Fitting Pattern Structures to Knowledge Discovery in Big Data

Sergei O. Kuznetsov

National Research University Higher School of Economics,
Pokrovskii bd. 11, 109028 Moscow, Russia
skuznetsov@hse.ru

Abstract. Pattern structures, an extension of FCA to data with complex descriptions, propose an alternative to conceptual scaling (binarization) by giving direct way to knowledge discovery in complex data such as logical formulas, graphs, strings, tuples of numerical intervals, etc. Whereas the approach to classification with pattern structures based on preceding generation of classifiers can lead to double exponent complexity, the combination of lazy evaluation with projection approximations of initial data, randomization and parallelization, results in reduction of algorithmic complexity to low degree polynomial, and thus is feasible for big data.

1 Introduction

In many real-world knowledge discovery problems researchers have to deal with complex descriptions different from binary datatables. In the last two decades the use of closed descriptions defined either in terms of Galois connections, semilattical similarity operation (i.e., operation which is idempotent, commutative, and associative) or in equivalent terms of counting inference proved to be very useful in various knowledge discovery applications, such as ontology and taxonomy engineering, mining association rules, machine learning, classification, and clustering. Several attempts were done in defining closed sets of graphs [27, 36, 29, 34, 32, 2, 19], strings [11, 12], numerical intervals [26, 25], logical formulas [7, 10], etc. In [16] a general approach called pattern structures was proposed, which allows one to extend FCA techniques to arbitrary partially ordered data descriptions. Using pattern structures, one can compute taxonomies, ontologies, implications, implication bases, association rules, concept-based (or JSM-) hypotheses in the same way it is done with standard concept lattices.

Big data gives another dimension to processing complex description. Using projections as approximation tools for pattern structures does not help enough, because general FCA-based knowledge discovery procedures, like generation of all concepts, implication bases, compact representations of association rules, sets of minimal hypotheses, have exponential worst-case complexity and many other complexity features making their computation hardly scalable [30, 33, 9, 3, 4]. To meet the big data challenge the problem settings of knowledge discovery should be recast to allow for faster procedures. In this paper we show how the classification and inference problems based on implications, association rules, and hypotheses can be reformulated to achieve scalability even for complex descriptions.

The rest of the paper is organized as follows: In Section 2 we recall basic definitions in pattern structures, give examples of applications with graph-based and interval based pattern structures. In Section 3 we describe our approach to efficient classification with pattern structures, we relate it to some other approaches outside FCA and make a conclusion in Section 4.

2 Knowledge Discovery with Pattern Structures

2.1 Main Definitions and Results

Let G be a set (of objects), let $(D, \sqcap)$ be a meet-semi-lattice (of all possible object descriptions) and let $\delta : G \to D$ be a mapping. Then $(G, \underline{D}, \delta)$, where $\underline{D} = (D, \sqcap)$, is called a *pattern structure*, provided that the set $\delta(G) := \{\delta(g) | g \in G\}$ generates a complete subsemilattice $(D_\delta, \sqcap)$ of $(D, \sqcap)$, i.e., every subset X of $\delta(G)$ has an infimum $\sqcap X$ in $(D, \sqcap)$. Elements of D are called *patterns* and are naturally ordered by subsumption relation $\sqsubseteq$: given $c, d \in D$ one has $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$. Operation $\sqcap$ is also called a *similarity operation.* A pattern structure $(G, \underline{D}, \delta)$ gives rise to the following derivation operators $(\cdot)^\diamond$:

$$A^\diamond = \prod_{g \in A} \delta(g) \qquad \text{for } A \subseteq G,$$

$$d^\diamond = \{g \in G | d \sqsubseteq \delta(g)\} \qquad \text{for } d \in (D, \sqcap)$$

These operators form a Galois connection between the powerset of G and $(D, \sqsubseteq)$. The pairs (A, d) satisfying $A \subseteq G$, $d \in D$, $A^\diamond = d$, and $A = d^\diamond$ are called the *pattern concepts* of $(G, \underline{D}, \delta)$, with extent A and *pattern intent* d. For $a, b \in D$ the *pattern implication* $a \to b$ holds if $a^\diamond \subseteq b^\diamond$, and the *pattern association rule* $a \to_{c,s} b$ with *confidence* c and *support* s holds if $s \geq \frac{|a^\diamond \cap b^\diamond|}{|G|}$ and $c \geq \frac{|a^\diamond \cap b^\diamond|}{|a^\diamond|}$. Like in case of association rules [37, 38], pattern association rules may be inferred from a concise representation that corresponds to the set of edges of the diagram of the pattern concept lattice. Operator $(\cdot)^{\diamond\diamond}$ is an algebraical closure operator on patterns, since it is idempotent, extensive, and monotone.

