
Direct Mining of Rules from Data with Missing Values

Vladimir Gorodetsky, Oleg Karsaev and Vladimir Samoilov

St. Petersburg Institute for Informatics and Automation, 39, 14-th Liniya,
St.Petersburg, 199178, Russia
{gor,ok,samovl}@mail.ias.spb.su

Summary. The paper presents an approach to and technique for direct mining of binary data with missing values aiming at extraction of classification rules, whose premises are represented in a conjunctive form. This approach does not assume an imputation of missing values. The idea is (1) to generate two sets of rules serving as the upper and low bounds for any other sets of rules corresponding to all arbitrary assignments of missing values, and then, (2) based on these upper and low bounds of the rules' sets, on testing procedure and on a classification criterion to select a subset of rules to be used for classification. The approach is primarily oriented to the application domains where an imputation is either cannot be theoretically justified or is impossible at all. Examples of such applications are given by domains where information used for classification is composed of asynchronous data streams of various frequencies and thus possessing different "life time", or such information is missing due to peculiarities of information collection system. Instead of missing value imputation, the proposed approach uses training dataset to cut down the potential rules set via forming its low and upper bounds with the subsequent testing the rules of the upper bound against the new dataset with missing values and selection of the most appropriate rules. The approach was applied to learning of intrusions detection in computer network based on asynchronous data streams incoming from multiple data sources. Experimental results confirm that the proposed approach to direct mining of data with missing values can yield good results.

1 Introduction

In the last three decades a large number of diverse powerful approaches, methods and techniques for data mining and Knowledge Discovery in Data bases (KDD) was developed. They are of wide use practically in any area of information technology, scientific research and industry that demonstrate ever increasing interest to extending practical application of data mining and KDD in new applications.

Unfortunately, as a rule, most of the popular and powerful data mining and KDD techniques cannot deal *directly* with real-life data due to the fact that

most data bases are incomplete, contain wrong items and noise. The available data mining and KDD software tools usually contain special means aiming at *data cleaning* and *outlier detection*. The above means help to cope with certain kind of distortions in real-life data bases, thus, making the “traditional” data mining and KDD techniques applicable. However, incompleteness of data bases, particularly, presence of missing values, remains in challenge. The reason is that, opposite to the distortion like noise and presence of outliers, the problem of data mining with missing values is of fundamental difference. An ambiguity of missing values forces KDD miner either to predict these values, thus making the task more definite or to select somehow a solution from all possible ones. In any case, the latter leads to the computationally intensive search.

In the last two decades the main effort of a researcher dealing with data mining with missing values concerned the methods of the first type, i.e. the methods oriented to determining a reasonable assignment (“imputation”) of the missing values. Significantly fewer investigations dealt with “*direct*” mining of data with missing values not presupposing the use of missing values imputation. A straightforward approach ignoring the examples with missing values is an exception. But as a rule such an approach significantly impoverish training and testing datasets, thus, making impossible a creation of powerful classification mechanisms. For instance, if percentage of missing values is 5% for each attribute and the dataset comprises 40 attributes then at average only 13% of the original sample can be used for training and testing [14]. However, in practice the percentage of missing values can be much higher, say, up to (20–30)% and over. In these cases the total number of examples without missing values can be about zero. Other existing approaches of the similar kind, i.e. approaches mainly not focused on missing values imputation, are outlined in Sect. 2 surveying the related works.

This paper proposes an approach to and technique for lattice-based direct mining of binary datasets with missing values aiming at extraction of classification rules with premises represented in a conjunctive form. The idea is to extract the sets of rules serving as the upper and low bounds for any other sets of rules corresponding to arbitrary assignments of missing values, and then, based on these upper and low bounds of rules’ sets, on the results of testing procedure and on a classification quality criterion, to select a set of rules to be used in the classification mechanism. Let us emphasize that this method does not assume the imputation of missing values.

The rest of the paper is organized as follows. Section 2 briefly surveys related works indicating the basic ideas of dealing with missing values proposed in the previous research. Section 3 briefly describes a number of newly arisen applications of high practical importance where data mining and KDD with missing values is a central subtask. A peculiarity of these applications is a limited possibility to impute missing values due to restricted size of training and testing dataset and also due to distributed and asynchronous nature of input data streams used in both learning and decision making procedures.

Section 4 presents data mining with missing values problem statement and outlines the respective algorithm of rules extraction for the case if training and testing datasets are binary. It also presents the GK2 algorithm used in this work for extraction of rules from binary data, the basic idea and main theorems constituting a basis for building upper and low bounds of the rules' sets corresponding to any arbitrary assignments of missing values. This section also demonstrates the basic algorithms by example. Section 5 describes the experiments and experimental results of computation of upper and low bounds of the rules' sets and peculiarities of this procedure. As a case study the learning of intrusion detection in computer network as applied to anomaly detection task is used. Testing procedure and use of its results to create a classification mechanism are also described in Sect. 5. In conclusion the paper results are evaluated, and a tentative plan of future works is outlined.

2 Related Works

One of the earliest works on data mining and KDD with missing values applied to decision tree mining is [20]. In it, while considering the use of C4.5 technique to induce decision trees based on data with missing values, the author emphasized three main problems that are 1) How to deal with missing values in selection of a test to partition the training dataset if different attributes contain various percentage of missing values; (2) How to treat the cases, in which values of X are unknown for the selected test based on attribute X ; and (3) How to proceed with a test on an attribute whose value is unknown if decision tree is induced. Although these questions were set within context of the particular method, C4.5, with certain variations they remain important up to now for majority of approaches to data mining and KDD dealing with missing values. In general case the above questions can be formulated as follows:

- (1) How to deal with missing values in training procedure?
- (2) How to evaluate the training quality? For instance, how to assign a truth value to a rule tested on a case when certain attributes of the rule premise cannot be assigned a value?
- (3) How to use the constructed classification mechanism (decision tree-based, rule-based, etc.) for classification of unseen data with missing values?

In most papers all these questions are reformulated into “*how to impute missing values or cut the dataset?*” at every of three aforementioned steps (1)–(3) of data mining procedure. Indeed, if this problem were solved then the data mining problem would be a “standard” one, thus implying an application of a broad spectrum of existing techniques.

In practice, a selection of any technique for missing values imputation should depend on “the missing mechanism” [14] and on the relationships between the data attributes. From statistical point of view, three classes of

missing data mechanisms can exist. In the simplest case (in the first class) the data are missing completely at random: *probability that an attribute X value is missing* does not depend on X value location in the respective attribute domain. In the second class of the missing mechanism, the fact that the value of an attribute X is missing depends on the value of some other attribute Y . This fact provides a data miner with additional information, which can be used in a certain way in order to more trustworthily impute the missing values. In the third class, a probability that value of an attribute X is missing depends on its value. Information concerning this probability can be used for missing value imputation. For example, in some cases data are missing because the value to be measured is either too large or too small, and this is why it is simply immeasurable. This fact can provide useful information for correct dealing with missing values. Additional source of a priori information that can be used for justification of missing data imputation is provided by knowledge about relationships existing between data attributes within particular classes.

The researchers propose many different approaches to and techniques for missing values imputation. In the simplest approach the records with missing values are erased from a training dataset, but this can lead to the fact that the rest of training sample could contain too small number of examples. An example was given in Sect. 1 [14].

Most of existing approaches is of statistical nature and use both knowledge about missing mechanism and correlations between attributes of each example computed based on available dataset. Imputation by regression (*linear regression* and *logistic regression*) [24] including *multiple imputation* [24] is one of such approaches to dealing with unknown data values. It can be applied to predict the value of one or several variables. These regression-based procedures use non-missing attributes. The disadvantage of this approach is that in many cases estimating by regression is as complex as learning of classification [30].

Hot Deck imputation [26] is a local approach, where the dataset is preliminary clustered on the basis of a similarity measure and then, within each cluster, the missing values are assigned the cluster most frequent values. Although this class of procedures has no rigorous mathematical justification, practitioners reported that this approach works well in many applications.

An approach called Expectation Maximization ([5, 14]), EM, is an algorithm based on estimations of the parameters of the probability distribution of an incomplete sample evaluated by maximum likelihood procedure.

The aforementioned approaches are based on the use of attributes with non-missing values. Several approaches use direct manipulation with missing data. As a rule, they are used to build decision trees ([13, 21]. The authors of the work [12] proposed to assign missing values the most probable ones estimated on the basis of training dataset. The paper [20] proposes to treat the value “unknown” as a new possible value of each attribute and to deal with it same way as with other attributes. However, the last approach works

satisfactory if the unknown value is located close to the upper or low bounds of the value domain.

The paper [30] is focused on binary classification of data with missing values. Its idea is to somehow generate for both classes the equal number of rules with premises in conjunctive form, to assess the rules on the basis of testing procedure and then to use these rules, for example, in a weighted voting mode, while only accounting for those rules, which take definite values over the unseen case to be classified.

Specific problems have to be solved when we deal with missing values in association rule mining task. Ignorance of the missing values leads either to the reduced rates of support and confidence or/and generation of inconsistent rules that cannot be distinguished from the useful ones. This problem and probable approaches to overcoming it are discussed in [22] and [23]. The idea of these two papers consists in the use of some kind of dataset preprocessing, which, given item set, allows to discover from original dataset a so-called valid dataset, which corresponds to maximal data subset not containing missing values in the given item set. One more approach to the same task is proposed in [16]. It intends to generate so-called approximate rules, \sim AR-rules. This algorithm is built on the well known Apriory algorithm and uses two main steps to deal with missing and noisy data. At the first step missing values are imputed via replacing with a probability distribution over possible values represented in terms of their frequencies in the dataset. At the second step this imputed dataset is used to mine association rules.

3 Examples of Applications

In the previous section it was indicated that one of the important aspects of data mining and KDD from databases with missing values is the nature of missingness. Indeed, the values of data can be missing due to imperfection of sensors and other components of measurement mechanism (errors, failures, e.g., transient failure, communication failure, etc.). In this case a reasonable data imputation procedure is pertinent. It is quite different if information cannot be collected due to certain reasons, like conditions of natural environment (e.g., if airborne observation equipment is used then information can be unavailable due to meteorological factors or due to masking). In this case imputation might be not relevant at all. Much more specific and complicate case, where missing values imputation is not applicable, can arise as a result of temporal and/or spatial nature of data, asynchronous and distributed mode of data collection, and discrete and distributed mode of decision making. An abstract example of such kind is as follows.

