Knowl Inf Syst (2013) 34:521–546
DOI 10.1007/s10115-012-0488-7

REGULAR PAPER

Decision rules extraction from data stream in the presence of changing context for diabetes treatment

Jakub M. Tomczak · Adam Gonczarek

Received: 24 May 2011 / Revised: 19 January 2012 / Accepted: 5 March 2012 /
Published online: 28 March 2012

Abstract The knowledge extraction is an important element of the *e-Health* system. In this paper, we introduce a new method for decision rules extraction called Graph-based Rules Inducer to support the medical interview in the diabetes treatment. The emphasis is put on the capability of hidden context change tracking. The context is understood as a set of all factors affecting patient condition. In order to follow context changes, a forgetting mechanism with a forgetting factor is implemented in the proposed algorithm. Moreover, to aggregate data, a graph representation is used and a limitation of the search space is proposed to protect from *overfitting*. We demonstrate the advantages of our approach in comparison with other methods through an empirical study on the *Electricity* benchmark data set in the classification task. Subsequently, our method is applied in the diabetes treatment as a tool supporting medical interviews.

Keywords Decision rules · Forgetting · Incremental learning · Hidden context · Diabetes

1 Introduction

According to the World Health Organization (WHO), diabetes[1] is one of the most dangerous chronic disease that afflicted around 171 million people in the world in the year 2000 and is projected to increase to 366 million in 2030 [68]. The increasing number of diabetics entails growing total costs of a treatment that are at the level of 120 billion Euros recently [20,33]. The costs of the treatment include not only pharmacological treatment, but also primarily hospitalization, laboratory tests, medical visits, and constant patient health monitoring. Thus,

[1] Diabetes is a condition primarily defined by the level of hyperglycemia giving rise to risk of microvascular damage, i.e., retinopathy, nephropathy and neuropathy [27,68].

J. M. Tomczak (✉) · A. Gonczarek
Institute of Computer Science, Faculty of Computer Science and Management,
Wrocław University of Technology, Wybrzeze Wyspianskiego 27, 53-137 Wrocław, Poland
e-mail: jakub.tomczak@pwr.wroc.pl

to lower the costs and make the disease bearable for a patient so-called *e-Health* systems are proposed [1,35,50,51].

Such systems help the patient conduct measurements (e.g., the level of glucose in the blood) or remind about taking medicines [13] but also to consult via mobile/computer network [65], make an appointment with a physician [41], support image analysis [44], or support decision making [7]. Sometimes they play a crucial role in the educational aspect of the disease and help to learn about patient's condition [38].

From a medical perspective, it is obvious that a medical interview, called *anamnesis*, is a crucial element in diagnosing and providing a medical care to the patient. Therefore, in most of the *e-Health* systems, a module for supporting medical interview is implemented. However, in the systems for the diabetes treatment, such module is rather designed to make simple aggregations (e.g., statistics or trends [49], rarely predictions [26]). Applications with knowledge extraction in a form easily understandable for a human being, for example, a decision tree [48] or decision rules [60], are rather uncommon. In either way, the medical society strongly emphasize an important role of anamnesis in the diabetes treatment [27]. The knowledge about *the context* of the patient determines the interpretation of measurements and further treatment schedule. The context is understood as a set of factors affecting patient's condition, such as feeding habits, sport activities, mood (e.g., stress, tension, relationships), and general health condition. However, measuring the context directly is difficult, if not impossible, hence it should be partially or fully obtained during anamnesis.

Therefore, in this work, a method for knowledge extraction in a form of decision rules to support anamnesis is presented. This method is implemented in a system called *eDiab* [60]. The *eDiab* system is in the preliminary phase of development (more technical details can be found in [60]), and this work contributes to proposing an algorithm for rules induction that enables a physician to conduct personalized and detailed medical interview. However, to solve the problem of the medical interview, two issues should be considered. First, knowledge representation should be chosen. In this work, decision rules are applied because they are easily understandable by a human being and could be seamlessly translated to a natural language. Second, the context is unknown and non-stationary (evolving in time). Thus, it is impossible to apply models that assume stationarity, e.g., Hidden Markov Models [6,36]. A reasonable solution is to apply a knowledge extraction method with *incremental learning* and a *forgetting mechanism*.

Incremental learning is primarily focused on processing data in a sequential way. In other words, knowledge is updated incrementally by processing examples one-by-one. However, to validate knowledge in the presence of changing hidden context, a forgetting mechanism should be proposed. The forgetting mechanism enables the knowledge to follow changes of the observed phenomenon. It means that parts of the knowledge which do not reflect current observations are *forgotten*, i.e., removed. In the literature, there are two ways of forgetting [45,59]: (i) *explicit* forgetting, (ii) *implicit* forgetting. The explicit forgetting assumes that knowledge is updated and validated based on the fixed number of recent examples with constant weighting, so-called *shifting window*, or with exponential weighting, so-called *forgetting factor*. Weighting means that each observation contributes to learning with a weight. In the forgetting with shifting window, all observations maintained in the window have weight equal to 1, and all other examples—0. In the exponential forgetting, a forgetting factor weighs observation in such a way that the older observation is, the less it influences a model. On the other hand, implicit forgetting does not apply weighting according to the time criterion but some other mechanism is used, for example, examples are weighted spatially [59]. In this work, a method with incremental learning and forgetting with forgetting factor is proposed.

The work is organized as follows. In Sect. 2, the review of existing solutions is presented. In Sect. 3, the problem is formally stated. In Sect. 4, the proposed algorithm is outlined. In Sect. 5, the forgetting mechanism for proposed algorithm is described. In Sect. 6, two empirical studies are presented: (i) based on *Electricity* benchmark data set [28], (ii) based on real-life measurements of diabetics [63]. Section 7 concludes the research paper.

2 Related work

There are many knowledge extraction methods that represent knowledge in a way easily understandable for a human being such as: rules inducers, for example, AQ [46], CEA [47], CN2 [15], RIPPER or its modifications [55], and tree inducers, for example, CART [9], ID3 [56], C4.5 [57]. All of them apply *batch learning* paradigm which means that the knowledge is extracted based on the entire set of training examples.

However, in the presence of data stream and an unknown, changing context, so-called *hidden context* [29,67], the batch learning fails. In the batch learning, a growing amount of training data increases the processing time and decreases the available memory space. Furthermore, even if all examples could be handled by the system, the knowledge extracted from past data is hardly valid and useful at the moment of decision-making. For this reason, an incremental learning paradigm should be applied. Additionally, in the presence of the changing context, a forgetting mechanism has to be used.

One of the first algorithms for rules induction with incremental learning and forgetting was introduced by Maloof and Michalski called AQ with Partial Memory (AQ-PM) [45]. Their approach uses the AQ algorithm [46] as a rules inducer but examples in the learning sequence are selected. The selection is made in two ways. First, only recent examples are maintained (explicit forgetting). Second, from the maintained examples only those are kept that were incorrectly classified by the knowledge (implicit forgetting). Then, a new knowledge is extracted from examples in the partial memory, that is, recent selected examples. The idea of AQ-PM is very interesting even though the performance of the method is unsatisfactory.

Recently, some other modifications of AQ algorithm were proposed [61]. The first modification uses the explicit forgetting only if the classification accuracy drops significantly. The second modification induces rules based on the shifting window at each learning step and a validation step is added: rules are removed if their evaluation value is below given threshold. This is another way of applying the implicit forgetting but on the model itself (not only on examples as in AQ-PM). In comparison with AQ-PM, both mentioned AQ modifications performed slightly better but still rather mediocre.

Another approach was presented by Widmer and Kubat [67] called FLORA (acronym standing for *FLOating Rough Approximation*). The basics of the method were originally introduced by Kubat [40] with the idea to use the rough set theory for inducing rules. The model consists of descriptions for positive, negative and noisy examples. Each description is parameterized by a coverage measure [30]. To keep up-to-date knowledge, the explicit forgetting is used to remove older examples and the implicit forgetting to change the coverage measure values. Additionally, to change the size of the shifting window, heuristics is applied. The problem of the FLORA methods is high computational complexity, which precludes its further practical usage.

For building decision trees, there are two approaches particularly noteworthy. The first approach, called SPLICE [29], uses temporal batch learning. The data stream is divided

into clusters and then the C4.5 algorithm is applied to extract knowledge on each cluster. Similar idea was proposed in [25] where training examples are divided into clusters and rules are induced from each cluster. However, SPLICE works off-line and can be used only to analyze historic data. The second approach, called CVFDT [32], applies the explicit learning for changing the tree structure by repeatedly applying the VFDT algorithm [19]. The algorithm formulates an alternative subtree for each attribute having a relatively high information gain and replaces the old subtree when a new one becomes more accurate.

An original approach is to use a graph structure as in OLIN algorithm [42]. The idea is to build information-fuzzy network (IFN) to learn and track the changing context. Next, to obtain rules, an additional method for rules induction from IFN has to be applied [43]. Each layer in the IFN corresponds to an attribute, and layers consist of nodes that contain values of previous attributes plus values of attribute in a given layer. Thus, the last layer represents nodes with values of different attributes, which are connected with decisions. An edge in the network represents a connection between two attributes. The decision about merging two attributes is based on the information-theoretic measure. To follow context changes, a shifting window is used. The OLIN is a novel approach from the rules induction perspective, since it uses graph-based representation to maintain the knowledge. However, knowledge updating is accomplished by a graph structure modification that entails possible knowledge loss (examples are not aggregated anywhere).

Another idea connected with graphical representation of rules is to use Pawlak's flow graphs [52]. Rules are represented as a flow network and flows in the network correspond to strength of rules. Moreover, Pawlak has shown several expressions that bound the accuracy and coverage with the form of Bayes'-like theorem but with a truth level interpretation instead of probabilities. Pawlak's approach is very interesting and gives another perspective at rules in terms of network flows. Nevertheless, flow graphs are used rather to represent rules than to aggregate data. Therefore, in the form they are given, they cannot be applied to rules induction successfully.

However, the OLIN algorithm and the flow graphs can be regarded as a part of a fast developing subfield of graph-based machine learning and knowledge extraction methods [16,31,66]. Graph-based methods were used for rules induction [54,58], clustering [23], substructure detection [16,34]. However, for such methods, there is still a need for developing proper forgetting mechanisms and a method for applying incremental learning.

