.80 %	0.41	1.36	-s 0 -t 2 -g 0.5 -c 8.0 -e 0.001
UniverSVM	88.20%	0.47	7.38	-s 0 -t 2 -g 0.5 -c 8.0 -e 0.001
LabelProp	73.10%	0.12	2.09	gamma 0.05 m_iter 10

- The publicly available Label Propagation procedure [14], [15] implemented in Python within the very good Machine Learning package scikit-learn [35], developed by Fabian Pedregosa et al., currently included in Scientific Python distributions.

A small number of records (5 and 10 percent of each data-set, except for MiniBooNE, that is so large that we used 2 and 5 percent) were randomly extracted from each data-set, and used as training set. After this, the rest of the data-set was randomly split in two equal parts, which constituted validation set and test set. For each dataset and training percentage, the training set extractions were performed 10 times, so that the same number of validation and test sets were obtained. Tests are conducted on a *best-against-best* basis: we selected, for each dataset and training percentage, the parameters of each classifier that give the best classification accuracy on the validation sets using cross validation. In particular, the choice for SVM methods (LIBSVM and UniverSVM) used the grid search described in [41], and the choice for LP was done in a similar exhaustive fashion. For SLAD and RLAD, some of the parameters are obtained by solving model (10). For the rest, patterns are generated by grouping up to p literals, with $p \in \{2, 3, 4, 5\}$. Literals are obtained from a support set of at most b elements, using $\eta^c = 2$, $\eta^e = 1$, $\eta^c_a = 1$, and $\eta^e_a = 0$. Parameter b controls the computational burden of the procedure: initially set at one tenth of the number of all possible binary attributes, it is then progressively decreased if the computational requirements of the procedure were excessive. Parametr W is just a very large value to avoid that the weight of a pattern could approach infinity. Parameters not indicated in the tables were fixed at the values mentioned when they have been introduced.

In Tables 2, 3, 4, 5, 6, 7, 8, and 9 we report the accuracy (Accur.) on the test sets, computed as the percentage of correct predictions w.r.t. the total number of predictions. All results are averaged on the 10 trials. We also report the normalized mutual information (NMI), that is an information-based measure that has been recently proposed for evaluating the correlation between the prediction of a classifier and the real classification [42]. Note that, while the maximum value for NMI is 1, this measure decreases very rapidly when classification errors are present: for example, a classifier producing 80 true positive, 10 true negative and 10 false negative has 90% accuracy but only 0.57 NMI. We then report computational times in seconds required by the

TABLE 4
Pima Indians Diabetes (768 Rec., 8 Fields)

Algorithm	Training 5% (38/768)			
	Accur.	NMI	Time	Parameters
LAD	68.48%	0.06	0.88	std, $\eta^c = 1, \eta^e = 0$
RLAD	66.10%	0.06	0.03	$b = 60$
SLAD	72.87%	0.12	0.40	$b = 60, p = 3$ $G = 3.8, W = 10^4$
LIBSVM	70.41%	0.09	0.80	-s 0 -t 2 -g 0.000173 -c 4096 -e 0.001
UniverSVM	70.39%	0.10	0.85	-s 0 -t 2 -g 0.000173 -c 4096 -e 0.001
LabelProp	65.20%	0.01	1.00	gamma 8.5, m_iter 5
Algorithm	Training 10% (76/768)			
	Accur.	NMI	Time	Parameters
LAD	72.80 %	0.10	1.15	std, $\eta^c = 1, \eta^e = 0$
RLAD	65.90 %	0.05	0.06	$b = 64$
SLAD	75.54 %	0.15	0.62	$b = 64, p = 3$ $G = 7.6, W = 10^4$
LIBSVM	72.74 %	0.15	2.84	-s 0 -t 2 -g 0.25 -c 11.31 -e 0.001
UniverSVM	72.40%	0.13	1.90	-s 0 -t 2 -g 0.25 -c 11.31 -e 0.001
LabelProp	67.25%	0.06	1.28	gamma 0.25, m_iter 10

TABLE 5
Statlog Heart (270 Records, 13 Fields)