Let us consider what is often called a *situation* [6]. Generally, situation is understood as a state of a complex system constituted, for example, by a number of semi-autonomous objects ("*situation objects*", for short) having their particular goals (intents) and operating in a coordinated mode to achieve

a common goal. Let us note that “situation object” can be a “physical” object (say, group of aircrafts) or an “abstract” one (e.g., components of software in which traces of attacks against computer network are manifested). Situation can be characterized by its “state” taking value from a finite set of labels. States of situation objects and respectively state of situation on the whole are of dynamic nature, i.e. they are varying in time.

There exists a distributed sensor system observing either particular objects or definite regions, where the situation objects can be located. Observations are distributed and asynchronous, and can be performed at different discrete time instants. The observations constitute the input of a decision making system aiming at classification of the state of the situations in discrete, as a rule, irregular time instants. This task is also well known as “*situation assessment*” [6]¹. As a rule, an observation system is capable to provide the situation assessment system only with a series of “snapshots” of the overall “picture”, where a part of data can be missing. Here it is important to note that situation is specified not only in terms of *states* of particular situation objects but also in terms of a number of relationships given over them (spatial, temporal, and other, most of which is application-specific). Therefore, loss of information about an object unavoidably leads to missing of the values of attributes specifying relationships between the object in question and other ones.

According to the modern view of the architecture of the situation assessment system [7], the decision making procedure in it is organized at least at two levels. At the first level, local decisions corresponding to assessments of the particular object states are produced. At the second level these decisions are combined. Combining of decision is a conventional decision making task, whose input is formed by local decisions assessing the states of particular situation objects and relationships between them. As a rule, this input contains missing values and the size of sample presenting former experience and available for situation assessment learning is limited and does not provide statistically reliable information needed for missing values imputation.

Let us explain the above abstract consideration by example that is a detection of intrusions in computer network. This task is a typical application from the situation assessment scope containing missing values in both training and testing samples.

Currently the coordinated distributed attacks performed by a team of malefactors from distributed hosts constitute main threats for computer networks and information. “Traces” of an attack can appear in input traffic or in different data generated by a computer network assurance system. For example (see Fig. 1, and also [9]), these traces are displayed in *IP* packets of traffic, in *Tcpdump* generated via traffic preprocessing, in audit data trail, in sequences of system calls of operating system, in data resulting from

¹ Situation assessment task is a central subtask of the well known and very important modern task called “situational awareness” [6].

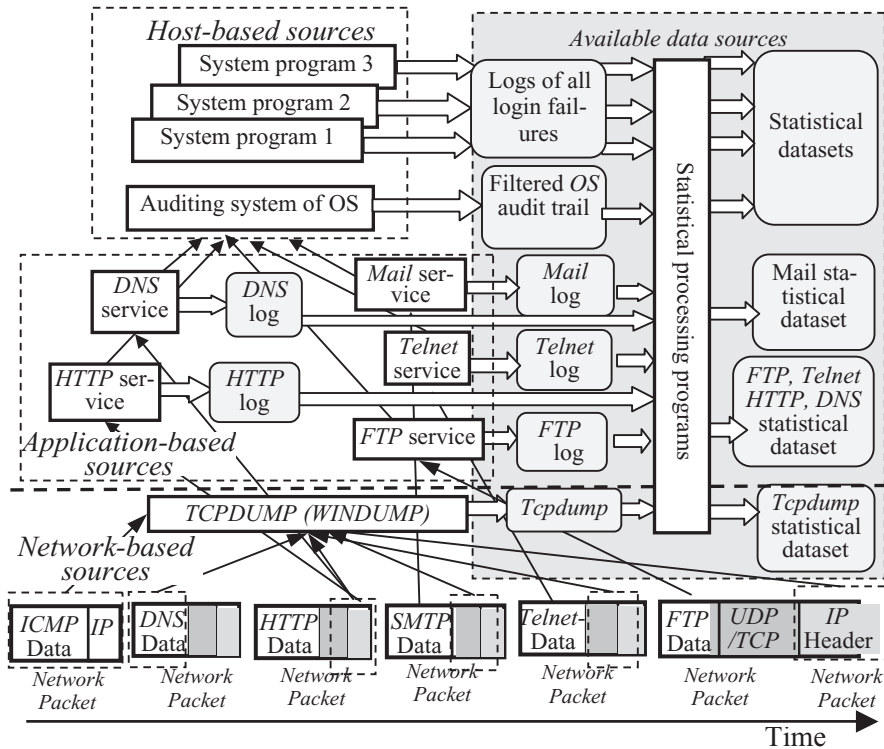


Fig. 1. Multiplicity of data sources (given in grey color) within a host which can alert intrusions (The content of the figure is taken from [9])

monitoring of application servers, queries to databases and directories, in data specifying users' profiles, etc. Such data are generated in different hosts of computer network. The timely detection of an illegitimate user's activity to assess a security status of computer network is potentially feasible only in case of information from different sources fusion. Formally, intrusion detection is a classification task, that has to detect intrusions based on *combining of particular alerts* produced by analyzers (local classifiers) dealing with particular data sources. Assessment of the security-related status of a computer network intending to detect malicious activity presents an interesting example of situation assessment tasks.

Let us describe a case study from intrusion detection scope, which is further used to validate an approach proposed for dealing with missing values in data mining and KDD procedures. The purpose of the description given below is also to explain why temporal nature of data from different sources leads to missing values in input data of an intrusion detection system (IDS).

In the case study the anomaly detection task is considered. In it classification of user's activity is produced on the basis of network-based data (see

Fig. 1). Along with the dataset of examples of the security status “Normal” the dataset of class “Abnormal” is considered. The dataset presenting cases of the class “Abnormal” comprises the examples of four *attack types*: *Probing*, *Remote to local (R2L)*; *Denial of service (DOS)* and *User to root (U2R)*. The examples of attacks of each type selected for the case study are *SYN-scan*, *FTP-crack attack*, *SYN flood*, and *PipeUpAdmin* ([4, 17, 28, 29]).

Let us briefly describe the data sources used for the anomaly detection and, respectively, for the anomaly detection learning. These data sources are produced through preprocessing of the traffic raw data, corresponding to the network-based level. These data are presented by four data sources as follows ([8, 9]):

1. *Data stream of vectors of binary sequences specifying stream of headers of IP packets.* The components of this vector are constituted by different parameters of packet headers. Mining of such data stream is performed by an algorithm developed by the authors specifically for such kind of data streams. It is based on correlation and regression techniques. Respective source-based classifier produces stream of binary decisions belonging to $\{Normal, Alert\}$ and assessing each particular connection.
2. *Statistical attributes of connections (sessions of users) manifested in input traffic.* As features the duration, status, total number of connection packets and also six additional attributes specifying other statistics of the connection are used. Source-based classifier produces binary stream of decisions belonging to $\{Normal, Alert\}$ with regard to each particular connection.
3. *Statistical attributes of traffic (users' activity) during the short time (5 sec) intervals.* This data source is presented by four features specifying integral characteristics of input traffic that are total numbers of connections and services of different types during 5 sec. Source-based classifier operating with these data also produces stream of binary decisions belonging to $\{Normal, Alert\}$ with regard to connections occurred within particular sliding windows of 5 sec time interval.
4. *Statistical attributes of traffic (users' activity) for long time intervals.* In this data source, the same statistics as for short time interval are used as the features. The length of time interval is varied and corresponds to 100 connections.

The training and testing samples for each of above four types were produced on the basis of processing of *Tcpdump/ Windump* data (see Fig. 1)². *TCPtrace* utility was used for this purpose as well as several other programs.

Thus, the strategy of anomaly detection is organized in two steps. First, source-based classifiers asynchronously produce binary decisions taking values from the set $\{Normal, Alert\}$. These decisions constitute a binary vector of

² This preprocessing was carried out by Ph.D. student of SPIIRAS M. Stepashkin under supervision of Prof. I. Kotenko.

features processed at the upper level, whose purpose is to combine these decisions and produce the final one. This strategy corresponds to a well known meta-classification approach [19].

Meta-classification procedure possesses certain peculiarities caused by the fact that IDS is a real-time system, and source-based classifiers produce their decisions asynchronously, because each source-based classifier produces its decision at the time when it receives all the data needed for making decisions, which is irregular process. Figure 2 demonstrates this property graphically.

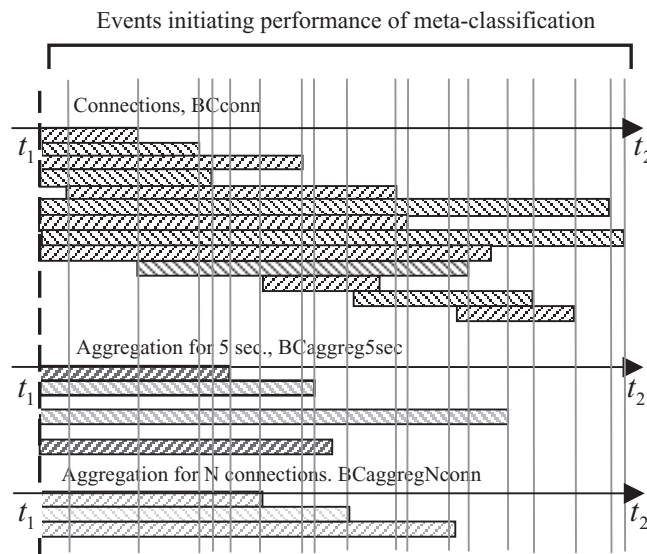


Fig. 2. Explanation of the model of meta-data used for training of meta-classifier and decision making within interval $[t_1, t_2]$. Vertical lines correspond to the times of arrival of new decisions of source-based classifiers that correspond to the times of producing decisions by meta-classifier

Let us call a decision newly produced by a source-based classifier an input *event* for meta-classifier and represent it as $\langle \text{Decision of source-based classifier } X, \text{Time} \rangle$. At a certain time some base classifiers have already produced decisions though other ones have not. To combine such decisions, meta-classifier can act in two modes: (1) to wait when all base classifiers produce the decisions and then combine them, and (2) update the combined decision at the time of a new input event arrival. Let us omit here a thorough analysis of both approaches, although this aspect is very interesting, and only explain and justify shortly the second variant selected in this work.