A lot of effort has been done to handle data streams and the hidden context. However, all approaches mentioned above have one substantial disadvantage—the performance is still unsatisfactory. Moreover, due to the non-parametric character of the rule-based model, the forgetting is performed on the model structure, not on parameters. This is a huge obstacle because not only knowledge has to be kept in memory but examples as well. If any changes are made in the model, for example, some rules are removed, lost knowledge cannot be restored. Hence, it is a great challenge to propose a method of aggregating data, so that the learning phase is conducted in polynomial time and the forgetting is performed on parameters, not on the model itself. Such forgetting would give a fast and easy method of adapting to context changes and does not force any additional procedures for examples handling and maintaining. Furthermore, if examples are aggregated somehow, the knowledge can be even completely removed and later restored from aggregated data at any time.

3 Problem background

3.1 Knowledge extraction problem statement

In general, the problem of *knowledge extraction* (or *learning*) is considered a problem of finding an unknown dependency of an analyzed system using a limited number of observations [12,14,17,64].

The knowledge extraction from examples can be described using the following components [14,64]:

(i) A **generator** of random input vectors $\mathbf{u} = [u^1 \ u^2 \ldots u^D] \in \mathcal{U}$, drawn independently from a fixed probability distribution function $P(\mathbf{u}|c_n)$, which is unknown.
(ii) An **object** that returns an output value $y \in \mathcal{Y}$ to a given $\mathbf{u}$, according to the fixed conditional probability density function $P(y|\mathbf{u}, c_n)$, which is also unknown.
(iii) A **learning algorithm** that is capable of implementing a model, $\bar{y} = \Phi(\mathbf{u})$.
(iv) An **environment** (or **context**) that influences the input generator and the object and is assumed to be unknown and unobservable, $c_n \in \mathcal{C}$.

Remark The term *model* and *knowledge* are very often used interchangeably. However, knowledge refers to the model with determined structure or/and parameters.

In the considered domain of diabetes, the input represents quantities that describe a patient and a measurement, for example, weight, pressure, part of a day, day of a week, information about a meal. The output can represent the level of glucose in blood or whether it is in norm or not. In further considerations, it is assumed that $\mathcal{Y} = \{-1, 1\}$ where $y = 1$ and $y = -1$ denote normal and abnormal levels of glucose in blood, respectively. Moreover, it is assumed that all input values are nominal (discrete), that is, for all $d = 1, 2, \ldots, D$, $u^d \in \mathcal{U}_d$, $\text{card}\{\mathcal{U}_d\} = K_d < \infty$. For further clarity, let K denote a sum of all K_d.

Morover, as previously mentioned, probabilities for the inputs and the output are unknown. There is only given a training sequence of N examples, which are drawn from a distribution $P(\mathbf{u}, y|c_n) = P(\mathbf{u}|c_n) \cdot P(y|\mathbf{u}, c_n)$:

$$\left\{\left(\mathbf{u}_n, y_n\right)\right\}_{n=1}^{N}. \tag{1}$$

The examples in the class $y_n = 1$ will be called *positive* examples and in the class $y_n = -1$—*negative*. Moreover, it is worth stressing that examples are drawn from the distribution dependent on the context.

The learning algorithm is supposed to choose the *best* model for the given training sequence and it has to follow the changing context. Hence, it should choose a sequence of models (knowledge) over a period when M values of context appear, $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_M)$. The sequence of models has to minimize following functional (quality criterion):

$$Q_M(\mathbf{\Phi}) = \sum_{m=1}^{M} \mathbb{E}_{\mathbf{u}, y|c_m}\Big[L\big(\Phi_m(\mathbf{u}), y\big)\Big], \tag{2}$$

where $\mathbb{E}[\cdot]$ is an expected value, $L(\cdot, \cdot)$—loss function, for example,

$$L\big(\Phi(\mathbf{u}), y\big) = \begin{cases} 0, & \text{if } y = \Phi(\mathbf{u}), \\ 1, & \text{if } y \neq \Phi(\mathbf{u}). \end{cases} \tag{3}$$

It is easy to notice that to minimize the (2) it is enough to minimize

$$Q_m(\Phi_m) = \mathbb{E}_{\mathbf{u}, y|c_m}\Big[L\big(\Phi_m(\mathbf{u}), y\big)\Big] \tag{4}$$

for each $m = 1, 2, \ldots, M$.

Then, the *Empirical Risk Minimization* (ERM) principle can be applied [12,14,17,64], and the model is chosen due to the following empirical criterion

$$Q_{m,N_m}(\Phi_m) = \frac{1}{N_m} \sum_{n=1}^{N_m} L\big(\Phi_m(\mathbf{u}_n), y_n\big) \tag{5}$$

for each $m = 1, 2, \ldots, M$, $\sum_{m=1}^{M} N_m = N$.

However, it is assumed that the context is unknown and unobservable. Therefore, values of the context are unknown. Additionally, moments of the context change are unknown as well. That is why, in order to use the ERM principle, an additional mechanism for context change detection has to be applied. Otherwise, it is impossible to minimize (2) and then a substitute criterion, a so-called *prequential error* [22], has to be considered. Then, the problem is to find a sequence of models $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$ that minimizes the prequential error, that is,

$$\hat{Q}_N(\mathbf{\Phi}) = \frac{1}{N} \sum_{n=1}^{N} L\big(\Phi_n(\mathbf{u}_n), y_n\big). \tag{6}$$

To solve the problem, an algorithm with incremental learning can be proposed. There are the following propositions:

1. Algorithms with temporal batch learning for the mth context:
$$\Phi_m = G_1\big(\mathbf{U}_m, \mathbf{y}_m\big)$$
where G_1 is a learning algorithm, m denotes the context, $\mathbf{U}_m$ and $\mathbf{y}_m$ are examples for mth context.
2. Algorithms with incremental learning and shifting window:
$$\Phi_n = G_2\big(\Phi_{n-1}, \mathbf{U}_{n-L:n}, \mathbf{y}_{n-L:n}\big)$$
where G_2 is a learning algorithm, n denotes the time step, $\mathbf{U}_{n-L:n}$ and $\mathbf{y}_{n-L:n}$ are L recent observations.
3. Algorithms with incremental learning and forgetting factor:
$$\Phi_n = G_3\big(\Phi_{n-1}, \mathbf{u}_n, y_n\big)$$
where G_3 is a learning algorithm, n denotes the time step, $\mathbf{u}_n$ and y_n are recent observations.

The algorithms of the first type need an additional method for context change detection, but they solve the problem in the form (2) using the ERM principle. The algorithms of the second type update the knowledge based on the shifting window, that is, L recent examples. The algorithms of the third type use only recent example to update the knowledge with weight 1 and rest with exponential weight. Both the algorithms of second and third types solve the problem using the prequential error (6).

3.2 Rule-based representation

As previously mentioned, to support the anamnesis, the knowledge should be expressed in a way understandable easily. Therefore, in further considerations, the set of models is represented by the *attribute-value logic* [11,46]. It means that there are input and output atomic

formulae that correspond to input variables and output variable, respectively. An input atomic formula is denoted by $\alpha_k^d =$"$u^d = k$" and means: "the dth input equals k", $k \in U_d$. An output atomic formula is denoted by $\alpha_l^{out} =$"$y = l$" and means: "the output equals l", $l \in \mathcal{Y}$.

Further, the atomic formulae are connected by logical operators such as $\wedge$, $\vee$, and $\Rightarrow$. The input formulae are in 1-CNF (Conjunctive Normal Form) [37], which is a conjunction (logical operators *and*) of no more than one input formula concerning each input. Then, a decision rule is an implication:

$$\phi = \Big(\phi_{in} \Rightarrow \phi_{out}\Big), \tag{7}$$

where ϕ_{in} is the *condition*, which is the 1-CNF expression of input formulae, and ϕ_{out} is the *decision* which is a single output formula. A decision rule is denoted by ϕ.

The rule-based knowledge is a disjunction of rules:

$$\Phi = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_J, \tag{8}$$

where J is the number of rules. Such models are called D-DNF (Disjunctive Normal Form for D inputs) [37] in which expressions in 1-CNF with the decision are connected by disjunctions (logical operator *or*). Hence, the set of models is in D-DNF form.

It is also worth noting that the rule-based knowledge representation is a discriminative type of models. In contradiction to generative models, discriminative models are unable to generate both output and input values, and for a given input, they return an appropriate output [6].

Moreover, for the rule-based knowledge representation treated as a classifier, a Vapnik-Chervonenkis dimension (VC-dim) can be given [64]. For the rules expressed in the attribute-value logic the VC-dim equals [2]

$$\text{VC-dim} = \prod_{d=1}^{D} K_d. \tag{9}$$

This result entails a big *capacity* of a rule-based classifier. In other words, there is a threat of an excessive adjustment to data, so called *overfitting*. Therefore, in formulating an algorithm for rules induction, a method of regularization should be proposed.

3.3 Problem of forgetting in rule-based models

In the presence of changing hidden context, the rule-based model has to be updated and validated continuously. It is accomplished by applying forgetting mechanism and incremental learning paradigm. However, there are two issues that need to be considered. First, the rule-based models are non-parametric. In such case, the forgetting has to be conducted directly on the model structure. Removing parts of the knowledge leads to irreversible loss of information. Second, the problem of optimal set of rules induction is proved to be NP-Complete [2]. Therefore, many algorithms are insufficient from a computational perspective, especially when data streams must be processed.

A solution to both problems can be to propose a method of parameterization. However, such parameterization needs to fulfill two restrictions: (i) parameters should reflect data (*data aggregation*), (ii) rule-based model should be easily induced based on parameters. Hitherto, according to the literature, there are some propositions of parameterization, as in FLORA. Nevertheless, they do not meet mentioned restrictions.

Hence, in this paper, a new way of parameterization is proposed. The idea is to use graph structures that are parameterized. Nodes of graphs are associated with input and output

formulae, and arcs denote logical relations. Weights on graph's arcs reflect occurrences of examples. Consequently, the graphs are used for rules induction.

The idea of the approach is derived mainly from three concepts. The first approach is the OLIN algorithm, which applies graphs to represent rules [42]. The second one is the work of Georgii et al. who use a graph as a space search and an inverse search method for finding clusters [23]. Last but not least, the presented approach is strongly influenced by Pawlak's flow graphs [52].

4 Graph-based rules inducer

4.1 Preliminaries

Prior to discussing formal details, a single rule should be considered. The decision rule consists of a condition, which is in 1-CNF and a decision. Moreover, the condition is a conjunction of input formulae and the decision—an output formula. Thus, the rule can be represented as a simple graph in which edges connecting input formulae are identified with the operator $\wedge$, and an edge between any input formula and output formula means $\Rightarrow$. Since the *and* operator is commutative, the inputs can be ordered in a chosen manner.