Algorithm	Training 5% (14/270)			
	Accur.	NMI	Time	Parameters
LAD	63.40%	0.06	0.78	std, $\eta^c = 1, \eta^e = 0$
RLAD	64.25%	0.08	0.02	$b = 54$
SLAD	78.21%	0.25	0.02	$b = 14, p = 4$ $G = 1.4, W = 10^4$
LIBSVM	77.58%	0.25	0.20	-s 0 -t 2 -g 0.00035 -c 512 -e 0.001
UniverSVM	77.43%	0.25	0.28	-s 0 -t 2 -g 0.00035 -c 512 -e 0.001
LabelProp	60.53%	0.04	1.07	gamma 0.15, m_iter 100
Algorithm	Training 10% (27/270)			
	Accur.	NMI	Time	Parameters
LAD	67.80 %	0.11	0.78	std, $\eta^c = 1, \eta^e = 0$
RLAD	73.31 %	0.18	0.04	$b = 64$
SLAD	81.11 %	0.31	0.04	$b = 22, p = 4$ $G = 2.7, W = 10^4$
LIBSVM	77.52%	0.25	0.34	-s 0 -t 2 -g 0.125 -c 512 -e 0.001
UniverSVM	77.13%	0.24	0.41	-s 0 -t 2 -g 0.125 -c 512 -e 0.001
LabelProp	60.88%	0.04	1.32	gamma 0.15, m_iter 100

TABLE 7
Adult (48,842 Records, 14 Fields)

Algorithm	Training 5% (2,442/48,842)			
	Accur.	NMI	Time	Parameters
LAD	76.32%	0.03	35.10	std, $\eta^c = 1, \eta^e = 0$
RLAD	77.40%	0.04	0.78	$b = 150$
SLAD	82.07%	0.17	7.10	$b = 80, p=3$ $G=244, W = 10^5$
LIBSVM	83.56%	0.20	10.02	-s 0 -t 2 -g 0.125 -c 8.0 -e 0.001
UniverSVM	82.90%	0.20	610.80	-s 0 -t 2 -g 0.125 -c 8.0 -e 0.001
LabelProp	76.14%	0.01	24.50	gamma 9.0, m_iter 10
Algorithm	Training 10% (4,884/48,842)			
	Accur.	NMI	Time	Parameters
LAD	75.72 %	0.03	75.30	std, $\eta^c = 1, \eta^e = 0$
RLAD	74.84 %	0.02	0.95	$b = 200$
SLAD	83.68 %	0.19	8.64	$b=100, p=3$ $G=488, W=10^5$
LIBSVM	84.28%	0.22	12.58	-s 0 -t 2 -g 0.125 -c 8.0 -e 0.001
UniverSVM	83.90%	0.22	895.50	-s 0 -t 2 -g 0.125 -c 8.0 -e 0.001
LabelProp	76.12%	0.01	26.02	gamma 9.0, m_iter 10

whole classification procedure running with the parameters used to obtain those best results, and the values of such parameters (with 'std' meaning standard values).

As a general outcome, our experiments show that the effort invested in evaluating the quality of the different binary attributes returns a superior classification accuracy with respect to the standard LAD procedure. In the totality of the analyzed cases, indeed, SLAD is more accurate than LAD. That additional effort clearly required an additional computational time, but that was almost negligible, and moreover, in the solution of the support set selection problem, weighted set covering problems can generally be solved in times which are much shorter than those needed for the corresponding non-weighted ones, so the balance is in favor of performing the above quality evaluation. Furthermore, the solution of the support set selection problem as binary knapsack (3) using the above quality evaluation and all $s_i^j = 1$ is even faster and produces a very good classification accuracy. The computational demand of SLAD is controlled by parameters b and p, so the scalability appears

TABLE 6
Mushroom (8,124 Records, 22 Fields)

Algorithm	Training 5% (406/8,124)			
	Accur.	NMI	Time	Parameters
LAD	95.77%	0.79	2.67	std, $\eta^c = 1, \eta^e = 0$
RLAD	87.63%	0.48	0.32	$b = 176$
SLAD	97.79%	0.83	0.83	$b = 20, p = 4$ $G = 40, W = 10^4$
LIBSVM	92.71%	0.57	1.12	-s 0 -t 2 -g 0.03125 -c 8.0 -e 0.001
UniverSVM	97.90%	0.86	5.62	-s 0 -t 2 -g 0.03125 -c 8.0 -e 0.001
LabelProp	98.60%	0.88	2.15	gamma 0.5, m_iter 5
Algorithm	Training 10% (812/8,124)			
	Accur.	NMI	Time	Parameters
LAD	96.11 %	0.79	3.85	std, $\eta^c = 1, \eta^e = 0$
RLAD	85.08 %	0.39	0.45	$b = 200$
SLAD	99.72 %	0.98	0.83	$b = 25, p = 4$ $G = 81, W = 10^4$
LIBSVM	99.60%	0.96	1.25	-s 0 -t 2 -g 0.03125 -c 8.0 -e 0.001
UniverSVM	99.66%	0.97	7.43	-s 0 -t 2 -g 0.03125 -c 8.0 -e 0.001
LabelProp	98.85%	0.92	3.24	gamma 0.5, m_iter 5