For this variant, there exists a risk that some of the decisions previously produced by the source-based classifiers are already not relevant to the situation (“too archaic”). To take this fact into account, for each base classi-

fier (data source) we introduce a so called *life time* of decisions produced by it. “Life times” are various for different sources, and their particular values are elicited from experts. If time interval elapsed from the moment of a decision producing exceeds its life time then in the meta-data the value corresponding to this decision is reckoned as missing. Therefore, in this case meta-classification is performed on the basis of data with missing values.

Thus, temporal and asynchronous nature of data sources leads to the data mining and decision making formal model with missing values.

The intrusion detection-based case study described above is used below for exploration of the properties and peculiarities of the developed lattice-based approach to data mining with missing values destined for rule extraction. Detailed specification of training and testing samples generated from input traffic containing both legitimate and illegitimate users’ activity is done in Sect. 5.

4 Mining Rules from Data with Missing Values

The main ideas of the developed approach to mining data with missing values are as follows:

- (1) To avoid imputation of missing values both in training and testing samples and also in classification of unseen data with missing values;
- (2) Instead of missing value imputation, to assess the upper and low bounds of the sets of rules that can be extracted from the training dataset under all possible assignments of missing values in it; and
- (3) To compute the above upper and low bounds by use of rigorous rule extraction techniques.

To better explain the ideas of the developed approach to extraction of rules from data with missing values, let us first outline general ideas of most existing mining techniques dealing with complete datasets. Although in this work we use GK2 algorithm developed by the authors [10, 11], for our purposes it is important to briefly remind about well known AQ basic algorithm, because GK2 differs from it only in algorithmic implementation of AQ basic step numbered below as step 3. It should be noted that AQ is the earliest proposed method of such kind [15], and up to now it remains a basic one for many other learning algorithms and systems (rough set-based [18], [27], RIPPER [3], CN2 [2], etc.). The last version of this method, AQ20, was implemented recently.

4.1 AQ Method

AQ method ([15]) aims at induction of rules in the form “*if <condition> then <class>*”, where *<condition>* is a conjunction of attributes (“attribute test”). Several rules can also be represented as disjunction of rules. A conjunct is said to *cover* an example if it takes value “*true*” for this example.

The basic AQ induces rules for each class step by step. Simple description of this algorithm is as follows [15]:

1. Divide all examples into subsets **PE** of *positive* and **NE** of *negative* examples;
2. Choose randomly or by design one example from **PE** and call it the *seed*;
3. Find a set of *maximally general rules* (*MGR*) characterizing the *seed*. The generalization limit is defined by the set **NE**: a resulted description of the *seed* is not allowed to cover any example from **NE**. The obtained set of rules is called a *star*;
4. According to a criterion select the best rules in the *star*;
5. If these rules jointly with the previously found rules cover all examples from **PE** then stop. Otherwise, find another *seed* among the uncovered examples in **PE** and go to 2.

Step 3 is performed by a special procedure called *star generation procedure* that is the central one in AQ and many other algorithms. The rule preference criterion is chosen according to the task at hand, and in many cases it combines several simple criteria, however, the rule coverage factor (the ratio of covered examples of the set **PE**) usually is the first among them.

4.2 Extraction Rules from Propositional Data with Missing Values: The Main Theorem

Let us assume that training dataset is presented in binary scale and assigned one of three values: “0”, “1” and “*” interpreted as “*false*”, “*true*” and “*unknown*” (missing) respectively. Also assume that the column corresponding to the class label does not contain missing values.

Briefly the idea of dealing with missing values without their imputation is as follows. Assume that the selected *seed* does *not contain missing values*, and the set of counterexamples, **NE**, does not contain cases equal to the selected *seed*. If we completely assigned missing values of training dataset in arbitrary way, we would be able to extract *maximally general rules* via a conventional technique, for example, through the AQ algorithm. Different variants of such assignments would lead to different sets of rules extracted. Let us denote the set of extracted rules for an arbitrary assignment of missing values as R_* .

It was found out that for each *seed* there exist two sets of *maximally general rules* serving as exact *low* and *upper* bounds for all possible sets of rules corresponding to any arbitrary assignments of missing values. They correspond to two special assignments of missing values in training dataset. *Given seed*, these bounds meet the following deducibility relations:

$$R_{low} \subseteq R_* \subseteq R_{upper} \quad (1)$$

where R_{low} —the exact *low* bound of all the sets of *MGR*; R_{upper} — the exact *upper* bound of all the sets of *MGR*, and R_* —the set of *MGR* corresponding to an arbitrary assignment of the missing values.

Informally, it could be said that R_{upper} bound corresponds to the “*optimistic*” assignment of the missing values, whereas R_{low} corresponds to the “*pessimistic*” one. Let us explain conceptually how “*optimistic*” and “*pessimistic*” assignments can be built and then justify the above assignments design formally.

Let us denote an arbitrary example of the training dataset as $t(i)$, where i is the index of this example in dataset table. Let k be the index of the *seed* (index of chosen positive example), I_k^+ be the set of indexes assigned to attributes of *seed*. While searching for *MGR* corresponding to the chosen seed $t(k)$, we ignore all the columns of training dataset, whose indexes do not belong to the set I_k^+ . After such reduction of the training dataset other positive examples and all negative examples can contain missing values for attributes with indexes from the set I_k^+ . Let us denote the index set of missing values in particular negative example $t(l)$ by $I_{l,k}^-$, the index set of undefined attributes in particular positive example $t(r)$, $r \neq k$, by $I_{r,k}^+$. Let us further consider *two variants of assignment of missing values* of attributes with indexes $i \in I_{l,k}^-$ (for negative examples) and with indexes $i \in I_{r,k}^+$ ³ (for positive examples):

$$t_i^l = -t_i^k, \quad i \in I_{l,k}^-, \quad l \in \mathbf{NE}, \quad (2)$$

$$t_i^r = t_i^k, \quad i \in I_{r,k}^+, \quad r \in \mathbf{PE}, \quad (3)$$

and

$$t_i^l = t_i^k, \quad i \in I_{l,k}^-, \quad l \in \mathbf{NE}, \quad (4)$$

$$t_i^r = -t_i^k, \quad i \in I_{r,k}^+, \quad r \in \mathbf{PE}. \quad (5)$$

The first assignment, (2)–(3), is such that it maximally “increases” the distinctions between the *seed* and negative examples, and maximally “increases” the similarities between the *seed* and other positive examples. On the contrary, the second one, (4)–(5), maximally “increases” the similarities between the *seed* and negative examples, and maximally “increases” the distinctions between the *seed* and other positive examples. Intuitively, the first assignment is called *optimistic* because it cannot decrease both the generalization level of the *MGR* generated from complete dataset and the values of the coverage factors of such *MGR*. In the second assignment that is called “*pessimistic*” both the generality of *MGR* generated from the complete dataset and values of their coverage factors cannot increase. The *Theorem 1* given below formulates formally the above facts introduced conceptually and indicates how to find the upper and low bounds of *MGR*.

³ Informally, $I_{l,k}^-$ comprises the subset of indexes of missing values in the negative example $t(l)$, which are assigned in *seed* $t(k)$. Analogously, $I_{r,k}^+$ comprises the subset of indexes of missing values in the positive example $t(r)$, which are assigned in *seed* $t(k)$.

Theorem 1. *Let us assume that seed $t(k)$ does not contain missing values. Let R_* be the set of all maximally general rules for an arbitrary assignments of missing values in the negative and positive examples, whose indexes $i \in I_{l,k}^-$, \mathbf{NE} , and $i \in I_{r,k}^+$, $r \in \mathbf{PE}$ respectively; be the set of all maximally general rules corresponding to the assignments of missing values in the positive and negative examples (2)–(3), and R_{low} be the set of all maximally general rules corresponding to the assignments of missing values (4)–(5).*

Then $R_{low} \subseteq R_* \subseteq R_{upper}$ ⁴.

This *Theorem* provides general framework for mining data with missing value. In other words, it restricts the search through explicit indication of the set of rules within which the *maximally general rules* should be found. Unfortunately it says nothing about how, based on the above upper and low bounds, to select the set of rules in a particular application. In general case, the rule set under search results from analysis of particular rule properties, appropriate organization of the testing procedure and use of requirements to the classification mechanism properties. The main of these aspects will be analyzed below as applied to the case study introduced in Sect. 3.

To demonstrate the *Theorem 1* numerically let us first explain briefly an algorithm of rule extraction that is used for this purpose below.

4.3 GK2 Algorithm for Extraction Rules

Rule induction algorithm GK2 [10, 11] operates with relational Boolean data. For demonstration of this algorithm, the data sample presented in Table 1 is used.

	C4.5		C5		C5 Boost		Genetically Induced Decision Trees		SVM(RBF)		MultiVeDec without Problem Adaptation		MultiVeDec	
	□	◇	□	◇	□	◇	□	◇	□	◇	□	◇	□	◇
av	76.7	44.5	81.4	48.6	83.7	50.0	90.7	71.4	83.7	55.7	93.0	78.7	90.0	71.4
breastcancer	96.4	94.4	95.3	95.0	96.6	96.7	96.6	95.5	96.5	97.1	96.6	95.1	96.9	97.8
heartdisease	74.3	74.4	79.2	80.7	82.2	81.1	78.2	80.2	39.6	42.7	78.2	78.3	83.1	82.8
Hepatitis	80.8	58.7	82.7	59.8	82.7	59.8	86.5	62.1	76.9	62.5	88.5	75.2	88.5	75.2
mracid	94.1	75.0	100.0	100.0	100.0	100.0	94.1	75.0	94.1	75.0	100.0	100.0	100.0	100.0
Lipids	60.0	58.6	66.2	67.6	71.2	64.9	76.3	67.6	25.0	58.9	75.0	73.3	75.0	73.3
Mvp	91.5	65.5	91.5	65.5	92.3	70.0	91.5	65.5	36.1	38.2	93.8	84.3	93.8	84.3

□ accuracy on test set ◇ average class accuracy on test set

The following notations are used hereinafter: $I_m = \{1, \dots, m\}$ – index set of attributes (see Table 1); $I_k \subset I_m$ – a subset of attribute indexes; \mathbf{PE} , \mathbf{NE} – the sets of indexes of positive and negative examples respectively; $t(k)$ – k -th example (row of the relational data table) of training dataset; $X = \{x_1, \dots, x_m\}$ – the set of propositions corresponding to the data attributes.

⁴ This Theorem is proved rigorously but here this proof is omitted for brevity.