Example 4.1 Consider an object with two inputs, $u^1 \in \{a, b\}$, $u^2 = \{1, 2\}$, and one output, $y \in \{-1, 1\}$. Thus, there are six atomic formulae, α_a^1, α_b^1, α_1^2, α_2^2, α_{-1}^{out}, α_1^{out}. Assume that the object can be described by following rules:

$$\phi_1 = \left(\alpha_a^1 \wedge \alpha_1^2 \Rightarrow \alpha_1^{out}\right), \quad \phi_2 = \left(\alpha_b^1 \Rightarrow \alpha_{-1}^{out}\right), \quad \phi_3 = \left(\alpha_2^2 \Rightarrow \alpha_{-1}^{out}\right).$$

Then, all possible rules are given in Fig. 1c, rules for the class 1—in Fig. 1a, and for the class 1—in Fig. 1b. The arcs corresponding to the *and* logical operator in a rule are dotted. The arcs between input formulae and the output formula are denoted by a double line. Then, the rule ϕ_1 is represented in Fig. 1a. The conjunction of α_a^1 and α_1^2 is the dotted arc, and the implication is the double line. However, for the rule ϕ_2, there is a direct connection between α_b^1 and an output (similarly, for the rule ϕ_3, between α_2^2 and an output), therefore the arcs denotes implications.

Hence, any rule-based model can be represented by a graph [18]

$$\mathcal{G} = (\mathcal{V}, \mathcal{A}), \tag{10}$$

where $\mathcal{V}$ denotes a set of vertices, that is, input and output formulae, $\mathcal{A}$ denotes a set of arcs.

Furthermore, each example can be seen as a most specific rule that have D conditions and a decision. Therefore, examples can be represented by graphs. Let us introduce graphs for positive examples, denoted by $\mathcal{G}_+$, and negative ones, denoted by $\mathcal{G}_-$. Both graphs have the same set of vertices but they can have different sets of arcs. Moreover, weights are associated with arcs in the positive graph, $\mathbf{w}_+$, and similarly in the negative graph, $\mathbf{w}_-$.[2] A weight of an arc denotes the number of occurrences of two conditions among positive examples for $\mathcal{G}_+$ or negative examples for $\mathcal{G}_-$. Additionally, number of negative and positive observations are denoted by w_-^{out} and w_+^{out}, respectively.

Due to commutativity of the *and* operator, it is possible to choose the order of inputs. Additionally, it is assumed that the input formulae of the given input cannot be connected by

[2] Instead of writing $\mathbf{w}_{-1}$ or $\mathbf{w}_1$ $\mathbf{w}_-$ and $\mathbf{w}_+$ will be used.

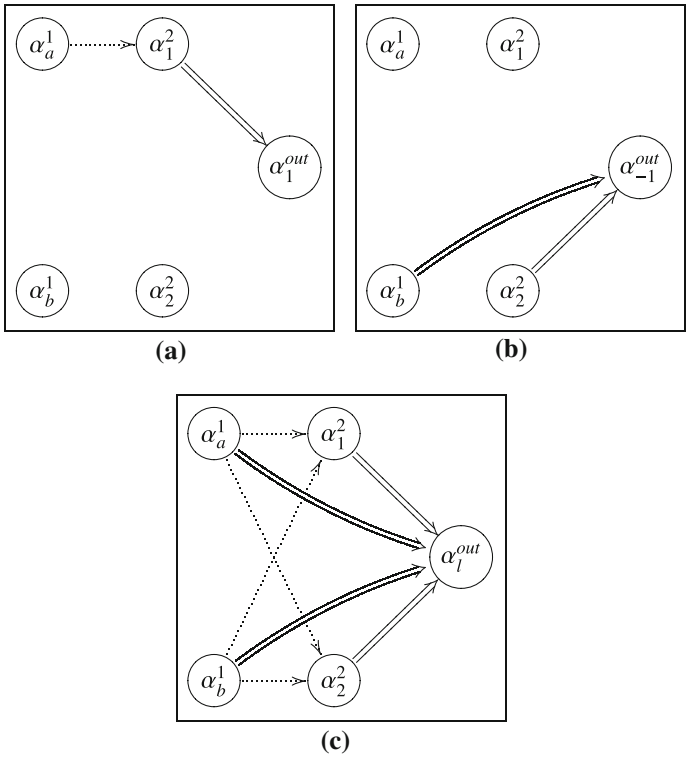

Fig. 1 Rules represented by graphs for the example 4.1. **a** Rules for the class 1. **b** Rules for the class -1. **c** All possible rules

and nor *or* operators. Hence, only the connections between two different inputs (layers) are possible. Therefore, the considered graph representing rules is layered, directed, and acyclic. Each layer consists of input formulae for one input and following order is chosen: the first layer corresponds to an input with the smallest number of values, the second—to an input in which the number of values is higher than for the first one but is smaller than the next one and so on to Dth layer. In other words inputs are ordered so that $K_{(1)} \leq K_{(2)} \leq \cdots \leq K_{(D)}$, where the lower index in parentheses denotes the ordered input number. The last layer, that is, $(D+1)$th layer, contains only an output formula.

For further simplicity, let us assume that a vertex associated with the input formula α_k^d is denoted by v_k^d. Similarly, for the output formula—v_l^{out}. The arc connecting two vertices: i in the sth layer and j in the tth layer is denoted by $a_{i,j}^{s\to t}$, for example, the arc between v_b^1 and v_{-1}^{out} is $a_{b,-1}^{1\to out}$. For each arc, a weight is associated, for positive graph $w_{+,i,j}^{s\to out}$ and for negative graph $w_{-,i,j}^{s\to out}$.

Moreover, normally the graph is represented by an adjacency matrix [18]. However, because the considered graph is layered, directed and acyclic, it is enough to keep less than K^2 weights of edges. The following lemma can be given for the number of arcs in the graph representing the rule-based model.

Lemma 4.1 *For the layered, directed, and acyclic graph representation of the rule-based model, there are* $\sum_{d=1}^{D-1} K_d \cdot \left(\sum_{\delta=d+1}^{D} K_\delta \right) + K$ *arcs.*

Proof Let us assume that dth layer consists of K_d nodes, $d = 1, 2, \ldots, D$. Then, for each node in the first layer, there are arcs to all nodes in each layer $d > 1$ and to the final node. Thus, for the first layer, there are $K_1 \sum_{\delta=2}^{D} K_\delta + K_1$ of arcs. Similarly, for the second layer, there are $K_2 \sum_{\delta=3}^{D} K_\delta + K_2$ of arcs because no arcs to the first layer are allowed. Analogously, the number of arcs can be given for the rest of layers for $d < D$. For the last layer, there are only arcs to the final node, meaning there are only K_D arcs.

Hence, there are $\sum_{d=1}^{D-1} K_d \Big(\sum_{\delta=d+1}^{D} K_\delta + 1 \Big) + K_D$ of arcs. This expression can be simplified into the following form

$$\begin{aligned}\sum_{d=1}^{D-1} K_d \Big(\sum_{\delta=d+1}^{D} K_\delta + 1 \Big) + K_D &= \sum_{d=1}^{D-1} K_d \Big(\sum_{\delta=d+1}^{D} K_\delta \Big) + (K - K_D) + K_D \\ &= \sum_{d=1}^{D-1} K_d \Big(\sum_{\delta=d+1}^{D} K_\delta \Big) + K.\end{aligned}$$

□

The Lemma 4.1 shows that instead of K^2 entities in the adjacency matrix there are only

$$\kappa = \sum_{d=1}^{D-1} K_d \cdot \Big(\sum_{\delta=d+1}^{D} K_\delta \Big) + K$$

arcs needed to be kept. Therefore, each graph representing a rule-based model is coded using κ parameters as follows:

$$\text{code}(\mathcal{G}) = \Big[\text{code}(layer_1) \;\; \text{code}(layer_2) \ldots \text{code}(layer_D) \Big]. \tag{11}$$

where $\text{code}(layer_d)$ is a code (0-1 sequence) for single layer, for $d = 1, 2, \ldots, D$,

$$\begin{aligned}&\text{code}(layer_d) = \\ &\Big[a_{1,1}^{d\to d+1} \;\; a_{1,2}^{d\to d+1} \ldots a_{1,k_{d+1}}^{d\to d+1} \;\; a_{1,1}^{d\to d+2} \ldots a_{1,k_{d+2}}^{d\to d+2} \ldots a_{1,1}^{d\to D} \ldots a_{1,k_D}^{d\to D} \;\; a_{1,l}^{d\to out} \\ &a_{2,1}^{d\to d+1} \;\; a_{2,2}^{d\to d+1} \ldots a_{2,k_{d+1}}^{d\to d+1} \;\; a_{2,1}^{d\to d+2} \ldots a_{2,k_{d+2}}^{d\to d+2} \ldots a_{2,1}^{d\to D} \ldots a_{2,k_D}^{d\to D} \;\; a_{2,l}^{d\to out} \\ &\ldots \\ &a_{k_d,1}^{d\to d+1} \;\; a_{k_d,2}^{d\to d+1} \ldots a_{k_d,k_{d+1}}^{d\to d+1} \;\; a_{k_d,1}^{d\to d+2} \ldots a_{k_d,k_{d+2}}^{d\to d+2} \ldots a_{k_d,1}^{d\to D} \ldots a_{k_d,k_D}^{d\to D} \;\; a_{k_d,l}^{d\to out} \Big]\end{aligned}$$

For example, the graphs from the Fig. 1a, b are the following:

$$\begin{aligned}\text{code}(\mathcal{G}_+) &= [10000010], \\ \text{code}(\mathcal{G}_-) &= [00000101].\end{aligned}$$

For instance, the first code states that the atomic formula α_a^1 is connected with α_1^2 and not with α_2^2 and α_1^{out}; the formula α_b^1 is connected with no formulae, and from the second layer, there exists only the connection between α_1^2 and α_1^{out}.