In [16] by applying the basic theorem of FCA [18] we showed that if $(D, \sqcap)$ is a complete meet-semi-lattice (where infimums are defined for arbitrary subsets of elements), in particular a finite semi-lattice, there is a subset $M \subseteq D$ with the following interesting property: The concepts of the formal context (G, M, I) where I is given as $gIm :\Leftrightarrow m \sqsubseteq \delta(g)$, called a *representation context* for $(G, \underline{D}, \delta)$, are in one-to-one correspondence with the pattern concepts of $(G, \underline{D}, \delta)$. The corresponding concepts have the same first components (called *extents*). These extents form a complete lattice, which is isomorphic to the concept lattice of

$(G,\ M,\ I)$. This result shows the way to binarizing complex data representation given by a pattern structure. The cost of this binarization may be a large amount of attributes of the representation context and hence, the space needed for storing this context.

The concept-based learning model for standard object-attribute representation (i.e., formal contexts) [13, 28, 30] is naturally extended to pattern structures. Suppose we have a set of positive examples G_+ and a set of negative examples G_- w.r.t. a *target attribute*, $G_+ \cap G_- = \emptyset$, objects from $G_\tau = G \setminus (G_+ \cup G_-)$ are called undetermined examples.

A pattern $h \in D$ is a *positive hypothesis* iff

$$h^\diamond \cap G_- = \emptyset \text{ and } \exists A \subseteq G_+ : A^\diamond = h$$

A positive hypothesis is the *least general generalization* of descriptions ("similarity") of positive examples, which is not contained in (does not cover) any negative example. A *negative hypothesis* is defined similarly. Various classification schemes using hypotheses are possible, as an example consider the following simplest scheme from [13, 29, 16]: If description $\delta(g)$ of an undetermined example g contains some positive hypothesis h, i.e., $h \sqsubseteq \delta(g)$, then g is *classified positively*. Negative classifications are defined similarly. If $\delta(g)$ contains hypotheses of both kinds, or if $\delta(g)$ contains no hypothesis at all, then the classification is contradictory or undetermined, respectively, and some probabilistic techniques allowing for a certain tolerance should be applied.

For some pattern structures (e.g., for the pattern structures on sets of graphs with labeled vertices) even computing subsumption of patterns may be NP-hard. Hence, for practical situations one needs approximation tools, which would replace the patterns with simpler ones, even if that results in some loss of information. To this end we use a contractive monotone and idempotent mapping $\psi : D \to D$ that replaces each pattern $d \in D$ by $\psi(d)$ such that the pattern structure $(G,\ \underline{D},\ \delta)$ is replaced by $(G,\ \underline{D},\ \psi \circ \delta)$. Under some natural algebraic requirements that hold for all natural projections in particular pattern structures we studied in applications, see [34], the meet operation $\sqcap$ is preserved: $\psi(X \sqcap Y) = \psi(X) \sqcap \psi(Y)$. This property of a projection allows one to relate hypotheses in the original representation with those approximated by a projection.The representation context of the projected case is obtained from the unprojected one by removing some attributes. If $\psi(a) \to \psi(b)$ and $\psi(b) = b$ then $a \to b$ for arbitrary $a,\ b \in D$. In particular, if $\psi(a)$ is a positive (negative) hypothesis in projected representation, then a is positive (negative) hypothesis in the original representation.