Each of them can be used with negation (\bar{x}_i) or without it (x_i); in both cases it is called “literal”. A literal is denoted as \tilde{x}_i and it can take value x_i or \bar{x}_i ; Φ_X $\{x_1, \dots, x_m\}$ – the entire set of formulae that can be built over the proposition set $\{x_1, \dots, x_m\}$ by use of connectives $\&$ (conjunction), \vee (disjunction) and negation.

A premise $\langle condition \rangle$ of a rule, which GK2 induces in the form “if $\langle condition \rangle$ then $\langle predicted class \rangle$ ”, is a conjunction of the literals constituted by a subset of attributes. Formally, any rule R_j is specified as follows:

$$R_j = F_j \supset Q, F_j = \&_{i \in I_j} \tilde{x}_i, \quad (6)$$

where R_j stands for rule indexed by j , $F_j = \&_{i \in I_j} \tilde{x}_i$ is a rule premise, I_j is a subset of attribute indexes, Q is a class label, and \supset denotes logical implication connective. Further it is default assumed that each rule to be extracted has to be *consistent* that is a rule premise F_j can take “true” value only over the positive training examples PE . This also means that there is no pair of equal examples belonging to both PE and NE sets of the training sample.

If formula $F_j \supset Q$ (we speak about the formula as the whole but not about its premise) takes value “true” over all positive and negative examples of training dataset then it is called “consistent” with training dataset. However, if the formula is *consistent* with training dataset but its premise F_j covers no positive examples of class Q then it cannot be “interesting”.

The purpose of GK2 is, given the *seed*, to search for *MGR*. In particular applications certain additional constraints (constraint on maximal length of rules, minimal value of coverage factor, etc.) can additionally be imposed.

Additional notations used below are as follows: Φ – the set of formulae F_j present in (6) (potential premises of rules $R_j = F_j \supset Q$), $\Phi \subset \Phi_X$; $U \subseteq \Phi$ – a subset of $F_j \in \Phi$ with positive coverage factor over examples of class Q ⁵; $Z \subseteq U$ – the set of all premises of consistent formulae $F_j \supset Q$ having positive values of the coverage factor; $Z^+ \subseteq Z$ – the set of premises of the set of the *MGR*. corresponding to the given *seed*. Let us note that formulae sets $Z^+ \subseteq Z \subseteq U \subseteq \Phi$ introduced are the *sets of premises* of rules if the seed belongs to the class Q . In the same way the set of premises Z^- of the *MGR* can be introduced for the seeds of the class \bar{Q} .

Let us introduce a definition of the partial order relation between premises F_j of rules $R_j \in Z$.

Definition 1. Let $F_j \in Z$ and $F_r \in Z$. We say that $F_j \bar{\prec} F_r$ (“formula F_j is less than or equal to F_r ”) if and only if $I_j \subseteq I_r$, and all common literals of these formulae have the same “sign” (i.e. each pair of common propositions both are either negative or positive).

In other words, formulae F_j and F_r are such, that the former comprises a subset of literals of the latter and their common literals are of the same

⁵ Let us note that formulae of the set U can be inconsistent.

“sign”. Thus, formula F_j is “shorter” and this is why “more general” than formula F_r . It can be proved that this order imposes a lattice structure over all formulae $F_j \in \mathbf{Z}$, hence, $\mathbf{Z} = \langle \mathbf{Z}, \preceq \rangle$ is a lattice ([1]). According to the lattice properties, the lattice \mathbf{Z} contains the subset of so-called “minimal” (shortest) formulae \mathbf{Z}^+ , and this subset is such that \mathbf{Z}^+ is deductively equivalent⁶ to the complete formulae set \mathbf{Z} . It should be reminded that the formulae set \mathbf{Z}^+ comprises all the shortest (“minimal”) premises of rules under search. Thus, the task of search for *MGR* is equal to the search for rules belonging to the subset \mathbf{Z}^+ . GK2 is destined to search for the subset \mathbf{Z}^+ .

Conceptually, the main course of GK2 can be as follows: (1) to search for the set \mathbf{U} , (2) to search for its subset \mathbf{Z} , and, finally, (3) to isolate the subset \mathbf{Z}^+ from the set \mathbf{Z} . The subset \mathbf{Z}^+ corresponds to the complete set of *MGR*.

Let us consider the theoretical basis of GK2 algorithm implementing step 3.2 of the AQ algorithm presented in Subsect. 4.2.

Let the seed $t(k) \in \mathbf{PE}$ correspond to the k -th line of the training dataset \mathbf{T} . Its formal specification in propositional logic looks as follows:

$$F_k = \&_{i \in I_k} \tilde{x}_i = \tilde{x}_1^k \tilde{x}_2^k \dots \tilde{x}_m^k \quad (7)$$

where

$$\tilde{x}_i = \{x_i, \text{ if } t(k, i) = 1, \text{ or } \bar{x}_i, \text{ if } t(k, i) = 0\} \quad (8)$$

Formula $F_k \in \mathbf{U}$, because its coverage factor is positive, actually its coverage factor is no less than $1/|\mathbf{PE}|$, where $|\mathbf{PE}|$ is equal to the cardinality of the set \mathbf{PE} . It is also consistent according to the assumption that there is no identical example in \mathbf{NE} .

Each formula F_k defines a formula set U_k as follows:

$$U_k = \{F_{ki} \in \mathbf{U} : \preceq F_k\} . \quad (9)$$

where F_{ki} comprises a subset of literals constituting F_k , and the set \mathbf{U} , the entire set of formulae having positive values of coverage factor, is as follows:

$$\mathbf{U} = \cup_{k \in \mathbf{PE}} U_k . \quad (10)$$

Let us recall that the set \mathbf{U} can contain inconsistent premises, and our tentative purpose is to extract all consistent ones, i.e. to find the set \mathbf{Z} of all *consistent* premises of rules having positive coverage factor.

The premise (7) corresponds to one of the most specialized rules for the class Q . Generalization corresponds to deletion of certain literals from it, and this deletion either preserves the premise coverage factor value or increases it. Such a deletion has also to preserve consistency of the resulting premise. The last requirement restricts the possibility of rule generalization. Like AQ, the idea of GK2 is to search for consistent premises of minimal length.

⁶ It is said that a formulae set \mathbf{A} is deductively equivalent to the formulae set \mathbf{B} if and only if each formula of \mathbf{B} is inferable from the formulae set \mathbf{A} .

⁷ In (7) and hereafter the conjunction symbol is omitted for brevity.

GK2 algorithm aims to search for the complete set of $MGR \mathbf{Z}^+$ without search for the entire set of consistent rules with positive coverage factor \mathbf{Z} , that leads to considerable decrease of computations. Below this procedure is described in strictly formal way, but all its steps are explained informally by examples.

Let us specify an arbitrary k -th positive example $t(k)$ in terms of propositional formula like (7) indexing its literals by superscript k . The “similarity” k -th and l -th positive examples of the training sample corresponds to the coincidence of the literals with the same subscripts of the formulae F_k and F_l (both of them must be either positive or negative):

$$S(x_i^k, x_i^l) = \{1, \text{ if } \tilde{x}_i^k = \tilde{x}_i^l, \text{ and } 0, \text{ otherwise.}\} \quad (11)$$

Component-wise negation of the similarity vector sometimes is called a *distinction vector*. For each pair of examples, one can compute the binary vector of similarities.

Let us select an *example* $t(k) \in \mathbf{PE}$, and call it (like in AQ algorithm, [6]) a *seed*. Representation of this *seed* in terms of propositional formula looks like it is given in formula (7). Let us introduce the *similarity matrix* $\mathbf{S}(k)$ reflecting similarities between given *seed* $t(k)$ specified by F_k and all negative examples represented in the same form (7). The dimensionality of this matrix is equal to the number of attributes in the training data table \mathbf{T} and its size is equal to the cardinality of the set \mathbf{NE} . Thus, in the Table 2 matrix $\mathbf{S}(k)$ indicates similarities between *seed* number k and negative examples in its positions assigned “1”, and it indicates their distinctions in its positions assigned “0”. An example of \mathbf{S} matrix for *seed* $t(1)$ and negative examples, presented in Table 1, is given in Table 2.

The next step of GK2 is to introduce a lattice-like structure over the lines of matrix $\mathbf{S}(k)$, whose use decreases the amount of further computations.

Definition 2. Let $S_{r,k}$ and $S_{l,k}$ be two lines of the matrix $\mathbf{S}(k)$. We say that $S_{r,k} \geq S_{l,k}$ if and only if this inequality is held for $S_{r,k}$ and $S_{l,k}$ component-wise and values “0” and “1” are considered here as integers, i.e. $1 > 0$.

An example of such a partial ordering of lines of \mathbf{S} matrix given in Table 2 in terms of Hasse diagram is depicted in Fig. 3.

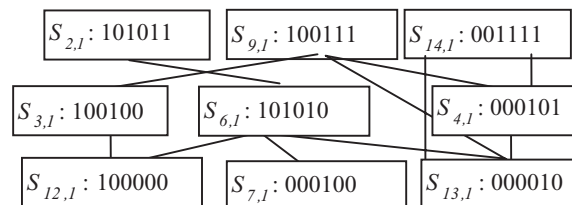


Fig. 3. Hasse diagram given over the lines of similarity matrix $S(1)$

The maximal elements of the set of lines of \mathbf{S} matrix play an important role in the GK2 algorithm: they and only they have to be accounted for in the further search for *MGR* because they accumulate the *maximal similarities* (indicated as “1” in \mathbf{S} -matrix) between *seed* and total batch of the negative examples. In Fig. 3 the maximal elements are $S_{2,1}$, $S_{9,1}$, and $S_{14,1}$.

Let $I_{\max}(k)$ stand for the set of maximal lines built for the *seed* k and $l \in I_{\max}(k)$ – index of a maximal line. Let $I_D(k, l)$ also stand for the set of indexes of attributes of the maximal line l assigned “0”. In other words, each $I_D(k, l)$ comprises the list of positions of the maximal lines $S_{l,k}$, where it differs from the *seed* k :

$$I_D(k, l) = \{i : i \in I_m, S_{k,l}(i) = 0\} . \tag{12}$$

For example, the sets $I_D(k, l)$ corresponding to the maximal negative lines in Fig. 3 are as follows:

$$I_D(2, 1) = \{2, 4\}, I_D(9, 1) = \{2, 3\}, I_D(14, 1) = \{1, 2\} . \tag{13}$$

The sets $I_D(k, l)$ indicate minimal distinctions between the *seed* and negative examples and therefore indicate the literals - candidates for keeping in the premise F_k (7) to preserve consistency.