4.2 Learning and rules induction

As it was mentioned before, each example can be seen as the most specific rule. Thus, an example can be coded using (11). For instance, as in Example 4.1, for an observation in the form $\mathbf{u} = (\alpha_a^1 \wedge \alpha_1^2)$, the code is $code(\mathbf{u}) = [10000010]$. Moreover, as described in Sect. 4.1,

weights in the positive and negative graphs are associated with numbers of occurrences of all pairs of inputs and an output in the example. Thus, having a training set, it is possible to calculate weights incrementally and independently on examples order in the training sequence. If the nth example is $(\mathbf{u}_n, y_n)$, then the weights are updated as follows:

$$\mathbf{w}_{y_n} := \mathbf{w}_{y_n} + \text{code}(\mathbf{u}_n) \tag{12}$$

$$w_{y_n}^{out} := w_{y_n}^{out} + 1. \tag{13}$$

It can be done is such a way as the codes for weights and the example are of the same length and reflect the same inputs. Besides, it is worth noting that the weights aggregate data in the incremental manner. Moreover, having weights values, it is possible to restore the whole training set.[3]

Example 4.2 For the object as in Example 4.1, data can be as follows

$$\Big\{\big([a\ \ 1], 1\big), \big([a\ \ 2], -1\big), \big([b\ \ 1], -1\big), \big([b\ \ 2], -1\big)\Big\}$$

Then, weights vectors are the following

$$\begin{aligned} \mathbf{w}_+ &= [10100010], \\ \mathbf{w}_- &= [01111212], \\ w_+^{out} &= 1, \\ w_-^{out} &= 3. \end{aligned}$$

However, positive and negative graphs reflect only aggregated examples and they can be barely, if at all, called rule-based knowledge. Prior to describing rule induction basing on both graphs, a manner of evaluating pairs of vertices and ultimately—paths, that is, rules, should be proposed.

First, we propose a criterion for evaluating pairs of atomic formulae, and then—paths with the end in the final vertex, that is, rules. In the machine learning literature for the rules induction *coverage* and *accuracy*[4] measures are used [55]. The *coverage* measure says about the generality of the rule while the *accuracy* measure expresses the specialization of the rule. For example, the rule $\alpha_a^1 \Rightarrow \alpha_1^{out}$ is more general than $\alpha_a^1 \wedge \alpha_1^2 \Rightarrow \alpha_1^{out}$.

Let $E_{\phi_{in}}$ denote a set of all examples covered by the condition of a rule ϕ_{in}, E_l—a set of all examples in the class $y = l$. Then the coverage measure is defined as follows

$$\mu_c(\phi_{in}, l) = \frac{\text{card}\{E_{\phi_{in}} \cap E_l\}}{\text{card}\{E_l\}}. \tag{14}$$

The accuracy measure is expressed in the following way

$$\mu_a(\phi_{in}, l) = \frac{\text{card}\{E_{\phi_{in}} \cap E_l\}}{\text{card}\{E_{\phi_{in}}\}}. \tag{15}$$

When above expressions are undetermined, then they are set to zero.

It is worth realizing that *accuracy* can be determined by *coverage* [52]

$$\mu_a(\phi_{in}, l) = \frac{\mu_c(\phi_{in}, l)\ \text{card}\{E_l\}}{\mu_c(\phi_{in}, -1)\ \text{card}\{E_-\} + \mu_c(\phi_{in}, 1)\ \text{card}\{E_+\}}. \tag{16}$$

[3] Due to lack of space the procedure is not presented here. However, the general idea is to go from the first node in the first layer through all layers, passing one vertex in a layer only, to the output vertex and decrease the value of weights on arcs (only if it is nonzero). The procedure is repeated for all vertices in the first layer. At the end, a set of most specific rules, i.e., examples, is obtained.

[4] Sometimes referred to as a *confidence* measure [30] or *certainty* measure [52].

Moreover, it is worth noting that for any pair of input formulae A and B, μ_c is anti-monotonic, that is,

$$\mu_c(A, y) \geq \mu_c(A \wedge B, y), \tag{17}$$

and μ_a is monotonic, that is,

$$\mu_a(A, y) \leq \mu_a(A \wedge B, y). \tag{18}$$

Both properties follow from the definitions. The set of covered examples by single formula A is the same or larger than for $A \wedge B$, that is, $\text{card}\{E_A\} \geq \text{card}\{E_{A\wedge B}\}$. Hence, by adding a formula in *coverage*, the nominator can decrease while the denominator is constant, so the value of *coverage* can decrease or remain the same. In *accuracy*, however, both the nominator and the denominator can decrease. But the denominator decreases at a faster pace than the nominator because $\text{card}\{E_{A\wedge B} \cap E_y\} \leq \text{card}\{E_{A\wedge B}\}$, and thus the value of *accuracy* can increase or remain unchanged.

However, *coverage* and *accuracy* are only suitable to measure the generalization and specialization of a rule separately. But the goal in the rules induction is to obtain the knowledge that is the most accurate generalization of examples [47]. Hence, to reach a balance between generalization and specialization in a synthetic criterion, following convex combination of (14) and (15) can be proposed

$$q(\phi, l) = \beta\ \mu_c(\phi_{in}, l) + (1 - \beta)\ \mu_a(\phi_{in}, l), \tag{19}$$

where $\beta \in [0, 1]$, which determines the weight of balance between the generality of rules, expressed by μ_c, and their specificity, expressed by μ_a.

Consequently, having weights for both positive and negative graphs, the *coverage* and *accuracy* can be calculated for a single arc $a_{i,j}^{s\to t}$ in the class $y = l$. Denoting by $w_{l,i,j}^{s\to t}$ a weight for the arc $a_{i,j}^{s\to t}$ in the class $y = l$, and w_l^{out} – the number of occurrences of the class $y = l$, the coverage of an arc is defined as follows

$$\mu_c(a_{i,j}^{s\to t}, l) = \frac{w_{l,i,j}^{s\to t}}{w_l^{out}}. \tag{20}$$

Then, applying (16), we get accuracy of an arc in the form

$$\mu_a(a_{i,j}^{s\to t}, l) = \frac{\mu_c(a_{i,j}^{s\to t}, l)\ w_l^{out}}{\mu_c(a_{i,j}^{s\to t}, -1)\ w_-^{out} + \mu_c(a_{i,j}^{s\to t}, 1)\ w_+^{out}}. \tag{21}$$

Next, it is important to evaluate the quality of a path which is an equivalence of evaluating a rule. The path π of a length no greater than D is a sequence of distinct vertices with the beginning in v_k^s and the end in v_l^{out} such that from each of its vertices there is an arc to the next vertex in the sequence. Obviously, each vertex in the path belongs to distinct layers. To calculate the quality criterion of the path, the anti-monotonicity of the μ_c is used, that is,

$$\mu_c(\pi, l) = \min_{a\in\pi}\{\mu_c(a, l)\}. \tag{22}$$

Then, having the coverage of the path, the accuracy can be calculated using (16), that is,

$$\mu_a(\pi, l) = \frac{\mu_c(\pi, l)\ w_l^{out}}{\mu_c(\pi, -1)\ w_-^{out} + \mu_c(\pi, 1)\ w_+^{out}}. \tag{23}$$

Thus, having negative and positive weights and given β, it is possible to calculate criterion (19) for any path in the graph. Now, all rules with the value of the criterion (19) larger than

0, that is, $q(\phi, l) > 0$ can be generated. However, in this approach, the total number 2^K of rules has to be considered and the knowledge, even if the rules are ordered according to the quality criterion, is rather useless. Therefore, the search space should be limited.

The idea of limiting the search space is to create a new graph in which an arc is either negative or positive. The rule is formulated only if all the arcs in the path are either positive or negative. First, the negative and positive graphs are calculated, that is, weights $\mathbf{w}_-$, $\mathbf{w}_+$, w_-^{out}, w_+^{out}. Next, for each arc, the quality criterion (19) is calculated in positive and negative graphs. It results in obtaining different weights for negative and positive graphs, denoted by $\mathbf{q}_-$, $\mathbf{q}_+$, respectively. Finally, a graph that determines a search space is computed as a difference between $\mathbf{q}_+$ and $\mathbf{q}_-$. The difference is denoted by $\mathbf{q}$. Then, all arcs in graph coded by $\mathbf{q}$ are either positive or negative. Each path that has only positive or negative weights is regarded as an admissible rule.

The procedure for determining the search space is described in Algorithm 4.1.

Algorithm 4.1 Determination of the search space.

Input: (i) examples, (ii) β, (iii) quality criterion (19), (iv) *coverage* (20), (v) *accuracy* (21).
Output: The graph that defines the search space.

Step 1: Calculate $\mathbf{w}_-$, $\mathbf{w}_+$, w_-^{out}, and w_+^{out} based on examples $\left\{\left(\mathbf{u}_n, y_n\right)\right\}_{n=1}^{N}$.
Step 2: For given β, for each arc in graphs $\mathcal{G}_+$ and $\mathcal{G}_-$ calculate the quality criterion (19) using (20), and (21). Denote those weights by $\mathbf{q}_+$ and $\mathbf{q}_-$ for $\mathcal{G}_+$ and $\mathcal{G}_-$, respectively. (Both vectors are of size κ).
Step 3: Calculate the difference between $\mathbf{q}_+$ and $\mathbf{q}_-$, that is:

$$\mathbf{q} = \mathbf{q}_+ - \mathbf{q}_-. \tag{24}$$

The rationale behind (24) is to classify an arc to one and only one output value. The graph represented by the code $\mathbf{q}$ contains the final output vertex that is neither positive nor negative.

Hence, having the search space defined by $\mathbf{q}$, it is clear that not all paths are allowed. Some paths can contain arcs that are positive and negative as well. Additionally, we are interested only in rules for which quality criterion is greater than a given value θ, that is, $q(\phi, l) \geq \theta$, where $\theta \in [0, 1)$ is a threshold. Hence, the final algorithm for rules induction, called ***G**raph-based **R**ules **I**nducer* (GRI), can be proposed. It is worth noting that the procedure starts from the final layer, that is, $(D+1)$th layer, and then proceeds to the first one. $\mathcal{P}_+$ and $\mathcal{P}_-$ denote the sets of all possible paths built only from positive and negative arcs, respectively.

Algorithm 4.2 Graph-based Rules Inducer.

Input: (i) examples, (ii) β, (iii) quality criterion (19), (iv) *coverage* (20), (v) *accuracy* (21), (vi) θ, (vii) D – number of input variables.
Output: The rule-based model Φ.