2.2 Pattern Structures in Applications

One may argue that a semi-lattice on descriptions is a too demanding requirement. We show easily that this is not the case, see also [1]. Any natural kind of descriptions available for data analysis has an explicitly or implicitly given partial order relation in the form of "is a" or "part of" relation. Having a partially ordered set $(P, \leq)$ of descriptions one can define a *similarity operation* $\sqcap$ on sets of descriptions as follows: For two descriptions X and Y from P

$$\{X\} \sqcap \{Y\} \ := \{Z | Z \leq X, Y, \ \forall Z_* \leq X, Y \ Z_* \not\geq Z\},$$

i.e., $\{X\} \sqcap \{Y\}$ is the set of all maximal common subdescriptions of descriptions X and Y. Similarity of non-singleton sets of descriptions $\{X_1, \ \ldots, \ X_k\}$ and $\{Y_1, \ \ldots, \ Y_m\}$ is defined as

$$\{X_1, \ \ldots, \ X_k\} \sqcap \{Y_1, \ \ldots, \ Y_m\} := MAX_{\leq}(\bigcup_{i,j}(\{X_i\} \sqcap \{Y_j\})),$$

where $MAX_{\leq}(\mathcal{X})$ returns maximal elements of $\mathcal{X}$ w.r.t. $\leq$. The similarity operation $\sqcap$ on sets of descriptions is commutative: $\mathcal{X} \sqcap \mathcal{Y} = \mathcal{Y} \sqcap \mathcal{X}$ and associative: $(\mathcal{X} \sqcap \mathcal{Y}) \sqcap \mathcal{Z} = \mathcal{X} \sqcap (\mathcal{Y} \sqcap \mathcal{Z})$. A set $\mathcal{X}$ of descriptions from P for which $\mathcal{X} \sqcap \mathcal{X} = \mathcal{X}$ holds defines a pattern. Then the triple $(G, (D, \ \sqcap), \delta)$, where D is the set of all patterns, is a pattern structure.

One can think of $\mathcal{X} \sqcap \mathcal{Y}$ in the following way, which also gives a straightforward approach to computing $\sqcap$: One takes the set of all subdesriptions of all descriptions of $\mathcal{X}$ and takes set-theoretic intersection (i.e., $\cap$) of this set with the set of all subdescriptions of all descriptions of $\mathcal{Y}$. Finally, from the resulting set of subdescriptions one chooses the maximal ones w.r.t. the partial order $\leq$ on descriptions.

From the lattice-theoretical viewpoint the whole construction looks as follows: One takes the distributive lattice of order ideals of $(P, \ \leq)$ [6], with $\sqcap$ being the meet in this lattice, and computes its subsemilattice generated by all descriptions of objects from G. For a finite sets G this subsemilattice is finite too, and the respective $\sqcup$ operator can be defined as

$$\mathcal{X} \sqcup \mathcal{Y} = \sqcap \{\delta(g) \mid g \in G; \mathcal{X}, \mathcal{Y} \sqsubseteq \delta(g)\}.$$

Note that $\sqcup$ is not the join of the distributive lattice of order ideals, it is the "ad hoc" join given by $\sqcap$ and descriptions of objects from G, and therefore the lattice given by $\sqcap$ and $\sqcup$ is not necessarily distributive.

Pattern Structures on Sets of Graphs. In [27, 29] we proposed a semi-lattice on sets of graphs with labeled vertices and edges. This semilattice is based on a partial order given by subgraph isomorphism or its generalizations. For example, in [29, 16] the following natural order relation on graphs with labeled vertices and edges, called *domination relation*, was proposed. Consider connected graphs[1] with vertex and edge labels from set $\mathcal{L}$ partially ordered by $\preceq$. Denote the set of graphs with labeled vertices and edges by P. Each graph Γ from P is a quadruple of the form $((V, \ l), \ (E, \ b))$, where V is a set of vertices, E is a set of edges, $l : V \to \mathcal{L}$ is a function assigning labels to vertices, and $b : E \to \mathcal{L}$ is a function assigning labels to edges. In $(P, \ \leq)$ we do not distinguish isomorphic graphs.

[1] Omitting the condition of connectedness, one obtains a similar, but computationally much harder model.

For two graphs $\Gamma_1 := ((V_1,\ l_1),\ (E_1,\ b_1))$ and $\Gamma_2 := ((V_2,\ l_2),\ (E_2,\ b_2))$ from P we say that Γ_1 *dominates* Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a *subgraph* of Γ_1) if there exists an injection $\varphi : V_2 \to V_1$ such that it *respects edges*: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$ and *fits under labels*: $l_2(v) \preceq l_1(\varphi(v))$, if $(v, w) \in E_2$, then $b_2(v, w) \preceq b_1(\varphi(v), \varphi(w))$.