Thus, given *seed*, a generalization algorithm intended to search for all *MGR* can be described as follows:

1. Given *seed* k , build a lattice of auxiliary family set $MAK(k) = \langle \{A_s(k)\}, \subseteq \rangle$ where all $A_s(k)$ meet two conditions:
 - a. $A_s(k) \in I_m$, i.e. it is a subset of the attribute indexes;
 - b. $A_s(k)$ has nonempty intersection with each $I_D(k, l)$:

$$[\forall l \in I_{\max}(k)] \{A_s(k) \cap I_D(k, l) \neq \emptyset\} .$$

2. Find the minimal sets of the lattice $MA(k)$.
3. Map each minimal set of the lattice $MA(k)$ the *maximally generalized rule* like it is prescribed in the *Theorem 2* given below.

Let us give an informal explanation of the algorithm described above. The major property of the sets $A_s(k) \subseteq I_m$ constituting the lattice is $MA(k)$ that each of them must contain *at least one attribute from every set of distinctions* $I_D(k, l), l \in I_{\max}(K)$. Therefore, minimal elements of $MA(k)$ correspond to the sets of minimal distinctions of the *seed* k for all lines of the similarity matrix $\mathbf{S}(k)$. Thus, the formula $F_{ki} = \&x_{j \in I_k} \tilde{x}_j \in U_k$ is a shortest premise of a consistent rule with positive value of the coverage factor if the set of indexes I_k is one of $A_s(k)$, and at that the “signs” of the respective literals in premise F_{ki} are the same as in formula F_k in (2). This condition guarantees the consistency of rule $F_{ki} \supset Q$. If such a formula is built on the basis of minimal elements of the lattice $MA(k)$, then it is one of the *maximally general* one. This conclusion is the content of the *Theorem 2*:

Theorem 2. A formula $F_{ki} = \&_{j \in I_k} \tilde{x}_j \in U_k$ is the premise of a consistent rule with positive value of coverage factor if and only if $I_k \in \{A_s(k)\}$.

Corollary 1. A formula $F_{ki} = \&_{j \in I_k} \tilde{x}_j \in U_k$ is the premise of a maximally general rule if and only if $I_k \in \{A_s(k)\}$ and $A_s(K)$ is a minimal element of the lattice $MA(k)$.

It is worth noting that *there is no need to compute the complete family set $\{A_s(k)\}$* . It is actually possible to compute directly its minimal elements on the basis of an algorithm of *minimal binary cover* (see, for example, [16]). One of such algorithms was developed by the authors of this paper and used in the GK2 software implementation.

Thus, a brief description of the GK2 algorithm for extraction rules from binary data is as follows:

1. (Same as in AQ) Divide training data sample into subsets **PE** of *positive* examples and **NE** of *negative* examples;
2. (Same as in AQ) Select a *seed*, $t(k) \in \mathbf{PE}$.
- 3.1. Compute $\mathbf{S}(k)$ matrix according to (11).
- 3.2. Compute the set of indexes $I_{\max}(k) = \{S_{l,k}\}$ of maximal negative lines of the $\mathbf{S}(k)$ matrix.
- 3.3. Compute the index sets $I_D(k, l)$ using (12).
- 3.4. Find minimal elements of the lattice $MA(k)$.
- 3.5. Generate the premises of *MGR* as indicated in *Theorem 2*.

The further steps of GK2 are the same as for AQ algorithm.

Numerical Example

Let us continue the example of search for *MGR* based on sample given in Table 1. The results corresponding to the step 3.3 are given in (13). The next step, 3.4, has to generate the family set $\{A_s(K)\}$ in order to build the lattice $MA(k)$ and to find its minimal elements. Although in the implemented software we use for this purpose an original algorithm of *minimal binary covers*, in example at hand $\{A_s(K)\}$ can just be built manually:

$$\{A_s(K)\} = \{\{2\}, \{1, 2\}, \{2, 4\}, \{2, 3\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\} .$$

Hasse diagram of the lattice $MA(1)$ structured according to the theoretic-set inclusion is given in Fig. 4. The *minimal* elements of the lattice $MA(1)$ are $\{2\}$ and $\{1, 3, 4\}$.

The next step 3.5 is to generate *maximally general rules* on the basis of minimal elements of the lattice $MA(1)$ that results in the following:

$$R_1 = \bar{x}_2 \supset Q, R_2 = \bar{x}_1 x_3 x_4 \supset Q .$$

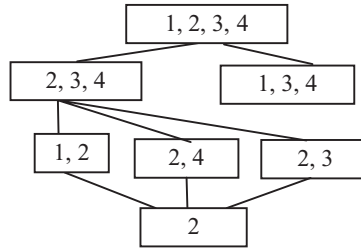


Fig. 4. Hasse diagram of the lattice $MA(I) = \langle \{\{2\}, \{1,2\}, \{2,4\}, \{2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}\}, \subseteq \rangle$

Let us notice that propositions x_1 and x_2 appear in rules R_2 and R_1 with negation because both of them have such “sign” in the formula F_1 built for the seed $t(1)$ (see Table 1).

The values of the coverage factors for these rules are as follows:

$$S(x_2, Q) = 0.4, S(\bar{x}_1 x_3 x_4, Q) = 0.4 .$$

4.4 Mining Rules from Data with Missing Values: An Example

Let the training sample with missing values be as given in Table 3. This dataset corresponds to the same data as in Table 1 but some attributes in the former are missing. The percentage of the missing data is about 25%. To perform a comparison of *MGR* that can be extracted from such a sample with the *MGR* extracted from completely defined sample, let us solve the task of extraction rules from data with missing values for the same seed, i.e. for the seed $t(1)$.

The *Theorem 2* assumes that seed does not contain missing values, therefore the first what is necessary to do is to delete from the training data the columns corresponding to the attributes, whose values are missing in the chosen seed, i.e. to delete columns corresponding to the attributes x_2 and x_5 , therefore $I_1^+ = \{1, 3, 4, 6\}$. The columns to be deleted are given in Table 3 in grey color. Thus, the training dataset must be such as presented in Table 3 without columns indexed by the subscripts second and fifth.

According to the *Theorem 2*, to search for the set of *MGR* rules R_{upper} , it is necessary to assign the missing values to the positive and negative examples as made in Table 4. Respectively, to search for the *MGR* rules of the set R_{low} , it is necessary to assign the missing values as shown in Table 5. Below both tasks (search for R_{upper} and search for R_{low}) are being solved in parallel.

The following computations correspond to the steps 3.1–3.5 of the GK2 algorithm:

Step 3.1 This step consists in computations of similarity matrices $S(1)$ for data given in Table 4 and in Table 5. The results are given in Table 6 and

Table 3. Boolean training data with missing values

$t(i)$	t_1^i		t_3^i	t_4^i	t_5^i	t_6^i	Class Q
$t(1)$	0	*	1	1	*	0	1
$t(2)$	*	1	1	0	1	*	0
$t(3)$	0	1	0	1	0	*	0
$t(4)$	1	1	0	*	0	0	0
$t(5)$	0	*	*	1	1	0	1
$t(6)$	*	*	1	0	1	1	0
$t(7)$	*	1	0	1	0	*	0
$t(8)$	1	0	1	*	1	*	1
$t(9)$	0	1	0	*	1	0	0
$t(10)$	1	1	0	0	*	0	1
$t(11)$	1	*	1	1	0	1	1
$t(12)$	*	1	*	0	0	1	0
$t(13)$	*	1	0	0	1	1	0
$t(14)$	1	*	1	1	1	0	0

Table 4. Training data used for extraction of the set R_{upper}

$t(i)$	t_1^i	t_3^i	t_4^i	t_6^i	Class Q
$t(1)$	0	1	1	0	1
$t(2)$	1	1	0	1	0
$t(3)$	0	0	1	1	0
$t(4)$	1	0	0	0	0
$t(5)$	0	1	1	0	1
$t(6)$	1	1	0	1	0
$t(7)$	1	0	1	1	0
$t(8)$	1	1	1	0	1
$t(9)$	0	0	0	0	0
$t(10)$	1	0	0	0	1
$t(11)$	1	1	1	1	1
$t(12)$	1	0	0	1	0
$t(13)$	1	0	0	1	0
$t(14)$	1	1	1	0	0

Table 5. Training data used for extraction of the set R_{low}

$t(i)$	t_1^i	t_3^i	t_4^i	t_6^i	Class Q
$t(1)$	0	1	1	0	1
$t(2)$	0	1	0	0	0
$t(3)$	0	0	1	0	0
$t(4)$	1	0	1	0	0
$t(5)$	0	0	1	0	1
$t(6)$	0	1	0	1	0
$t(7)$	0	0	1	0	0
$t(8)$	1	1	0	1	1
$t(9)$	0	0	1	0	0
$t(10)$	1	0	0	0	1
$t(11)$	1	1	1	1	1
$t(12)$	0	1	0	1	0
$t(13)$	0	0	0	1	0
$t(14)$	1	1	1	0	0

Table 7 respectively. Let us denote these matrices as $S_{upper}(1)$ and $S_{low}(1)$ respectively.