Step 1: Having data and fixed β apply the Algorithm 4.1 to obtain $\mathbf{q}$. Set $\mathcal{P}_+ = \emptyset$, $\mathcal{P}_- = \emptyset$. Set $d := D$.
Step 2: For all $i = 1, \ldots, K_d$ consider the path $\pi = (a_{i,y}^{d \to out})$. If $q_{i,y}^{d \to out} > 0$, then $\mathcal{P}_+ := \mathcal{P}_+ \cup \{\pi\}$. Else if $q_{i,y}^{d \to out} < 0$, then $\mathcal{P}_- := \mathcal{P}_- \cup \{\pi\}$.
Step 3: For each $\pi \in \mathcal{P}_+$ such that $\pi = (a_{j,\cdot}^{b \to \cdot} \ldots)$ and for all $i = 1, \ldots, K_d$ consider the path $\overline{\pi} = (a_{i,j}^{d \to b}\, a_{j,\cdot}^{b \to \cdot} \ldots)$. If $q_{i,j}^{d \to b} > 0$, then $\mathcal{P}_+ := \mathcal{P}_+ \cup \{\overline{\pi}\}$. For

each $\pi \in \mathcal{P}_-$ such that $\pi = (a_{j,\cdot}^{b\rightarrow\cdot} \dots)$ and for all $i = 1, \dots, K_d$ consider the path $\overline{\pi} = (a_{i,j}^{d\rightarrow b}\, a_{j,\cdot}^{b\rightarrow\cdot} \dots)$. If $q_{i,j}^{d\rightarrow b} < 0$, then $\mathcal{P}_- := \mathcal{P}_- \cup \{\overline{\pi}\}$.
Step 4: If $d > 1$, then $d := d - 1$ and go to **Step 2**.
Step 5: For each $\pi \in \mathcal{P}_- \cup \mathcal{P}_+$. If $q(\pi, 1) > \theta$ or $q(\pi, -1) > \theta$, then create the rule ϕ coupled with the path π and update the model $\Phi := \Phi \vee \phi$.

In Algorithm 4.2, in the first step, the graph determining the search space is obtained. In order to create only the paths that are coupled with a proper rule, that is, the sign of the path, the algorithm runs backward, that is, from the final vertex to the first layer. Despite the fact that the number of paths in both sets might grow exponentially with respect to the number of features, in many practical cases the number of paths seems to be reasonable. Nevertheless, formulating a rule-based model in problems with many inputs might be intractable, therefore some heuristics should be proposed. On the other hand, in many cases, for example, in diabetics treatment where human health and life is crucial, the accuracy of the rules should be as high as possible. Then, all possible paths ought to be checked and evaluated.

4.3 Remarks

1. Graphs $\mathcal{G}_-$ and $\mathcal{G}_+$ can be regarded as a special kind of Pawlak's flow graphs. However, in presented approach, each vertex is connected with a single atomic formula, not with a conjunction of atomic formulae like in the original flow graphs. Nevertheless, similarly to Pawlak's flow graphs, the weights associated with arcs of positive and negative graphs can be deemed as information flow between layers. The amount of information inflow of the dth layer in the class $y = l$, $f_{in}(d, l)$, equals the amount of information outflow from the layer, $f_{out}(d, l)$, that is,

$$f_{in}(d, l) = \frac{1}{d} \sum_{s=1}^{d-1} \sum_{i=1}^{K_s} \sum_{j=1}^{K_d} w_{l,i,j}^{s\rightarrow d}, \tag{25}$$

$$f_{out}(d, l) = \frac{1}{D+1-d} \sum_{t=d+1}^{D+1} \sum_{i=1}^{K_d} \sum_{j=1}^{K_t} w_{l,i,j}^{d\rightarrow t}. \tag{26}$$

   Because of the way of the weights updating (12), it can be checked that $f_{in}(d, l) = f_{out}(d, l)$.
   Equations (25) and (26) can be considered as flow conservation equations [53].
2. The limitation of the search space applied in the GRI algorithm matters in two ways. First, only either positive or negative paths are allowed. Second, not all models can be obtained from such a structure. To provide evidence, let us refer to the example in Fig. 2. There are two inputs with three values each, that is, $\alpha_a^1, \alpha_b^1, \alpha_c^1$, and $\alpha_1^2, \alpha_2^2, \alpha_3^2$. The class 1 is depicted by the gray color, and the class -1 by the white color (gray and white rectangles in Fig. 2a). However, Algorithm 4.1 does not allow to obtain both rules $\alpha_b^1 \wedge \alpha_2^2 \Rightarrow \alpha_1^{out}$ and $\alpha_a^1 \wedge \alpha_2^2 \Rightarrow \alpha_{-1}^{out}$ because it would mean that the arc in the graph $\mathbf{q}$ between v_2^2 and the final node v_y^{out} is positive and negative at the same time. Thus, depending on data, knowledge can be in a similar form to Fig. 2b. Therefore, the limitation of the search space results in the model robustness to overfitting. However, if the true object description is as in Fig. 2a, then the obtained knowledge is less accurate. Nevertheless, in the case of changing context, the robustness is usually at the expense of accuracy.

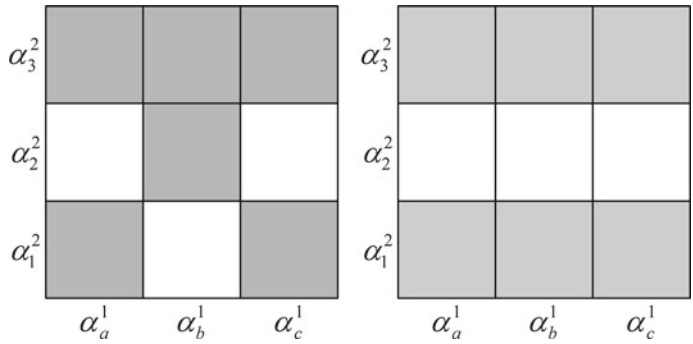

Fig. 2 Example of applying the Algorithm 4.1 in the GRI algorithm. **a** Original description. **b** Description after applying the Algorithm 4.1

3. The graph coded by **q** can be used as a classifier itself. The procedure for classification of a new example is as follows. First, all possible subpaths of the example (note: example is treated as a path) should be found.[5] However, only subpaths that are either positive or negative are considered. Second, the subpath with the highest quality value is chosen. The sign of that subpath is returned as the class. In case, the GRI algorithm is used as a classifier it will be referred as the GRI classifier.
   In the case when only the class is needed, this simple procedure enables classification of an up-coming observation usually very fast.

5 Graph-based rules inducer with forgetting

The presented form of the GRI algorithm is a knowledge extraction method with the incremental learning as the weights are updated step-by-step. Moreover, the weights $\mathbf{w}_-$ and $\mathbf{w}_+$ reflect data aggregation and thus examples can be restored at any time. However, the main goal of this paper is to update and validate knowledge according to the changing context. To follow context changes, a forgetting mechanism has to be proposed. The first idea is to apply shifting window, but it would require the calculation of weights values to be done constantly. The second proposition is the forgetting factor (exponential forgetting). Applying the forgetting factor, $\gamma \in [0, 1]$, in negative and positive graphs would result in re-calculating weights values once only, before the updating phase. Then, if a new example emerges, $(\mathbf{u}_{n+1}, y_{n+1})$, the updating formulas (12) and (13) can be re-written in the following way. If $y_{n+1} = 1$, then

$$\mathbf{w}_+ := \gamma \mathbf{w}_+ + \text{code}(\mathbf{u}_{n+1}), \tag{27}$$
$$\mathbf{w}_- := \gamma \mathbf{w}_-, \tag{28}$$
$$w_+^{out} := \gamma w_+^{out} + 1, \tag{29}$$
$$w_-^{out} := \gamma w_-^{out}, \tag{30}$$

or if $y_{n+1} = -1$, then

[5] For an observation of the length D there are $2^D - 1$ of all subpaths.

$$\mathbf{w}_+ := \gamma \mathbf{w}_+, \tag{31}$$

$$\mathbf{w}_- := \gamma \mathbf{w}_- + \text{code}(\mathbf{u}_{n+1}), \tag{32}$$

$$w_+^{out} := \gamma w_+^{out}, \tag{33}$$

$$w_-^{out} := \gamma w_-^{out} + 1. \tag{34}$$

The forgetting factor γ is usually very close to 1 meaning that such an approach can be seen as a weighted shifting window of the size approximately equal $3/(1-\gamma)$ [8]. All examples that are prior to the shifting window, that is, $n < 3/(1-\gamma)$, influence the model with the weight smaller than 0.05.

Obviously, applying the forgetting factor precludes restoration of the original data from the graphs $\mathcal{G}_+$ and $\mathcal{G}_-$. However, it is a price that has to be paid for the context tracking.

Hence, to make Algorithms 4.1 and 4.2 capable of the context tracking, the exponential forgetting should be applied to determination of the search space.

Algorithm 5.1 Determination of the search space with forgetting.

Input: (i) data stream, (ii) β, (iii) quality criterion (19), (iv) *coverage* (20), (v) *accuracy* (21), (vi) γ, (vii) $\mathbf{w}_+$, $\mathbf{w}_-$, w_+^{out}, w_-^{out} – null vectors.
Output: The graph that defines the search space.

Step 0: Set $n := 0$.
Step 1: Set $n := n + 1$, take an example from the data stream, $\mathbf{u}_n$, y_n. If the example is misclassified and $y_n = 1$, then update weights using (27), and (28). If the example is misclassified and $y_n = -1$, then update weights using (31), and (32). Otherwise update weights using (12) and (13).
Step 2: For a given value of β, for each arc in graphs $\mathcal{G}_+$ and $\mathcal{G}_-$ calculate the quality criterion (19) using (20), and (21). Denote those weights by $\mathbf{q}_+$ and $\mathbf{q}_-$ for $\mathcal{G}_+$ and $\mathcal{G}_-$, respectively. (Both vectors are of size κ).
Step 3: Calculate the difference between $\mathbf{q}_+$ and $\mathbf{q}_-$, that is (24).

The forgetting in the Step 1 can be applied after each observation, not only if the misclassification occurs.

In Algorithm 4.2, the rules induction is performed based on the calculated $\mathbf{q}$. It does not matter if in the first step of Algorithm 4.2 is Algorithm 4.1 or Algorithm 5.1. The only difference is that the first approach is without forgetting (see Fig. 3a and the second one—with a forgetting mechanism and rules are induced only if the procedure is set to be run (see Fig. 3b, e.g., after a fixed number of iterations. Both versions of the GRI algorithm are schematically presented in the Fig. 3. It is worth remembering that in Algorithms 4.1 and 5.1, the weights updating is performed in an incremental manner.

Moreover, the rules induction does not necessarily have to be conducted at each time step. For instance, in the diabetes treatment, the rules induction is applied only if the physician or patients wants it, and not after each measurement.