Obviously, $(P,\ \leq)$ is a partially ordered set. Having a partial order on graphs, one can use the definitions above to define similarity operation $\sqcap$ and closure operator $(\cdot)^{\diamond\diamond}$. A set of graphs X is called *closed* if $X^{\diamond\diamond} = X$. This definition is related to the notion of a closed graph in data mining and graph mining, which is important for computing association rules between graphs. Closed graphs are defined in [40] in terms of "counting inference" as follows. Given a graph dataset E, support of a graph g or *support*(g) is a set (or number) of graphs in E that have subgraphs isomorphic to g. A graph g is called *closed* if no supergraph f of g (i.e., a graph such that g is isomorphic to its subgraph) has the same support. In terms of pattern structures, E is a set of objects, each object $e \in E$ having a graph description $\delta(e)$, support(g) $= \{e \in E | \delta(g) \leq e\}$. Closed sets of graphs [27, 29] form a *meet semi-lattice* w.r.t. $\sqcap$. Closed graphs [40] do not have this property, since in general, there are pairs of closed graphs with no infimums. However, closed graphs and closed sets of graphs are intimately related [34, 31] as stated in the following.

Proposition 1. *Let a dataset described by a pattern structure* $(E, (D, \sqcap), \delta)$ *on graphs be given, i.e., E is a set of objects with graph descriptions, and* $(D, \sqcap)$ *is a semilattice on graph sets. Then the following two properties hold:*

1. *For a closed graph g there is a closed set of graphs G such that $g \in G$.*
2. *For a closed set of graphs G and an arbitrary $g \in G$, graph g is closed.*

Hence, one can use the algorithms for computing (*frequent*) closed sets of graphs [29, 32] to compute closed graphs. A learning model based on *graph pattern structures* along the lines of the previous subsection was used in series of applications in bioinformatics [17, 34], in text analysis [15] and conflict analysis [14].

Pattern Structures on Intervals. In practice, a typical object-attribute data table is not binary, but has many-valued attributes. In FCA a quadruple $\mathbb{K}_1 = (G,\ S,\ W,\ I_1)$, where G, S, W are sets and I_1 is a ternary relation $I_1 \subseteq G \times S \times W$, is called a *many-valued context.* Consider an example of analyzing *gene expression data* (GED) given by many-valued tables [25]. The names of rows correspond to genes, the names of the columns correspond to *situations* where genes are tested. A table entry is called an *expression value.* A row in the table is called *expression profile* associated to a gene. In terms of many-valued contexts, the set of genes makes the set of objects G, the set of situations makes the set of many-valued attributes S, the set of expression values makes the set $W \subset \mathbb{R}$ and $J \subseteq G \times S \times W$. Then $\mathbb{K} = (G, S, W, J)$ is a many-valued context representing a GED. The fact $(g,\ s,\ w) \in J$ or simply $g(s) = w$ means that gene g has an expression value w for situation s. The objective of GED analysis is to extract subsets of genes sharing "similar values" of W, i.e. lying in a same interval.

To represent intervals of numerical values, one can use *interordinal scaling* (see p. 42 in [18]) of the form $\mathbb{I}_{W_s} = (W_s,\ W_s,\ \leq)|(W_s,\ W_s,\ \geq)$, where *apposition* $|$ *of two contexts*, applied to a pair of contexts with the same set of objects, returns a context with the same set of objects and the set of attributes being the disjoint union of the attribute sets of the apposed contexts. For example, for the set $W_s = \{4, 5, 6\}$, the interordinal scale is

	$s_1 \leq 4$	$s_1 \leq 5$	$s_1 \leq 6$	$s_1 \geq 4$	$s_1 \geq 5$	$s_1 \geq 6$
4	×	×	×	×		
5		×	×	×	×	
6			×	×	×	×

The intents of an interordinal scale are intervals of attribute values.