Steps 3.2. According to this step, it is necessary to order the negative lines of the matrices $S_{upper}(1)$ and $S_{low}(1)$ (see Fig. 5 and Fig. 6 respectively) and to find the maximal elements I_{max}^{upper} and I_{max}^{low} for both of them. The results are as follows:

Table 6. $S(l)$ matrix used to search for the set MA_{upper}

$t(i)$	t_1^i	t_3^i	t_4^i	t_6^i	Class Q
$t(1)$	1	1	1	1	1
$t(2)$	0	1	0	0	0
$t(3)$	1	0	1	0	0
$t(4)$	0	0	0	1	0
$t(5)$	1	1	1	1	1
$t(6)$	0	1	0	0	0
$t(7)$	0	0	1	0	0
$t(8)$	0	1	1	1	1
$t(9)$	1	0	0	1	0
$t(10)$	0	0	0	1	1
$t(11)$	0	1	1	0	1
$t(12)$	0	0	0	0	0
$t(13)$	0	0	0	0	0

Table 7. $S(l)$ matrix used to search for the set MA_{low}

$t(i)$	t_1^i	t_3^i	t_4^i	t_6^i	Class Q
$t(1)$	1	1	1	1	1
$t(2)$	0	1	0	1	0
$t(3)$	1	0	1	1	0
$t(4)$	0	0	1	1	0
$t(5)$	1	0	1	1	1
$t(6)$	1	1	0	0	0
$t(7)$	1	0	1	1	0
$t(8)$	0	1	0	0	1
$t(9)$	1	0	1	1	0
$t(10)$	0	0	0	1	1
$t(11)$	0	1	1	0	1
$t(12)$	1	1	0	0	0
$t(13)$	1	0	0	0	0

$$I_{max}^{upper} = \{3, 9, 14\} = \{1101, 1001, 0111\} ,$$

$$I_{max}^{low} = \{2, 3, 14\} = \{1101, 1011, 0111\} .$$

Step 3.3. Computation of the index sets of distinctions for both cases. This step results in the following:

$$I_D^{upper}(1, 3) = \{3, 6\}, I_D^{upper}(1, 9) = \{3, 4\}, I_D^{upper}(1, 14) = \{1\} ;$$

$$I_D^{low}(1, 2) = \{4\}, I_D^{low}(1, 3) = \{3\}, I_D^{low}(1, 14) = \{1\} .$$

Step 3.4. Building the lattices $MA^{upper}(1)$ and $MA^{low}(1)$ and determining its minimal elements. For brevity, we present the final result of this step, i.e. the sets $MA_{min}^{upper}(1)$ and $MA_{min}^{low}(1)$ of minimal elements of the above partially ordered sets:

$$MA_{min}^{upper}(1) = \{\{1, 3, 4\}, \{1, 3, 6\}\}; MA_{min}^{low}(1) = \{\{1, 3, 4\}\}$$

Step 3.5. The resulting sets of *maximally general rules* (upper and low bounds) are as follows:

$$R_{upper} = \{\bar{x}_1x_3x_4 \supset Q, \bar{x}_1x_4\bar{x}_6 \supset Q\} ;$$

$$R_{low} = \{\bar{x}_1x_3x_4 \supset Q\} .$$

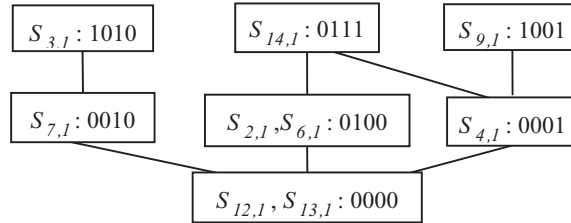


Fig. 5. Hasse diagram of the partially ordered set of the negative lines of similarity matrix $S_{upper}(1)$

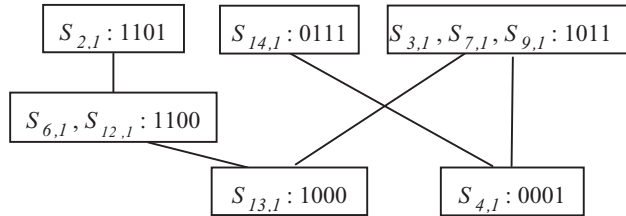


Fig. 6. Hasse diagram of the partially ordered set of the negative lines of similarity matrix $S_{low}(1)$

Let us discuss the results of the example. It is obvious that $R_{low} \subset R_{upper}$. The coverage factors of the rules from the set R_{upper} are

$$S(\bar{x}_1x_3x_4, Q) = 0.4, S(\bar{x}_1x_4\bar{x}_6, Q) = 0.6 ,$$

and the same for the rule from the set R_{low} is

$$S(\bar{x}_1x_3x_4, Q) = 0.2 .$$

Let us compare the sets of *maximally general rules* computed for complete data, $R_* = \bar{x}_2 \supset Q, \bar{x}_1x_3x_4 \supset Q$ with R_{upper} and R_{low} sets of rules. The rule $\bar{x}_1x_3x_4 \supset Q$ is included in all sets, but the rule $\bar{x}_2 \supset Q \in Z^+$ does not appear in both R_{upper} and R_{low} . This fact does not contradict the *Theorem 2*, because in the last example the deducibility relation presented in formula (1) is held.

While comparing the values of the coverage factors, one can see that $S(\bar{x}_1x_3x_4, v) = 0.4$ preserves the same value for rules from R_* and R_{upper} , but both of them are greater than the coverage factor of the same rule belonging to the set R_{low} . All these conclusions are compatible with the *Theorem 2*.

5 Classification Mechanism Design and Experimental Results

This section demonstrates how to practically exploit the result of the *Theorem 1* for direct extraction of rules from binary data with missing values and also presents some experimental results obtained based on the case study described in Sect. 2, i.e. intrusion detection case study.

5.1 Training and Testing Datasets

The training and testing datasets were computed through preprocessing the real-life *Tcpdump/Windump* data containing *Normal* and *Abnormal* users' activity. For preprocessing purpose *TCPtrace* utility as well as several other programs were used. Through preprocessing of data traffic level four datasets were computed. General description of the structure of the training and testing datasets was given in Sect. 3 although in the experiments whose results are presented below these datasets were simplified.

It was assumed that detection of intrusions is performed according to two-level classification structure. At the first level four source-based classifiers produce decisions based on "local" data. The *output* of each local classifier (they are also called *base-classifiers*) is represented as a binary time-stamped stream of decisions taking values from the set (*Normal, Alert*). These four *asynchronous* streams of decisions extended by three additional attributes indicating the name of data sources form the *input* of the classifier of the second level, *meta-classifier*, whose task is to on-line combine asynchronous streams of decisions produced by the base classifiers. Meta-classifier produces its decisions at the time when it receives a new decision ("*event*") from any base classifier. Due to asynchronous nature of the base classifiers decision making and finite "life time" of decisions produced, some previous inputs of meta-classifier to the time when the latter has to update classification can become out of date. If so then at that time the respective inputs of meta-classifier are considered as missing. The nature of missingness was explained in Sect. 3 (see also Fig. 2).

We omit here descriptions of the base classifiers training and testing algorithms, the results of rule mining and mechanisms of base-classifiers' decision making, since these subjects are beyond the paper scope. For our purpose it is sufficient to describe the inputs of meta-classifier, *meta-data with missing values*, specified in terms of streams of binary attributes. Since the primary goal of the experiments is to explore the proposed idea of direct mining of data with missing values, and not to design an effective intrusion detection system, we simplified the mechanism that accounts for the data ageing resulting in missingness of values. The idea behind this is to compare the sets of rules and classification qualities for two statements of the same task. The first task is to mine a complete dataset (if to ignore the finite values of "life

time”), generate classification rules and classification mechanism and to evaluate the quality of the resulting classification. The second task is similar but in it training and testing datasets were generated from the former datasets with the use of random missing data mechanism resulting in 20% of missing values. In the resulting datasets the number of missing values within particular examples varies from one to three and there exist about 20% of examples without missing values.

The resulting training datasets (both complete and with missing values) contain 128 examples; 14 of them correspond to the class “Normal” users’ activity and 114 correspond to the class “Abnormal”. The testing meta-data contain entirely different examples and consist of 11 examples of the class “Normal” and 233 examples of the class “Abnormal”. Both tasks, with and without missing values are solved in parallel what allows to assess the developed direct mining algorithm. Hereinafter for brevity, the following notations of meta-classifier input data attributes are used:

BK_ConnPacket \equiv X1; BK_ConnAggreg \equiv X2; BK_Aggreg5sec \equiv X3;
 BK_Aggreg100con \equiv X4; InitConn \equiv X5; Init5sec \equiv X6; Init100conn \equiv X7.

5.2 Algorithm for Direct Mining of Rules from Data with Missing Values

The procedure of direct mining of rules from data with missing values developed based on the *Theorem 1* consists in the following steps:

1. Assign the missing values of the training dataset “pessimistically” and mine the rules of the set R_{low} using an algorithm (we used for this purpose the GK2 algorithm) for classes Q and \bar{Q} . Denote the resulting rule sets as $R_{low}(Q)$ and $R_{low}(\bar{Q})$ respectively.
2. Assign the missing values of the training dataset “optimistically” and mine the rules of the set R_{upper} for classes Q and \bar{Q} . Denote the resulting rules’ sets as $R_{upper}(Q)$ and $R_{upper}(\bar{Q})$ respectively.

Note. In the experiments the length of rule premises was restricted by 3 literals.

3. Assess the quality of the extracted rules of the sets $R_{upper}(Q)$, $R_{upper}(\bar{Q})$ using certain criteria through testing procedure.

Notes. (1) We do not write about separate testing of the rules from the sets $R_{low}(Q)$ and $R_{low}(\bar{Q})$ due to deducibility relations $R_{low}(Q) \subseteq R_{upper}(Q)$ and $R_{low}(\bar{Q}) \subseteq R_{upper}(\bar{Q})$ following from the *Theorem 1*. (2) As the criteria of each rule quality we used a value of the coverage factor (in percentage with regard to the number of the positive examples in training dataset) and percentage of false positives. (3) In most cases the rules from the sets $R_{low}(Q)$ and $R_{low}(\bar{Q})$ possess more or less good quality.

4. Based on evaluation criteria, select the best rules from the sets for a use in classification mechanism that jointly provide a reasonable tradeoff between the total number of the selected rules and quality.

Note. The selection procedure is carried out by experts, and the selection is based on application dependent requirements to the probabilities of correct classification, false positives (false alarms) and false negatives (missing of attack).

5. Design classification mechanism and assess its performance quality.
6. If necessary, repeat some steps of the above algorithm to add new rules from the previously generated sets and/or generate additional rules.

5.3 Experimental Results

Let us describe experimental results in step by step fashion according to the above algorithm with comments at some of the steps. This description also aims to give more detailed explanations concerning the proposed algorithm for direct mining of data with missing values and to dwell upon some technical details of the algorithm implementation.

Steps 1 and 2 generate rules of the sets $R_{low}(Q)$, $R_{upper}(Q)$, $R_{low}(\bar{Q})$, and $R_{upper}(\bar{Q})$. Rule extraction procedure was restricted by search for rules, whose premises contain no more than three literals. Under this condition the total number of extracted rules for “*optimistic*” assignment of missing values is equal to 58 at that 35 of them belong to the rule set $R_{upper}(Q)$ (two of them also belong to $R_{low}(Q)$), and 23 – to $R_{upper}(\bar{Q}) \cup R_{low}(\bar{Q})$ at that $R_{low}(\bar{Q})$ contains eight rules not belonging to $R_{upper}(\bar{Q})$ but deducible from it.