The remarks given for the GRI algorithm (see Sect. 4.3) holds true for the GRI algorithm with forgetting.

6 Experimental results

To evaluate the GRI algorithm with forgetting for data stream in the presence of changing context, two experiments were conducted. The first one involves thirteen other methods and

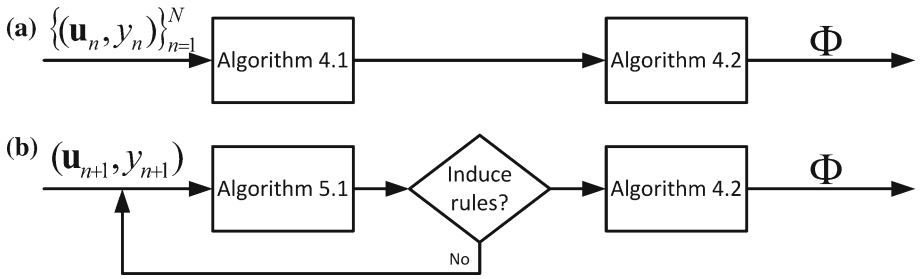

Fig. 3 Schemes for the GRI algorithm with limiting search space **a** without forgetting, **b** with forgetting

a benchmark data set *Electricity* [28]. The second one is the application in the diabetes treatment and involves five other methods and a data set collected by Michael Kahn, M.D., Ph.D., and published in the UCI Machine Learning Repository [63].

6.1 Electricity

The data used in the experiment were first described by Harries [28]. The data set was prepared basing on the observations collected from the Australian New South Wales (NSW) Electricity Market. In this market, the prices are not fixed and they are affected by the demand and the supply of the market. There are several factor influencing the market demand such as weather, time of day, district population density, market expansion. In other words, this domain is known to exhibit seasonality and sensitivity to short-term events, for example, weather changes. Therefore, all influences can be treated as a hidden context that affects the market.

The *Electricity* data set (referred to $ELEC2$ in the literature [28]) contains 45312 examples dated from May 1996 to December 1998. An example consists of following inputs: u^1—day of a week, u^2—period of a day, u^3—NSW price, u^4—NSW demand, u^5—Victoria region price, u^6—Victoria region demand, and u^7—the scheduled electricity transfer between states. Because inputs 3–7 were numeric, the discretization was applied. Finally, there were following number of input values: $K_1 = 7$, $K_2 = 48$, $K_3 = 10$, $K_4 = 6$, $K_5 = 10$, $K_6 = 7$, and $K_7 = 7$ (all values $K = 95$).

The output value identifies changes of the price related to a moving average of the last 24 hours. The class reflects deviations of the price on a one day average and removes the impact of longer term price trends. There are two possible output values: UP ($y = 1$), and DOWN ($y = -1$).

In the experiment, the following methods were compared with the GRI algorithm:

- IB1 (also known as 1-NN) with DDM (**D**rift **D**etection **M**ethod) [21] or EDDM (**E**arly **D**rift **D**etection **M**ethod) [5]—is a lazy classifier (the nearest neighbor) with a method for context change detection;
- J48 with DDM or EDDM—it is an implementation of tree-based classifier called C4.5 [57] with a method for context change detection;
- DWM-NB (**D**ynamic **W**eighted **M**ajority of **N**aïve **B**ayes) [39]—is an ensemble method of Naïve Bayes classifiers that uses the idea of weighted majority;
- PL-NB (**P**aired **L**earner of **N**aïve **B**ayes) [4]—is a special kind of an ensemble method of Naïve Bayes classifiers that uses two classifiers: one learns from all data, and second learns from data maintained in a shifting window;

- SWIM (**S**hifting **WIN**NOW) [3]—is a classifier that implements a Boolean threshold function (based on the WINNOW algorithm) but with shifting window;
- NB (**N**aïve **B**ayes) classifier with exponential forgetting [62]—an estimation of probabilities is made based on the frequency matrix that aggregates data and forgetting is conducted on that matrix;
- AQ-P1 [61]—is the AQ algorithm with shifting window;
- AQ-P2 [61]—is the AQ algorithm with shifting window and implicit forgetting on a model;
- CART [9] with shifting window—is a classification tree that is induced from examples maintained in a shifting window;
- Random Forest [10] with shifting window—it is an ensemble classifier of classification trees that is learned from examples maintained in a shifting window;
- SVM [64] with shifting window—is a Support Vector Machine classifier that is obtained from examples maintained in a shifting window; the following kernel function was applied [6]:

$$k(u, v) = 2^{\text{card}\{u \cap v\}} \tag{35}$$

where $\text{card}\{u \cap v\}$—number of conditions shared by rules u and v.

It is worth noting that only J48, CART, AQ-P1 and AQ-P2 are rule-based representations.

All methods are evaluated using the (6) criterion. Furthermore, in all cases, the learning is conducted on the stream of data (an example arrives according to the original time stamp). A new arriving example is first classified and then used for learning (after classification true output is given).

Due to authors knowledge, the methods SWIM, NB with forgetting, AQ-P1, AQ-P2, CART, Random Forest, and SVM have not been yet evaluated on the *Electricity* data set. All of mentioned algorithms, as well as the GRI, were implemented in Matlab® environment. Results for the others methods are taken from the literature specified in Table 1.

Only the output value is important in this task therefore the GRI was used as a classifier (the procedure described in the Sect. 4.3).

6.1.1 Results and discussion

The *Electricity* data set contains real-life data and the number of data is enough, from statistical perspective, to compare different methods. The results are presented in Table 1. The best performance is achieved by SWIM, which is better than the GRI by approximately 0.01. The second is GRI with criterion value of 0.12 and is comparable with the Random Forest. The rule-based algorithms are 0.13 for CART, 0.16 for J48 with EDDM, and around 0.19 for AQ-P1 and AQ-P2. The performance values for all methods vary from 0.11 to 0.23.

However, the performance of the GRI algorithm is very promising. The results indicate that the GRI algorithm can extract very accurate knowledge. Especially that it outperforms the AQs algorithms by around 0.06, the J48 with EDDM by approximately 0.03, and CART by about 0.01.

The best performance of the GRI classifier is achieved for $\gamma = 0.88$ and $\beta = 0.1$, which is quite an interesting result. The γ parameter arbitrates weights of the examples and quantifies examples for further consideration in knowledge extraction process. It is important to notice that the result of the GRI without forgetting, that is, $\gamma = 1$, is much poorer than for $\gamma = 0.88$ (the difference is around 0.13 in favor of $\gamma = 0.88$, see Table 2). The β parameter determines the balance between choosing general or specialized rules. In other words, the more rules are

Table 1 Results for methods compared in the experiment for the criterion (6) in the electricity price market

Method	Rule-based	Criterion (6)	Source
IB1 with DDM	N	0.23	[21]
IB1 with EDDM	N	0.14	[5]
J48 with DDM	I	0.21	[21]
J48 with EDDM	I	0.16	[5]
DWM-NB	N	0.17	[39]
PL-NB	N	0.19	[4]
SWIM	N	**0.11**	[3]
NB with forgetting	N	0.17	[62]
AQ-P1	Y	0.19	[61]
AQ-P2	Y	0.19	[61]
CART with shifting window	I	0.13	–
Random Forest with shifting window	N	**0.12**	–
SVM with shifting window	N	0.14	–
GRI	Y	**0.12**	–

Each method contains additional information whether it has rule-based representation (Y-yes, N-no, I-can be interpreted as rules). Best three results in bold

Table 2 Results for the GRI algorithm with different values of parameters, and for the criterion (6) in the electricity price market

γ	$\beta = 0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$
1.00	0.25	0.25	0.25	0.28	0.41
0.99	0.20	0.17	0.17	0.19	0.28
0.95	0.18	0.13	0.14	0.16	0.20
0.90	0.18	0.13	0.14	0.15	0.17
0.89	0.21	0.12	0.13	0.14	0.17
0.88	0.21	**0.12**	0.13	0.14	0.16
0.87	0.22	0.12	0.13	0.14	0.16
0.86	0.22	0.13	0.14	0.14	0.17

Best result in bold

specialized, the broader knowledge describes single situations. In this case, a small value of β can be explained by the domain specificity. The electricity price market is dependent on short-term events and seasonality. For instance, in the winter days are shorter and electricity demand is higher so these conditions determine specific situations that influence prices fluctuations.

6.2 Supporting anamnesis in diabetes treatment

To evaluate the GRI algorithm for the anamnesis module in the *e-Health* system for the diabetes treatment, a real-life data set is used [63]. The data was collected by Michael Kahn, M.D., Ph.D. and covers 70 patients. Diabetes patient records were obtained from two sources: an automatic electronic recording device and paper records. The automatic device had an inter-

nal clock to time stamp events, whereas the paper records only provided "logical time" slots (breakfast, lunch, dinner, bedtime) provided by a patient (e.g., breakfast is 6:30, or 7:35). Each patient's medical history corresponds to a period from 20 to 149 days of measurements, depending on a patient.

Original diabetes files consist of four information per record: (i) date, (ii) time, (iii) code (nominal), (iv) glucose level in blood (numeric). The code describes the measurement, for example, regular insulin dose, pre-lunch glucose measurement, typical meal ingestion, typical exercise activity, and others (details can be found in [63]).

However, the aim of the anamnesis is to find the hidden context. In other words, to provide such knowledge that could help a physician in formulating adequate questions. For instance, if the patient glucose level is always bad on a Saturday morning, this might do an indication that his/her habits may influence the glucose level in blood.

Therefore, the original inputs were transformed into the following attributes: u^1—day of week (Monday, Tuesday, and so on), u^2—part of a day (from 4:00 until 10:00, from 10:00 until 16:00, from 16:00 until 22:00, and from 22:00 until 4:00), u^3 – measurement code (before a meal, after a meal, after an insulin dose, other). The number of values for inputs are following: $K_1 = 7$, $K_2 = 4$, $K_3 = 5$. Besides, the glucose level in blood was transformed into the output that describes whether the glucose level is in norm ($y = 1$), or not ($y = -1$). The output is determined basing on the original code, for example, pre-lunch glucose measurement, and the glucose level in blood. For instance, the allowed glucose level in blood is different before a meal (80–120 mg/dl) and after a meal (80–140 mg/dl). Details can be found in [63]. Having these inputs, a physician is able to identify, for example, the periods during which the glucose is not in norm or what the glucose level is before and after meals, and then inquire into the reasons of such state.