Instead of scaling, one can directly work with many-valued attributes by applying *interval pattern structures*, which were successfully applied to the GED analysis [24, 25]. For two intervals $[a_1,\ b_1]$ and $[a_2,\ b_2]$, with $a_1,\ b_1,\ a_2,\ b_2 \in \mathbb{R}$, we define their meet as

$$[a_1,\ b_1] \sqcap [a_2,\ b_2] = [\min(a_1,\ a_2),\ \max(b_1,\ b_2)].$$

This operator is obviously idempotent, commutative and associative, thus defining a pattern structure on tuples (vectors) of intervals of attribute values.

The natural order relation (subsumption) on intervals is given as follows:

$$\begin{aligned} &[a_1,\ b_1] \sqsubseteq [a_2,\ b_2] \\ \Leftrightarrow &[a_1,\ b_1] \sqcap [a_2,\ b_2] = [a_1,\ b_1] \\ \Leftrightarrow &[\min(a_1,\ a_2),\ \max(b_1,\ b_2)] = [a_1,\ b_1] \\ \Leftrightarrow &a_1 \leq a_2 \text{ and } b_1 \geq b_2 \end{aligned}$$

Contrary to the usual intuition, smaller intervals subsume larger intervals that contain the former. The meet $\sqcap$ for vectors (tuples) of intervals is defined component-wise. Interval p-vector patterns are p-dimensional rectangular parallelepipeds in Euclidean space. Another step further would be to allow for any type of patterns for each component. The general meet operator on a vector like that is defined by component-wise meet operators.

For a many-valued context (G, M, W, J) with $W \subset \mathbb{R}$ consider the respective pattern structure $(G, (D, \sqcap), \delta)$ on interval vectors, the interordinal scaling $\mathbb{I}_{W_s} = (W_s,\ W_s,\ \leq)|(W_s,\ W_s,\ \geq)$, and the context K_I resulting from interordinal scaling $\mathbb{I}_{W_s}$ to (G, M, W, J). Consider usual derivation operators $(\cdot)'$ in context K_I. Then the following proposition establishes an isomorphism between the concept lattice of K_I and the pattern concept lattice of $(G, (D, \sqcap), \delta)$.

Proposition 2. *Let $A \subseteq G$, then the following statements 1 and 2 are equivalent:*

1. A is an extent of the pattern structure $(G, (D, \sqcap), \delta)$ *and* $A^{\diamond} = \langle[\underline{m}_i, \overline{m}_i]\rangle_{i\in[1,p]}$
2. A is a concept extent of the context K_I *so that for all* $i \in [1,\ p]$ $\underline{m}_i$ *is the largest number* n *such that the attribute* $s_i \geq n$ *is in* A' *and* $\overline{m}_i$ *is the smallest number* n *such that the attribute* $s_i \leq n$ *is in* A'.

So, the lattice of interval pattern structures is isomorphic to the concept lattice of the context that arises from the interordinal scaling of the initial many-valued numerical context. However, interval tuples give better understanding of results and computation with them is faster than that with interordinal scaling [23–25].

Other Types of Pattern Structures. Partition pattern structures [5] are useful for describing and computing dependencies in many-valued contexts when attribute values are understood nominally, i.e., having no similarity as in case of similarity intervals for numbers. Taking attributes of the many-valued context as new objects and partitions on the set of (old) objects w.r.t. attribute values as patterns allows one to compute functional dependencies directly from the table, without quadratic blow-up resulting from reducing many-valued context to a binary context where new objects are pairs of the objects of the initial many-valued context, and implications in the new context are syntactically the same as functional dependencies in the original many-valued context, see [18]. Pattern structures were also used for computing ontologies from partial ordered data on annotations [8].

3 Pattern Structures for Big Data

On the one hand, the goal of computing implications, association rules, hypotheses, and there concise representations is to "understand" data by creating "knowledge" in the form of implicational dependencies (classifiers). On the other hand, the goal is to use these dependencies for making predictions for new data. Intractability results on the sizes of concepts [30], implication bases [33, 9, 4], (minimal) hypotheses [30, 3] say that the amount of "knowledge" generated from data can be exponential in the size of data, this amount being hardly possible to estimate before computing the whole set of dependencies. This kind of knowledge cannot give us better explanation of data than data themselves and for large datasets may be even inractable. Possible solutions can be *feature selection* approaches (i.e., selecting representative attributes), sampling (i.e., selecting representative objects), generating small subsets of dependencies which would classify "almost in the same way" as the whole set of dependencies. Another option is not to generate dependencies at all, since if one needs "knowledge" for making predictions, i.e., defining missing information like classes of objects described by new data, one does not need having (all) knowledge given explicitly, one just needs having predictions equivalent to those made when all knowledge is there.