Figure 7 demonstrates the sets of rules $R_{low}(\bar{Q})$ and $R_{upper}(\bar{Q})$, and the rules of the set $R_C(\bar{Q})$ extracted from the same training data though having no missing values. In this figure, the rules of the sets $R_{low}(\bar{Q})$, $R_C(\bar{Q})$ and $R_{upper}(\bar{Q})$ are marked with symbols “L” (“*Low*”), “C” (“*Complete*”) and “U” (“*Upper*”) respectively. It could be seen that the rules RS_{10} , RS_{21} , RS_{22} , RS_{23} are deducible from any of the rules RS_7 , RS_8 or RS_{15} ; the rules RS_4 , RS_5 and RS_6 are deducible from the rule RS_{17} , and the rule RS_{24} is deducible from any of the rules RS_1 , RS_2 , or RS_{17} . Let also note that the rules of the set $R_C(\bar{Q})$ either belong to or are deducible from the set of rules $R_{upper}(\bar{Q})$. Therefore deducibility relation $R_{low}(\bar{Q}) \subseteq R_C(\bar{Q}) \subseteq R_{upper}(\bar{Q})$ is held.

The finally selected rules’ sets $R(Q)$ and $R(\bar{Q})$, mapped by their relative *coverage factors* and *false positives* are presented in Table 8. Among the selected rules fourteen belong to the set $R(Q) \subseteq R_{upper}(Q)$ and eight – to $R(\bar{Q}) \subseteq R_{upper}(\bar{Q})$. The rules selected from the set $R_{upper}(\bar{Q})$ are highlighted grey in Fig. 8.

These rules were used to construct a classification mechanism, and it was constructed as a weighted voting algorithm where each rule of the set $R(Q)$ was weighted proportionally to the difference between relative coverage and relative false positives and then normalized to 1. The same weighting procedure was applied to the rules of the set $R(\bar{Q})$.

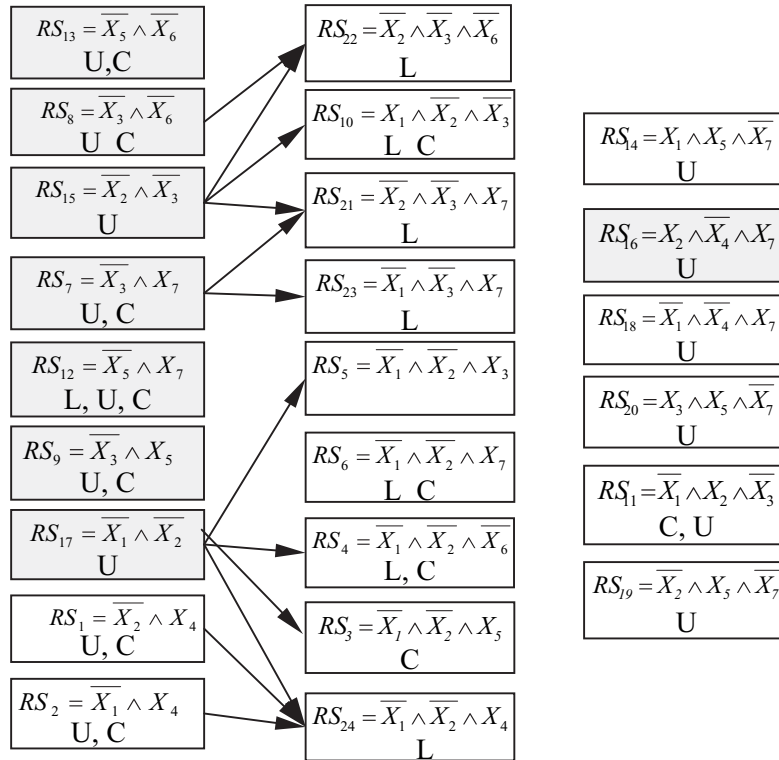


Fig. 7. The rules of the sets $R_{low}(\bar{Q})$, $R_C(\bar{Q})$, and $R_{upper}(\bar{Q})$. The rules selected for use in classification mechanism are given in grey

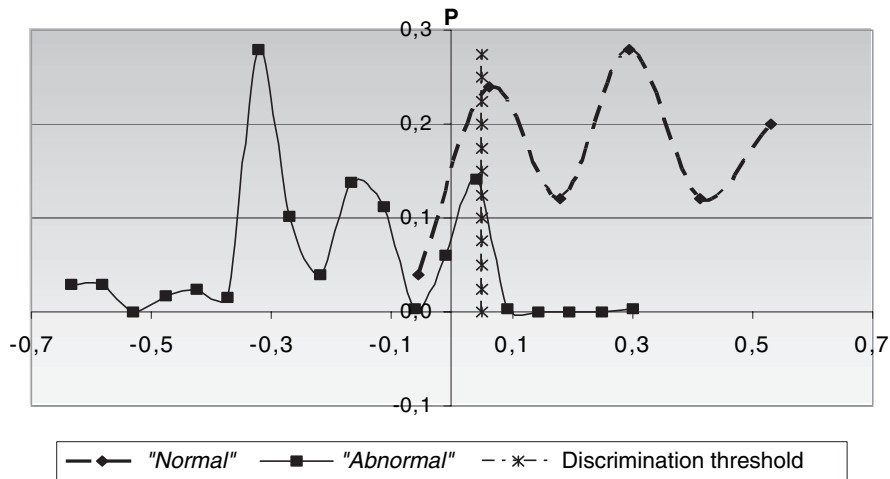


Fig. 8. Explanation of the choice of a threshold d' in classification mechanism (16)

Table 8. Qualities of the rules selected for a use in decision making mechanism

Rules for class Q (Normal)			R3	R4	R5	R7	R10	R12	R15	R19	R20	R22	R24
Coverage %			18	9	18	36	18	18	18	18	27	27	27
Refusals %			27	64	55	45	27	45	54	36	36	54	54
False positives %			55	27	27	18	54	36,	27	45	36	18	18
Coverage %			67	47	49	50	61	49	48	57	67	52	46
Refusals %			33	53	51	50	37	49	51	41	31	46	53
False negatives %			0	0	0	0	0,4	0,4	0	1,3	0,9	0,9	0
R25	R29	R38	Rules of class \bar{Q} (Abnormal)			RS7	RS8	RS9	RS12	RS13	RS15	RS16	RS17
18	9	18	Coverage %			63	63	54	63	72	45	45	54
54	45	36	Refusals %			36	36	45	36	27	54	36	45
27	45	45	False positives %			0	0	0	0	0	0	18	0
48	52	47	Coverage %			9,4	19	21	27	27	30	22	33
51	47	52	Refusals %			39	41	39	36	36	41	48	33
0	0	0	False negatives %									28	33

5.4 Classification Mechanisms

Three classification mechanisms were tested. The first of them was built based upon the weighted rules set $R(Q)$:

$$\tilde{Q} = \begin{cases} Q, & \text{if } W(X) \geq 0, \\ \bar{Q} & \text{otherwise,} \end{cases} \tag{14}$$

where $W = \sum_{i=1}^{|R(Q)|} a_i v[R_i(Q)]$, a_i —weights of the rule $R_i(Q) \in R(Q)$, $|R(Q)|$ —cardinality of the rule set $R(Q)$, and $v[R_i(Q)]$ —truth value of the premise of the rule $R_i(Q)$.

The second mechanism is analogous to the first one but built on the basis of the weighted rules set $R(\bar{Q})$:

$$\tilde{Q} = \begin{cases} \bar{Q}, & \text{if } W(X) \geq 0, \\ Q & \text{otherwise,} \end{cases} \tag{15}$$

where $W = \sum_{i=1}^{|R(\bar{Q})|} b_i v[R_i(\bar{Q})]$, b_i —weight of the rule $R_i(\bar{Q}) \in R(\bar{Q})$, $|R(\bar{Q})|$ —cardinality of the rule set $R(\bar{Q})$, and $v[R_i(\bar{Q})]$ —truth value of the premise of the rule $R_i(\bar{Q})$.

The third classification mechanism exploits both $R(Q)$ and $R(\bar{Q})$ sets of rules. Since the cardinalities of these sets are different, it is necessary to previously normalize them to provide the equal values of the sums of the weights

of rules a_i and b_i :

$$\sum_{i=1}^{|R(Q)|} a'_i = \sum_{i=1}^{|R(\bar{Q})|} b'_i = 1,$$

that can be easily achieved by respective normalization of the previously computed weights a_i and b_i .

Thus, the third classification mechanism is as follows:

$$\tilde{Q} = \begin{cases} Q, & \text{if } D(X) \geq \bar{d}, \\ \bar{Q} & \text{otherwise} \end{cases} \quad (16)$$

where $D = \sum_{i=1}^{|R(Q)|} a'_i v[R_i(Q)] - \sum_{i=1}^{|R(\bar{Q})|} b'_i v[R_i(\bar{Q})]$, \bar{d} – experimentally chosen threshold. Figure 8 demonstrates how the value of \bar{d} was chosen. (in our case the chosen value is equal to 0,05).

The results of the testing of three above mentioned mechanisms are depicted in Fig. 8 and Fig. 9. In Fig. 8 two curves are depicted; the right one presents the estimation of the probability density for the value of d built for the cases when an input example of the classification mechanism (16) belongs to the class Q (“Normal”), whereas the left one presents the estimation of the probability density for the examples belonging to the class \bar{Q} (“Abnormal”). The choice of the threshold value of \bar{d} depends on the application-oriented requirements to the classification mechanism quality, which as a rule is selected as a tradeoff between probabilities of correct classification, false positive and false negative. Let us note that the square under right curve given $d \leq \bar{d}$ corresponds to the *false negatives* (in our case – false alarms) probability whereas the square under left curve given $d > \bar{d}$ corresponds to the *false positives* (in our case – missing of signals) probability.

Figure 9 shows that within the case study in question classification mechanism (16) reveals rather good performance quality. In particular, it provides the estimated probability of the correct classification close to 0,997 if tested on joined testing and training samples with missing values (the right most diagram in Fig. 9). Testing of this mechanism on joined testing and training samples without missing values results in the estimated probability of the correct classification close to 0,99 (the second diagram at the left most). This figure also demonstrates the results of testing classification mechanisms (14) and (15). It can be seen that their performance qualities are worse than performance quality of the mechanism (16).