Diabetes is an illness, which is caused not only by the insulin production problems but also depends on other factors like psychological tension, feeding and drinking habits, sport activities, health condition. All of these factors can be treated as the hidden context. Moreover, the diabetic condition evolves in time and the results of treatment are rather noticeable in a long term. Therefore, in the experiment, only 10 out of 70 patient records were used from which the smallest number of examples was 926 (116 days), and the biggest number was 1,327 (149 days).

To evaluate the GRI algorithm in comparison with other rule-based inducers (AQ-P1, AQ-P2, and CART with shifting window), and well-known classifiers like Random Forest and SVM with kernel function (35) and shifting window, the criterion (6) was used. In a real-life application, a physician uses the extracted knowledge. However, in the experiment, it would be methodologically incorrect to evaluate the results based on the physician's opinions. Hence, it is assumed that if the algorithm is able to track the context changes, it could be argued that the *usefulness* of the knowledge is appropriate for medical interview. Furthermore, similarly to the *Electricity* data set, in all cases, the learning is conducted on the stream of data (an example arrives according to the original time stamp). A new example is first classified and then used for learning (after classification the true output is given).

All algorithms were checked with different parameters values, and results for the highest performing algorithms are presented in Table 3. The results of mean prequential error of 10 patients for the GRI with different γ and β values are provided in Table 4.

Table 3 The results for the GRI, AQ-P1, AQ-P2, CART, random forest, and SVM, for the criterion (6) in the diabetes treatment

	GRI	AQ-P1	AQ-P2	CART	Random forest	SVM
Mean	**0.10**	0.15	0.16	0.14	0.13	0.13
Worst case	**0.16**	0.21	0.22	0.20	0.19	0.18
Best case	**0.05**	0.10	0.11	0.09	0.09	0.09
SD	0.04	0.04	0.03	0.04	0.03	0.03

Best results in bold

6.2.1 Results and discussion

First of all, the obtained results indicate that applying knowledge in the form of rules, that is, GRI, AQs, and CART, for diabetes is satisfactory (see Table 3). The mean value of criterion (6) is at the level of 0.15 and 0.16 for AQ-P1 and AQ-P2, respectively, while CART even 0.14. The GRI classifier performed better by about 0.05, meaning the criterion is at the level of 0.10. Moreover, the GRI classifier outperformed Random Forest and SVM by about 0.03.

Nevertheless, because only 10 patients were considered, it is worth checking whether the differences between the GRI and other algorithms are significant. Therefore, a statistical hypothesis testing is applied. For that purpose, the following methodology was considered:

1. Verify if the obtained results for all patients are drawn from a normal distribution.
2. Verify if the standard deviations (or variances) of all algorithms are the same.
3. Verify if the mean value of the GRI is worse than for others methods.

To verify if the methods are statistically comparable, the one-tailed Student's t-test is used. The null hypothesis is as follows: $H_0 : \mu < \mu_{\text{GRI}}$, where μ is a mean value of an algorithm other than the GRI, and μ_{GRI}—a mean value of the GRI. However, the Student's t-test can be used only if observations are drawn from the normal distribution, and variances are equal. Therefore, to solve the first problem, the Kolmogorov-Smirnov test is applied. For the second problem the F-statistics is used. All test were conducted in the Matlab® environment. Hence:

1. According to the Kolmogorov-Smirnov test at the significance level of 0.05 the results are drawn from the normal distribution (all p-values were smaller than 0.005).
2. The value of F statistics for 10 degrees of freedom at the significance level 0.05 equals $F(0.05, 9, 9) = 3.18$. If $\frac{1}{F} < f < F$, where f is a fraction of two variances, then both variances can be assumed to be the same. The f values are as follows:

$$\begin{aligned} f_{\text{GRI,AQ-P1}} &= 1.16 \in [0.31, 3.18], \\ f_{\text{GRI,AQ-P2}} &= 0.99 \in [0.31, 3.18], \\ f_{\text{GRI,CART}} &= 1.14 \in [0.31, 3.18], \\ f_{\text{GRI,Random Forest}} &= 1.28 \in [0.31, 3.18], \\ f_{\text{GRI,SVM}} &= 1.48 \in [0.31, 3.18]. \end{aligned}$$

   Hence, it can be assumed that variances are equal.
3. The value of Student's t distribution with 10 degrees of freedom level of 0.05 equals $t(0.05, 9) = 2.262$. Then, the null hypothesis holds true if $T < t$, where T values are as follows (calculated based on differences of two sequences):

Table 4 Results for the GRI algorithm with different β and γ values, for the mean criterion (6) of 10 patients in the diabetes treatment

γ	$\beta = 0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$	$\beta = 0.6$
1.0	0.13	0.11	0.11	0.11	0.11	0.11	0.12
0.99	0.13	0.12	0.12	0.11	0.11	0.12	0.12
0.98	0.13	0.11	0.11	0.11	0.10	0.11	0.12
0.97	0.14	0.12	0.12	0.11	0.11	0.12	0.12
0.96	0.14	0.12	0.12	0.11	0.11	0.12	0.12
0.95	0.13	0.12	0.11	0.11	**0.10**	0.11	0.12
0.94	0.14	0.13	0.12	0.11	0.11	0.12	0.12
0.93	0.14	0.13	0.12	0.11	0.11	0.12	0.12
0.92	0.14	0.13	0.12	0.11	0.11	0.12	0.12
0.91	0.14	0.13	0.12	0.11	0.11	0.13	0.12
0.90	0.14	0.13	0.12	0.11	0.11	0.12	0.12

Best result in bold

$$
\begin{aligned}
T_{\text{GRI,AQ-P1}} &= 8.24,\\
T_{\text{GRI,AQ-P2}} &= 9.05,\\
T_{\text{GRI,CART}} &= 6.02,\\
T_{\text{GRI,Random Forest}} &= 7.21,\\
T_{\text{GRI,SVM}} &= 5.37.
\end{aligned}
$$

In all cases, the T was greater than t and thus the null hypotheses can be rejected (in all cases with p-value ≈ 1).

Conclusions of the statistical hypothesis testing are the following. Comparing GRI with other algorithms, that is, AQ-P1, AQ-P2, CART, Random Forest, and SVM, it can be stated that the difference between GRI and other algorithms is significant, and in the application of diabetes treatment, the GRI performs better than the other algorithms.

The GRI classifier performs best for $\gamma = 0.95$ and $\beta = 0.4$ (see Table 4). This result indicates two issues. First, the hidden context and treatment have influenced patient's conditions. Therefore, the application of the forgetting mechanism had slight but positive effect on the knowledge quality (see the difference between $\gamma = 1$ and $\gamma = 0.95$, Table 4). However, in the biomedical application, even a small difference is priceless and can save human lives.

Second, in the considered application, the value of β close to 0.5 indicates that both generalization and specialization are important. According to the domain, the diabetics condition is rather slowly evolving and thus there are few repeating routines.[6] Therefore, in this case, it is more important to generalize data rather than to focus on single situations as it is in the domain of the electricity price market.

Finally, it is worth realizing how rules can be presented to a physician. It is assumed that only rules connected with an abnormal glucose level in blood are considered. In Table 5 all rules only for the glucose level not in norm for the patient No. 67 are presented. Hence, the rules can be reported to the physician in a *raw* form (Table 5) or in a *translated* form (Table 6).

[6] Authors have consulted this result with physicians from Wrocław Medical University (Poland) and this explanation was confirmed.

Table 5 Rules for the patient no. 67 and for the glucose level not in norm

IF (u^3 = before meal) THEN (y = not in norm)
IF (u^3 = other) THEN (y = not in norm)
IF (u^3 = after meal) THEN (y = not in norm)
IF (u^2 = [22:00-4:00)) THEN (y = not in norm)
IF (u^2 = [22:00-4:00)) AND (u^3 = other) THEN (y = not in norm)
IF (u^2 = [10:00-16:00)) AND (u^3 = other) THEN (y = not in norm)
IF (u^2 = [4:00-10:00)) AND (u^3 = other) THEN (y = not in norm)
IF (u^2 = [16:00-22:00)) AND (u^3 = other) THEN (y = not in norm)
IF (u^2 = [10:00-16:00)) AND (u^3 = before meal) THEN (y = not in norm)
IF (u^2 = [4:00-10:00)) AND (u^3 = before meal) THEN (y = not in norm)
IF (u^2 = [22:00-4:00)) AND (u^3 = before meal) THEN (y = not in norm)
IF (u^2 = [16:00-22:00)) AND (u^3 = before meal) THEN (y = not in norm)
IF (u^2 = [10:00-16:00)) AND (u^3 = after meal) THEN (y = not in norm)

Table 6 A report about the patient no. 67 and for the glucose level not in norm

For the patient No. **67** the glucose level is **not in norm** especially **before meal**, and in **other** conditions
Sometimes it happens also **after meal**, rarely **between 20:00 and 4:00**
In some occasions it happens **after meal** and **between 10:00 and 16:00**

This translation is relatively easy to perform as the rule-based knowledge is represented by logical formulae.

7 Conclusions

Application of the e-Health systems to support diabetes treatment is a challenging task both from the technical and the medical perspective. However, a lot of effort should be put to make patients' lives more bearable and decrease the costs of treatment. In this paper, a new method of knowledge extraction for supporting anamnesis was described. To allow the context tracking, a novel method for data aggregation was proposed. In order to avoid overfitting, a method of limiting search space was presented. The limitation of the search space can be regarded as a kind of *regularization*. It is especially important in the rules induction because rule-based models have a high value of Vapnik-Chervonenkis dimension, which entails high susceptibility to overfitting. Experimental validation shows that the GRI algorithm is not only more effective in comparison with other methods but also that it can be successfully applied to support the anamnesis in the diabetes treatment.

Future developments of this work will address: (i) development of the eDiab system as a service oriented system [24], (ii) conducting experiments not only with glucometer but also with weight and pressure gauge, (iii) deeper insight into the theoretical aspects of the limitation of the search space, (iv) other measures than coverage and accuracy, (v) heuristics for path selection.