TABLE 8
Madelon (4,400 Records, 500 Fields)

Algorithm	Training 5% (220/4,400)			
	Accur.	NMI	Time	Parameters
LAD	56.05%	0.02	2.36	std, $\eta^c = 1, \eta^e = 0$
RLAD	60.36%	0.03	1.13	$b = 24$
SLAD	60.96%	0.04	1.28	$b = 24, p=5$ $G=22, W = 10^4$
LIBSVM	57.72%	0.02	1.64	-s 0 -t 2 -g 0.10511 -c 1.0 -e 0.001
UniverSVM	57.11%	0.03	24.33	-s 0 -t 2 -g 0.10511 -c 1.0 -e 0.001
LabelProp	54.14%	0.01	1.38	gamma 0.5, m_iter 10
Algorithm	Training 10% (440/4,400)			
	Accur.	NMI	Time	Parameters
LAD	57.55 %	0.02	5.97	std, $\eta^c = 1, \eta^e = 0$
RLAD	62.02 %	0.04	1.34	$b = 36$
SLAD	62.20 %	0.04	1.98	$b=36, p=5$ $G=44, W=10^4$
LIBSVM	58.55%	0.03	2.44	-s 0 -t 2 -g 0.10511 -c 1.0 -e 0.001
UniverSVM	60.04%	0.04	26.60	-s 0 -t 2 -g 0.10511 -c 1.0 -e 0.001
LabelProp	54.90%	0.01	2.32	gamma 0.5, m_iter 10

TABLE 9
MiniBooNE (130,065 Records, 50 Fields)

Algorithm	Training 2% (2,601/130,065)			
	Accur.	NMI	Time	Parameters
LAD	76.84%	0.16	175.5	std, $\eta^c = 1, \eta^e = 0$
RLAD	72.07%	0.10	28.2	$b = 890$
SLAD	83.75%	0.28	120.0	$b = 210, p=3$ $G=260, W = 10^4$
LIBSVM	82.19%	0.22	118.0	-s 0 -t 2 -g 0.125 -c 1.0 -e 0.001
UniverSVM	82.88%	0.23	6,760.0	-s 0 -t 2 -g 0.125 -c 1.0 -e 0.001
LabelProp	-	-	-	out of memory
Algorithm	**Training 5% (6,503/130,065)**			
	Accur.	NMI	Time	Parameters
LAD	76.17 %	0.15	212.0	std, $\eta^c = 1, \eta^e = 0$
RLAD	76.65 %	0.14	55.8	$b = 4200$
SLAD	82.24 %	0.24	194.5	$b=420, p=3$ $G=650, W=10^4$
LIBSVM	83.55%	0.28	162.0	-s 0 -t 2 -g 0.125 -c 1.0 -e 0.001
UniverSVM	83.40%	0.28	8,900.0	-s 0 -t 2 -g 0.125 -c 1.0 -e 0.001
LabelProp	-	-	-	out of memory

satisfactory, as shown by the times required for the large datasets. Comparison with LIBSVM, which is currently deemed to be among the best classifiers, show the effectiveness of the proposed approach. Indeed, SLAD obtains a classification accuracy that is always comparable and sometimes better than that of LIBSVM. Comparison with UniverSVM shows that learning the parameters by means of model (10) can be a competitive option. Indeed, in the cases when TSVM demonstrate an advantage over classical SVM, the accuracy of SLAD is comparable to that of TSVM, while computational times of SLAD scale better on the large datasets. Comparison with LabelProp, finally, gives a similar result: when LP exhibits the advantages of a semi-supervised approach, the accuracy of SLAD is always comparable. Moreover, the simple classifier RLAD is considerable faster and scales better than the others, because no time consuming problems must be solved in the different steps of this procedure. Its accuracy is clearly inferior, but it is sometimes comparable. Hence, it can be used when a timely (or even a real-time) classification is needed, or when very large datasets should be treated.

7 Conclusion

To classify in short times with a good degree of accuracy on the basis of small training sets is required in a variety of practical applications. Unfortunately, obtaining these three desirable features together can be very difficult. We consider here the framework of the logical analysis of data, and propose several enhancements to this methodology based on statistical considerations on the data. In particular, we use more information extracted from the training set to guide the support set selection step, and propose two reformulations of such a problem having several advantages. Moreover, we consider the problem of selecting the best parameters for the procedure (pattern weights and classification threshold), and formulate it as an optimization problem. The proposed methodology, called Statistical and Logical Analysis of Data, is tested on a test bed of publicly available datasets from the UCI repository, and compared to: standard LAD methodology; support vector machines (also in the form of Transductive SVM); label propagation technique. Experiments show that the presented enhancements are able to sensibly increase the classification accuracy and reduce computation times with respect to standard LAD methodology. The comparison with the other classifiers proves that SLAD has very good accuracy and timing results using very small training sets, and that it scales well on the large datasets. Moreover, a simplified version of SLAD, called Reduced Logical Analysis of Data, is proposed. All steps of this latter procedure can be solved in very short times, allowing a sensible speed-up of the whole classification procedure.