FCA allows one to benefit from this important distinction by using the equivalence of implicational closure and closure given by the double prime operator $(\cdot)''$, or $(\cdot)^{\diamond\diamond}$ in the pattern structures.

Missing information about an object g_n with a description $\delta(g_n)$ is something that distinguishes $\delta(g_n)$ from a "larger" information $F \in D$ such that $\delta(g_n) \sqsubseteq F$. A very natural example of such a missing information is a value of one binary attribute, which can be a target or class attribute, all other information about objects and their descriptions being given by a pattern structure $(G, (D, \sqcap), \delta)$, so one has to predict the value of the target attribute. The situation can be described in the extended pattern structure

$$(G, (D*, \sqcap*), \delta*) = (G, ((D, \sqcap) \times (\{0,1\}, \wedge)), \delta \cup val),$$

where $\wedge$ is logical conjunction and the mapping $val : G \to \{0,1\}$ says whether an object has the target attribute or not. In the following subsections we show how it works for various types of dependencies. Let us first consider the complexity of computing in pattern structures and in projections.

Many algorithms for computing concept lattices, like NextClosure and CbO, may be adapted to computing pattern lattices in bottom-up way. The worst-case time complexity of computing all pattern concepts of a pattern structure $(G,\ \underline{D},\ \delta)$ in the bottom-up way is $O((p(\sqcap) + p(\sqsubseteq)|G|) \cdot |G| \cdot |L|)$, where $p(\sqcap)$ is time needed to perform $\sqcap$ operation, $p(\sqsubseteq)$ is time needed to test $\sqsubseteq$ relation, and L is the set of all patterns. In case of graphs, even $p(\sqsubseteq)$ may be exponential w.r.t. the number of graph vertices, that is why approximations (like those given by projections) are often needed. For a fixed projection size $p(\sqsubseteq)$ and $p(\sqcap)$ can be considered constant. To compute graph patterns in the top-bottom way, e.g., for computing *frequent patterns*, one should seriously remake an existing FCA algorithm by getting access to the "fine" structure of descriptions, like it was done for graphs in [32]. The worst-case time complexity of computing the set of interval pattern structures is $O(|G|^2 \cdot |M| \cdot |L|)$, which in practice can be much lower than the worst-case complexity of computing the set of all concepts of the interordinally scaled numerical context, which is $O(|G|^2 \cdot |W| \cdot |L|)$, where W is the set of all attribute values.

3.1 Classifying with Implications and Association Rules

One of the basic observations in FCA is the equivalence of the implicational closure of a subset of attributes $B \subseteq M$ (i.e., applying to B implications of the base until the result cannot be extended anymore) and the closure given by the double prime operator $(\cdot)''$ [18]. For example, in [35] we used this fact to correct data tables by filling missing attributes. Due to the equivalence of a pattern structure to a representation context [16], the same equivalence holds in an arbitrary pattern structure. So, when the class attribute of the description of a new object g_n to be classified with respect to the implications that hold in the training set given by a pattern structure $(G, (D, \sqcap), \delta)$ is missing, one can just compute the closure (w.r.t. $(G, (D*, \sqcap *), \delta *)$) of the intersection of the description of the new object with description of every object $g \in G$. If for some object g the closure contains the target attribute, g_n is classified positive by implications of $(G, (D*, \sqcap *), \delta *)$, otherwise it is classified negatively. This can be described as the following simple two-stage procedure:

1. For every $g \in G$ compute $(\delta(g_n) \sqcap \delta(g))^{\diamond}$, i.e. select all objects from G whose descriptions contain $\delta(g_n) \sqcap \delta(g)$. This takes $O(|G| \cdot (p(\sqcap) + |G| \cdot p(\sqsubseteq)))$ time.