References

1. Alemdar H, Ersoy C (2010) Wireless sensor networks for healthcare: a survey. Comput Netw 54:2688–2710
2. Andersen TL, and Martinez TR (1995) NP-completeness of minimum rule sets. In: Proceedings of the 10th international symposium on computer and information sciences, pp 411–418
3. Auer P, Warmuth MK (1998) Tracking the best disjunction. Mach Learn 32:127–150
4. Bach SH, and Maloof MA (2008) Paired learners for concept drift. In: Proceedings of eighth IEEE international conference on data mining, pp 23–32
5. Baena-García M, del Campo-Ávila J, Fidalgo R, Bifet A et al (2006) Early drift detection method. In: ECML PKDD 2006 workshop on knowledge discovery from data streams, Berlin
6. Bishop CM (2006) Pattern recognition and machine learning. Springer, Singapore
7. Bouamrane M-M, Rector A., Hurrell M. (2011) Using OWL ontologies for adaptive patient information modelling and preoperative clinical decision support. Knowl Inf Syst 29(2):405–418
8. Box GEP, Jenkins GM (1976) Time series analysis. Forecasting and control, Revised edn. Holden-Day, Oakland
9. Breiman L, Friedman JH, Olshen RA, Stone PJ (1984) Classification and regression trees. Wadsworth, Belmont
10. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
11. Bubnicki Z (1994) Knowledge-based approach as a generalization of pattern recognition problems. Syst Sci 19(2):5–20
12. Bubnicki Z (1980) Identification of control plants. Elsevier, Oxford
13. Chang WW, Sung TJ, Huang HW et al (2011) A smart medication system using wireless sensor network technologies. Sens Actuators A Phys 172(1):315–321
14. Cherkassky V, Mulier F (2007) Learning from data: concepts, theory, and methods. Wiley, New Jersey
15. Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3:261–283
16. Cook DJ, Holder LB (2000) Graph-based data mining. IEEE Intell Syst Appl 15(2):32–41
17. Devroye L, Györfi L, Lugosi G (1997) A probabilistic theory of pattern recognition. Springer, New York
18. Diestel R (2000) Graph theory. Springer, New York
19. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of KDD 2000, pp 71–80
20. European Coalition for Diabetes (2009). EU Diabetes Working Group (2009–2014) Delivering for Diabetes in Europe. Policy paper. http://www.ecdiabetes.eu/documents/EUDWG-policy-paper-2009-2014
21. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. Lect Notes Comput Sci 3171:66–112
22. Gama J, Sebastião R, Rodrigues P (2009) Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on KDD, pp 329–338
23. Georgii E, Tsuda K, Schölkopf G (2011) Multi-way set enumeration in weight tensors. Mach Learn 82(2):123–155
24. Grzech A, Rygielski P, Świątek P (2010) Translations of service level agreement in systems based on service-oriented architectures. Cybern Syst 41(8):610–627
25. Gonczarek A, Tomczak JM, Świątek J (2010) Decision rules clustering using K-means algorithm with different distance measures. In: Grzech A, Świątek P, Drapała J (eds) Advances in systems science. Exit, Warsaw, pp 139–147
26. Grandinetti L, Pisacane O (2011) Web based prediction for diabetes treatment. Futur Gener Comput Syst 27:139–147
27. Grzeszczak W (ed) (2010) Clinical recommendations for diabetics 2010. A Standpoint of Polish Diabetes Association. Pismo Polskiego Towarzystwa Diabetologicznego, vol 11, issue A (in Polish)
28. Harries MB (1999) Splice-2 comparative evaluation: electricity pricing. Technical Report UNSW-CSE-TR-9905
29. Harries MB, Sammut C, Horn K (1998) Extracting hidden context. Mach Learn 32:101–126
30. Herrera F, Carmona CJ, Gonzlez P, del Jesus MJ (2011) An overview on subgroup discovery: foundations and applications. Knowl Inf Syst 29(3):495–525
31. Holder LB, Cook DJ (2005) Graph-based Data Mining. In: Wang J (ed) Encyclopedia of data warehousing and mining. Information Science Reference, Hershey, pp 540–545

32. Hulten G, Spencer L, and Domingos P (2001) Mining time changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data dining, San Francisco, California, ACM, pp 97–106
33. International Diabetes Federation and Federation of European Nurses in Diabetes (2005) The policy puzzle: towards benchmarking in the EU 25. IDF/FEND Report. http://www.idf.org/webdata/docs/idf-europe/DiabetesReport2005
34. Inokuchi A, Washio T, Motoda H (2003) Complete mining of frequent patterns from graphs: mining graph data. Mach Learn 50:321–354
35. International Telecommunication Union (2008) Implementing e-health in developing countries. Guidance and Principles, ITU Report
36. Jordan MI (2004) Graphical models. Stat Sci 19(1):140–155
37. Kearns M, Li M, Valiant L (1994) Learning Boolean formulae. J ACM 41(6):1298–1328
38. Koleszynska J (2007) GIGISim—the intelligent telehealth system. Computer aided diabetes management—a new review. Lect Notes Comput Sci 4692:789–796
39. Kolter JZ, and Maloof MA (2005) Using additive expert ensembles to cope with concept drift. In: Proceedings of the twenty-second international conference on machine learning, ACM Press, New York, NY, pp 449–456
40. Kubat M (1993) Flexible concept learning in real-time systems. J Intell Robotic Syst 1:155–171
41. Kulkarnia P, Ozturk Y (2011) mPHASiS: mobile patient healthcare and sensor information system. J Netw Comput Appl 34:402–417
42. Last M (2002) Online classification of nonstationary data streams. Intell Data Anal 6:129–147
43. Last M, Klein Y, Kandel A (2001) Knowledge discovery in time series databases. IEEE Trans Syst Man Cybern Part B Cybern 31:160–169
44. Macía I, Grańa M, Paloc C (2011) Knowledge management in image-based analysis of blood vessel structures. Knowl Inf Syst 30(2):457–491
45. Maloof MA, Michalski RS (1999) Selecting examples for partial memory learning. Mach Learn 41(1): 27–52
46. Michalski RS (1969) On the quasi-minimal solution of the general covering problem. In: Proceedings of the Vth international symposium on information processing. Yugoslavia, A3:125–128
47. Mitchell T (1997) Machine learning. McGraw Hill, New York
48. Mohktar MS, Basilakis J, Redmond SJ, and Lovell NH (2010) A guideline-based decision support system for generating referral recommendations from routinely recorded home telehealth measurement data. In: Proceedings of 32nd annual international conference of the IEEE EMBS Buenos Aires, Argentina, pp 6166–6169, 31 August–4 September 2010
49. Mougiakakou SG, Bartsocas CS, Bozas E et al (2010) SMARTDIAB: a communication and information technology approach for the intelligent monitoring, management and follow-up of type 1 diabetes patients. IEEE Trans Inf Technol Biomed 14:622–633
50. Pantelopoulos A, Bourbakis NG (2010) A survey on wearable sensor-based systems for health monitoring and prognosis. IEEE Trans Syst Man Cybern C Appl Rev 40(1):1–12
51. Pattichis CS, Kyriacou E, Voskarides S et al (2002) Wireless telemedicine systems: an overview. IEEE Trans Antennas Propag Maga 44(2):143–153
52. Pawlak Z (2002) Decision Algorithms, Bayes' Theorem and Flow Graphs. In: Rutkowski L, Kacprzyk J (eds) Neural networks and soft computing. Physica, Springer, Heidelberk, New York
53. Pawlak Z (2004) Data analysis and flow graphs. J Telecommun Inf Technol 3:1–5
54. Potts J, Cook DJ, Holder LB (2007) Learning from supervised graphs. Stud Comput Intell 52:183–201
55. Qin B, Xia Y, Prabhakar S (2011) Rule induction for uncertain data. Knowl Inf Syst 29(1):103–130
56. Quinlan JR (1983) Learning efficient classification procedures and their application to chess end games. In: Michalski RS, Carbonell JG, Mitchell TM (eds) Machine learning: an artificial intelligence approach. Morgan Kaufmann, San Mateo, pp 463–482
57. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, Massachusetts
58. Rensink A (2004) Representing first-order logic using graphs. Lect Notes Comput Sci 3256:319–335
59. Salganicoff M (1997) Tolerating concept and sampling shift in lazy learning using prediction error context switching. Artif Intell Rev 11:133–155
60. Świątek J, Brzostowski K, Tomczak JM (2011) Computer aided physician interview for remote control system of diabetes therapy. In: InterSymp 2011, 23rd international conference on system research, informatics and cybernetics, Baden-Baden, Germany
61. Tomczak JM, Brzostowski K, Świątek J (2010) Knowledge extraction using shifting window from non-stationary datastreams. In: Grzech A (ed) Information systems architecture and technology: networks and networks' services. Oficyna Wydawnicza PWr, Wrocław, pp 321–331

62. Tomczak JM, Świątek J (2010) Bayesian classifiers with incremental learning for nonstationary data-streams. In: Grzech A, Świątek P, Drapała J (eds) Advances in systems science. EXIT, Warszawa, pp 251–260
63. UCI Machine Learning Repository (1994) Dataset prepared by Michael Kahn, MD, PhD. http://archive.ics.uci.edu/ml/datasets/Diabetes
64. Vapnik VN (1998) The statistical learning theory. A Wiley-Interscience Publication. John Wiley & Sons, New York
65. Verhoeven F, van Gemert-Pijnen L, Dijkstra K et al (2007) The contribution of teleconsultation and videoconferencing to diabetes care: a systematic literature review. J Med Internet Res 9(5):e37
66. Washio T, Motoda H (2003) State of the art of graph-based data mining. ACM SIGKDD Explor Newsl 5(1):59–68
67. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23:69–101
68. World Health Organization (2006) Definition and diagnosis of diabetes mellitus and intermediate hyperglycemia. Report of a WHO/IDF Consultation. http://whqlibdoc.who.int/publications/2006/9241594934_eng

Author Biographies

Jakub M. Tomczak received a double M.Sc. degree in computer science within a Double Diploma Programme from Wrocław University of Technology, Faculty of Computer Science and Management, Poland, and Blekinge Institute of Technology, Sweden, in 2009. Since 2009, he participates in a project about service-oriented systems that is supported by the European Union within the European Regional Development Fund Program. In 2009, his Master Thesis was awarded with the first place in the contest for the best Master Thesis organized by the Polish Information Processing Society. Currently, he is a Ph.D. student at Wrocław University of Technology, Institute of Computer Science. His research interests include machine learning, data mining, and context-aware systems with applications in biomedical systems.

Adam Gonczarek received a M.Sc. degree in computer science from Wrocław University of Technology, Faculty of Computer Science and Management, Poland, in 2009, and a M.Sc. degree in mathematics from Wrocław University of Technology, Faculty of Fundamental Problems of Technology, Poland, in 2010. In 2010, he was awarded with a scholarship for young scientists. Currently, he is a Ph.D. student at Wrocław University of Technology, Institute of Computer Science. His research interests include machine learning and computer vision, with special concern on human pose estimation.