Appendix A
Main Notation

R record scheme
r record instance, or simply record
r^+, r^- positive or negative record
R_b binarized record scheme
r_b binarized record instance, or simply binar. record
f_i fields of the record scheme R, with $i = 1, \dots, m$
v_i values of the record instance r, with $i = 1, \dots, m$
a_i^j binary attributes of the binarized record scheme R_b, with $i = 1, \dots, m$ and $j = 1, \dots, n_i$
b_i^j binary values of the binarized record r_b, with $i = 1, \dots, m$ and $j = 1, \dots, n_i$
α_i^j cut-point over field f_i, $i = 1, \dots, m$, $j = 1, \dots, n_i$
${}_{class}+(\alpha_i^j)$, ${}_{class}-(\alpha_i^j)$ classified as positive or negative by α_i^j: stays on the positive (or negative) side of α_i^j
q_i^j quality of cut-point α_i^j, $i = 1, \dots, m$, $j = 1, \dots, n_i$
$n = \sum_i n_i$ total number of cut-points
U support set
U^+, U^- positive or negative part of the support set
S training set
S^+, S^- set of positive or negative records of S
s^+, s^- positive or negative training record
C set of predefined classes
$c(s)$ real class of record $s \in S$
c_s predicted class of record $s \in S$
T test set
T^+, T^- set of positive or negative records of T
t^+, t^- positive or negative test record
P_h pattern, with $h \in H$
H set of pattern indices
w_h weight of pattern P_h
P^+, P^- positive or negative pattern
H^+, H^- set of positive or negative patterns
$P_h(r)$ value of pattern P_h on record r
η^c minimum correct coverage for patterns
η^e maximum erroneous coverage for patterns
$\Delta(r)$ discriminant $\sum_{h \in H} w_h P_h(r)$ for record r
δ classification threshold
$d_{hs} = P_h(s)$ value of pattern P_h on record $s \in S$

PS matrix $[d_{hs}]$ with $h \in H$ and $s \in S$
$d_{ht} = P_h(t)$ value of pattern P_h on record $t \in T$
PT matrix $[d_{ht}]$ with $h \in H$ and $t \in T$
γ class distribution tolerance

ACKNOWLEDGMENTS

The authors are grateful to Prof. G. Koch for useful discussions on probabilistic issues, and to Dr. V. Minetti for her important contribution to the implementation work.

Renato Bruni is the corresponding author.