In general, the presented experimental results allow to optimistically evaluate the developed approach to direct mining of rules from the dataset with missing values.

6 Conclusion

The paper develops an approach to and technique for the lattice based direct mining of binary data with missing values aimed at extraction of classification

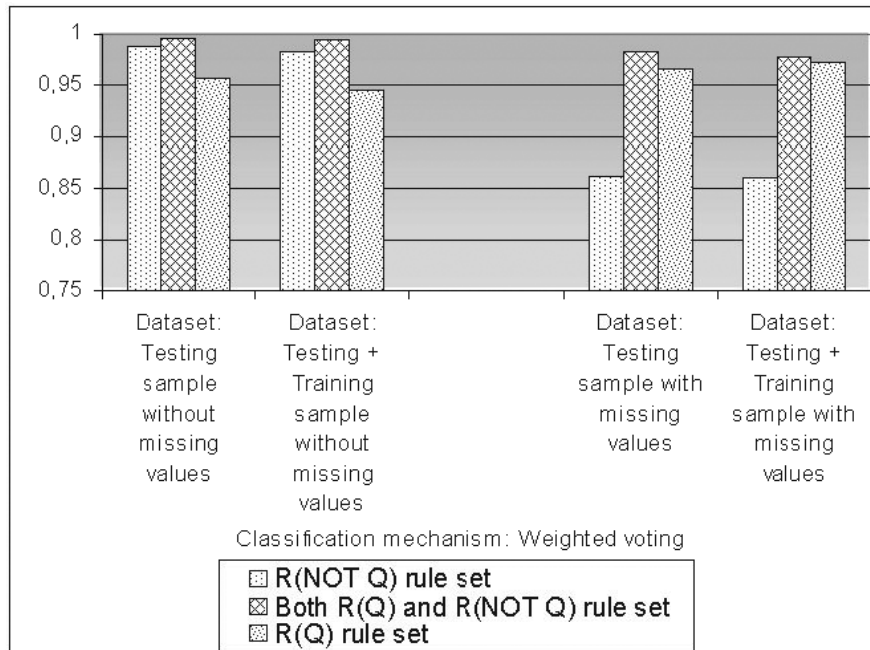


Fig. 9. The testing results demonstrating performance quality of three classification mechanisms. Vertical axis corresponds to the estimations of the probabilities of correct classification for four different testing samples

rules with premises represented in a conjunctive form. The proposed approach does not assume an imputation of missing values. The idea is to generate two sets of rules, which serve as the upper and low bounds for any other sets of rules corresponding to all possible assignments of missing values, and then, based on these upper and low bounds of the rules' sets, on testing procedure and on a classification criterion, to select a subset of rules to be used for classification. Instead of missing values imputation, the proposed approach uses training dataset to cut down the upper bound rule set via testing its rules on other dataset with missing values with the subsequent selection of the most appropriate rules.

The motivation of imputation avoidance is that there exist a number of important applications where information cannot be collected due to certain reasons; for example, due to conditions of natural environment (for example, if airborne observation equipment is used, the information can be unavailable due to meteorological factors or due to masking). Much more specific and complicate case, in which missing values imputation is not applicable at all, can arise as a result of temporal and/or spatial nature of data, asynchronous and distributed mode of data collection, and discrete and distributed mode of decision making. The detection of intrusions in computer network

demonstrates an application of such kind. In this application, input data of an intrusion detection system operating on the basis of multiple data sources can be represented as temporal stream of decisions received from distributed solvers (agents) responsible for forming alerts in case of illegitimate users' activity.

The proposed approach to the direct mining of rules from data with missing values is mathematically sound. One of its advantages is that it can be implemented by the use of any well known methods intended for extraction of rules having the premises in conjunctive form. Particularly, it can be implemented on the basis of such well known methods as AQ, CN2, RIPPER, etc. In our research we use the GK2 algorithm which is close to AQ. The paper gives a brief description of GK2 algorithmic basis.

Experimental results aiming at exploration and validation of the proposed approach to direct mining of rules from data with missing values confirm that it can yield strong and reliable results. Indeed, analysis of the set of rules extracted by its use and its comparison with the set of rules that would be extracted from the same dataset if it did not contain missing values exhibited their considerable overlapping. Example of such a comparison given in Fig. 7 demonstrates this fact, particularly, here only one rule, RS_{16} , belonging to the upper bound rules' set does not belong to the set of rules that could be extracted from the same data having no missing values.

The approach will be further tested on different applications from various areas with the focus on applications from a situational awareness scope. An interesting future task in regard to direct mining of data with missing values is the use of dependencies between attributes to evaluate the predictive power of the rules extracted by direct mining along with their evaluations on the basis of testing procedures.

Acknowledgement

This research is supported by European Office of Aerospace Research and Development (AFRL/IF), Project #1993P, and Russian Foundation of Basic Research (grant #04-01-00494a).

References

1. Birkhoff, G.: Lattice Theory. Providence, Rhode Island (1963)
2. Clark, P. Niblett, T.: The CN2 Induction Algorithm. Machine Learning Journal, Vol. 3 (1989) 261–283
3. Cohen, W.W.: Fast Efficient Rule Induction. Machine Learning: The 12th Intern. Conference, CA, Morgan Kaufmann (1995)
4. Cole, E.: Hackers Beware. New Riders Publishing (2002)
5. D.A.P., L.N.M., R.D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B, Vol.39 (1977)

6. Endsley, M.R., Garland, D.G. (Eds.): *Situation Awareness Analysis and Measurement*. Mahwah, NJ: Lawrence Erlbaum (2000)
7. Goodman, I., Mahler R., Nguen, H.: *Mathematics of Data Fusion*. Kluwer Academic Publishers (1997)
8. Gorodetsky, V., Karsaev, O., Kottenko, I., Samoilov, V.: *Multi-Agent Information Fusion: Methodology, Architecture and Software Tool for Learning of Object and Situation Assessment*. In *Proceedings of the International Conference "Information Fusion-04"*, Stockholm, Sweden, June 28 to July 1 (2004)
9. Gorodetsky, V., Karsaev, O., Samoilov, V.: *Multi-agent Technologies for Computer Network Security: Attack Simulation, Intrusion Detection and Intrusion Detection Learning*. *International Journal of Computer Systems Science and Engineering*. Vol.18, No.4, (2003) 191–200
10. Gorodetsky, V., Karsaev, O.: *Mining of Data with Missing Values: A Lattice-based Approach*. *International Workshop on the Foundation of Data Mining and Discovery in the 2002 IEEE International Conference on Data Mining, Japan*, (2002) 151–156
11. Gorodetsky, V., Karsaev, O.: *Algorithm of Rule Extraction from Learning Data*. *Proceedings of the 8th International Conference "Expert Systems & Artificial Intelligence" (EXPERTSYS-96)*, (1996) 133–138
12. Kononenko, I., Bratko, I., and Roskar: *Experiments in Automatic Learning of Medical Diagnostic Rules*. Technical Report. Josef Stefan Institute, Ljubljana, Yugoslavia (1984)
13. Liu, W.Z., White, A.P., Thompson, S.G. Bramer, M.A.: *Techniques for Dealing with Missing Values in Classification*. *Advances in Intelligent Data Analysis* (Liu, X., Cohan, P., Bertold, M., Eds). *Proceedings of the Second International Symposium on Intelligent Data Analysis*. Springer Verlag, (1997) 527-536
14. M.Magnani. *Techniques for Dealing with Missing Values in Knowledge Discovery Tasks*. (2003) <http://magnanim.web.cs.unibo.it/data/pdf/missingdata.pdf>
15. Michalski, R.S.: *A Theory and Methodology of Inductive Learning*. *Machine Learning*, vol.1, Eds. Carbone, J.G., Michalski, R.S. Mitchel, T.M., Tigoda, Palo Alto, (1983) 83–134
16. Nayak, R., Cook, D.J.: *Approximate Association Rule Mining*. *Proceedings of the Florida Artificial Intelligence Research Symposium* (2001)
17. Northcutt, S., McLachlan, D., Novak, J.: *Network Intrusion Detection: An Analyst's Handbook*. New Riders Publishing (2000)
18. Pawlak, Z.: *Rough Sets*. *International Journal of Computer and Information Sciences*, #11, (1982) 341–356
19. Prodromidis, A., Chan, P., Stolfo, S.: *Meta-Learning in Distributed Data Mining Systems: Issues and Approaches*, *Advances in Distributed Data Mining*, AAAI Press, Kargupta and Chan (eds.), (1999) <http://www.cs.columbia.edu/~sal/hpapers/DDMBOOK.ps.gz>
20. Quinlan, J.R.: *Unknown Attribute Values in Induction*. *Proceedings of the Sixth International Workshop on Machine Learning* (B.Spatz, ed.), Morgan Kaufmann, (1989) 164–168
21. Quinlan, J.R.: *Induction of Decision Trees*. *Machine Learning*. Vol. 1, (1986) 81–106
22. Ragel, A., Cremilleux, B.: *Mvc—a Preprocessing Method to Deal with Missing Values*. *Knowledge-Based Systems*, (1999) 285–291

23. Ragel, A., Cremilleux, B.: Treatment of Missing Values for Association Rules. Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD-98), X. Wu, R. Kotagiri, and K. B. Korb, eds., LNCS, Vol. 1394, Berlin, Springer Verlag, (1998) 258–270
24. Rubin, D.B.: A Overview of Multiple Imputation. Survey Research Section, American Statistical Association (1988) 79–84
25. Rubin, D.B.: Multiple Imputation for Nonresponse in Surveys. John Willey and Sons (1987)
26. Sandle, I.G.: Hot-deck Imputation Procedures. Incomplete Data in Sample Survey, Vol.3 (1983)
27. Skowron, A.: Rough Sets in KDD. In Proceedings of 16th World Computer Congress, Vol. “Intelligent Information Processing”, Beijing (2000) 1–17
28. Scambray, J., McClure, S.: Hacking Exposed Windows 2000: Network Security Secrets. McGraw-Hill (2001)
29. Scambray, J., McClure, S., Kurtz, G.: Hacking Exposed. McGraw-Hill (2000)
30. Weiss, S., Indurkha, N.: Decision Rule Solutions for Data Mining with Missing Values. IBM Research Report RC-21783 (2000)