REFERENCES

[1] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[2] T. M. Mitchell, *Machine Learning*. Singapore: McGraw-Hill, 1997.
[3] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.
[4] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. London, U.K.: MIT Press, 2001.
[5] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer-Verlag, 2002.
[6] W. Klosgen, J.M. Zytkow (eds), *Handbook of Data Mining and Knowledge Discovery*. London, U.K.: Oxford Univ. Press, 2002.
[7] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. 30, no. 4, pp. 451–462, Nov. 2000.
[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273-297, 1995.
[9] C.-C. Chang and C.-J. Lin, "Training ν-support vector classifiers: Theory and algorithms," *Neural Comput.*, vol. 13, no. 9, pp. 2119–2147, 2001.
[10] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY, USA: Springer, 1995.
[11] X. Zhu, "Semi-supervised learning literature survey," Comput. Sci. TR 1530, Univ. Wisconsin-Madison, Madison, WI, USA, 2008.
[12] F. Nie, D. Xu, I.W.H. Tsang, C. Zhang, "Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction," *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1921–1932, Jul. 2010.
[13] S. Xiang, F. Nie, C. Zhang, "Semi-supervised classification via local spline regression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 11, pp. 2039–2053, Nov. 2010.
[14] X. Zhu, Z. Ghahramani, J. Lafferty, "Semisupervised learning using gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 912–919.
[15] Y. Bengio, O. Delalleau, N. Le Roux, "Label propagation and quadratic criterion," in *SemiSupervised Learning*, O. Chapelle, B. Schlkopf, A. Zien, Eds. Cambridge, MA,USA: MIT Press, 2006, pp. 193–216.
[16] D. H. Wolpert, "The lack of A priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, pp. 1341-1390, 1996.
[17] N. Jankowski, W. Duch, K Grabczewski, Eds,. *Meta-Learning in Computational Intelligence*. Berlin, Germany: Springer-Verlag, 2011.
[18] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197-227, 1990.
[19] Y. Freund, "Boosting a weak learning algorithm by majority," *Inf. Comput.*, vol. 121, no. 2, pp. 256–285, 1995.
[20] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123-140, 1996.
[21] Y. Crama, P. L. Hammer, and T. Ibaraki, "Cause-effect relationships and partially defined boolean functions," *Ann. Oper. Res.*, vol. 16, pp. 299–326, 1988.
[22] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, "Logical analysis of numerical data," *Math. Program.*, vol. 79, pp. 163–190, 1997.
[23] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik, "An implementation of logical analysis of data," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 2, pp. 292–306, Mar./Apr. 2000.
[24] E. Boros, Y. Crama, P.L. Hammer, T. Ibaraki, A. Kogan, and K. Makino, "Logical analysis of data: Classification with justification," *Ann. Oper. Res.*, vol. 188, pp. 33–61, 2011.
[25] Y. Crama and P. L. Hammer, *Boolean Functions: Theory, Algorithms, and Applications*. New York, NY, USA: Cambridge Univ. Press, 2011.
[26] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," *Artif. Intell.*, vol. 13, pp. 1022–1027, 1993.
[27] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley, 1988.
[28] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: Wiley, 1986.
[29] E. Boros, T. Horiyama, T. Ibaraki, K. Makino, and M. Yagiura, "Finding essential attributes from binary data," *Ann. Math. Artif. Intell.*, vol. 39 no. 3, pp. 223–257, 2003.
[30] H. Almuallim and T. G. Dietterich, "Learning boolean concepts in the presence of many irrelevant features," *Artif. Intell.*, vol. 69, no. 1, pp. 279–306, 1994.
[31] P. L. Hammer, A. Kogan, B. Simeone, and S. Szedmak, "Pareto-optimal patterns in logical analysis of data," *Discrete Appl. Math.*, vol. 144, no. 1/2, pp. 79–102, 2004.
[32] A. Frank, A. Asuncion. (2010). UCI Machine Learning Repository [Online]. Available: http://archive.ics.uci.edu/ml, Irvine, CA, USA: Univ. California, School of Inf. Comput. Sci.
[33] C.-C. Chang and C.-J. Lin, "LIBSVM : A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.
[34] R. Collobert, Fabian Sinz, J. Weston, and L. Bottou, "Large scale transductive SVMs," *J. Mach. Learn. Res.*, vol. 7, pp. 1687-1712, 2006.
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, 2011.
[36] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*, Wiley series in Probability and Statistics, 4th ed. New York, NY, USA: Wiley, 2010.
[37] H.P. Williams, *Model Building in Mathematical Programming*. Chichester, U.K.: Wiley, 1993.
[38] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[39] IBM: *Ilog Cplex 12.1 Reference Manual*, Int. Bus. Mach. Corporation, 2009.
[40] F. Sinz and M. Roffilli. (2012). *UniverSVM*, in Machine Learning Open Source Software (MLOSS) repository, project 19 [Online]. Avaialble: http://mloss.org/software/view/19/
[41] C.-W. Hsu, C.-C. Chang, C.-J. Lin, "A practical guide to support vector classification," Dept. Comput. Sci., Nat. Taiwan Univ., Taipei, Taiwan, 2010, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
[42] B.-G. Hu and Y. Wang, "Evaluation criteria based on mutual information for classifications including rejected class," *Acta Automatica Sinica*, vol. 34, no. 11, pp. 1396–1403, 2008.

Renato Bruni received the PhD degree in operations research in 2001 and the MS degree in computer engineering in 1996, both from the University of Rome "Sapienza." He was a researcher at the University of Perugia, and he is currently an assistant professor in the Combinatorial Optimization Group of the the University of Rome "Sapienza." He authored more than 60 research papers and one patent. His research interests are in combinatorial optimization, data analysis, information reconstruction and knowledge discovery, also in collaboration with the Italian National Institute of Statistics "Istat."

Gianpiero Bianchi received the PhD degree in operations research in 2013 and the MS degree in computer engineering in 2001, both from the University of Rome "Sapienza." He has been working since 2002 on the technical issues related to census data treatment for the Department of Censuses and Administrative and Statistical Registers of the Italian National Institute of Statistics "Istat", and he is currently with the Department for Integration, Quality, Research and Production Networks Development of the same Institute. His research interests are in data mining, database systems management, computer systems design, data editing, and imputation.