2. If for some $g \in G$ all objects from $(\delta(g_n) \sqcap \delta(g))^{\diamond}$ have the target attribute, classify g_n positively, otherwise negatively. This takes $O(|G|^2)$ time for looking for the target attribute in object descriptions in at most $|G|$ families of object subsets, each subset consisting of at most $|G|$ objects.

If there is still need for collecting implications, an option is to extract only those implications from the minimal generator or proper premise bases, which would produce this very classification if it had been done with the use of an explicitly generated implication base. To this end one computes minimal generators or proper premises of the pattern E (target attribute) given the set of objects $G \cup \{g_n\}$ together with their descriptions, using standard algorithms and their improvements, like e.g. in [39]. Thus, the process of collecting implications will follow classification tasks which come from practical needs. One can call this collection of implications an *anecdotal* or *experimental subbase.*

Example. A formal context represents the simplest form of a pattern structure. Consider the following context, where m_0 is the target attribute.

$G \setminus M$	m_0	m_1	m_2	m_3	m_4	m_5	m_6
g_1			×	×		×	×
g_2		×		×	×		×
g_3		×	×		×	×	
g_4	×		×	×	×	×	×
g_5	×	×		×	×	×	×
g_6	×	×	×		×	×	×
g_7	×	×	×	×		×	×
g_8	×	×	×	×	×		×
g_9	×	×	×	×	×	×	

Here, we have $2^3 = 8$ pseudo-intents: $\{m_1, m_2, m_3\}$, $\{m_1, m_2, m_6\}$, $\{m_1, m_5, m_3\}$, $\{m_1, m_5, m_6\}$, $\{m_4, m_2, m_3\}$, $\{m_4, m_2, m_6\}$, $\{m_4, m_5, m_3\}$, $\{m_4, m_5, m_6\}$ and the corresponding implications in the Duquenne-Guigues base [20]. To classify two new objects with intents $\{m_1, m_2, m_5\}$ and $\{m_1, m_2, m_3\}$ w.r.t. m_0 in the standard way, one needs to compute all implications in the base (the number of them is exponential in $|M|$) and apply them all to new object intents. Instead of doing this, one can just compute closures $(\{m_1, m_2, m_5\} \cap g')''$ and $(\{m_1, m_2, m_3\} \cap g')''$ for every $g \in G$ - which takes just $O(|G|^2 \cdot |M|)$ time - to realize that the first object should be classified negatively (no target attribute m_0), and the second object should be classified positively.

In case of using association rules for classification instead of implications, a natural way is to accept that the new object g_n has the missing subpattern (attribute) F if there is an association rule $D \to E$ with a confidence more than θ, where θ is a parameter such that $D \sqsubseteq \delta(g_n)$ and $F \sqsubseteq E$. If one does not need to produce all valid association rules (with sufficiently large confidence and support), but just needs to know how the set of the valid rules would classify the new object g_n w.r.t the missing subpattern F, one can proceed in the following way: for every object g from G one computes $\delta(g_n) \sqcap \delta(g)$ and tests whether at least θ part of all objects from G that contain $\delta(g_n) \sqcap \delta(g)$ also contain F. This takes $O(|G| \cdot p(\sqcap))$ time for computing all intersections and $O(|G|^2 \cdot p(\sqsubseteq))$ for testing all containments.

3.2 Classifying with Hypotheses

Classification with hypotheses even when they are not to be generated, is not tractable, unless P=NP [30]. However, one can slightly change the definition of the classification to obtain the following tractable version, which we call *lazy hypothesis evaluation.*

- Suppose that the object g_n to be classified is added to the set of positive examples. Can the hypotheses arising from the new context classify the object g_n positively if we "forget" its class value?
- Suppose also that the object g_n to be classified is added to the set of negative examples. Can the hypotheses arising from the new context classify the object g_n negatively if we "forget" its class value?

If only the first question is answered "yes", object g_n is classified positively, if only the second question is answered "yes", then g_n is classified negatively. If both or none, the object remains unclassified. Note that if there is a positive hypothesis in favor of positive classification of g_n as described in Section 2.1 the first question is answered "yes", symmetric for negative classification and second question.

In this classification setting, one does not need to compute all hypotheses, but can take the largest ones that are contained in $\delta(g_n)$. For considering positive classification these hypotheses are sought among all intersections of the form $\delta(g_n) \sqcap \delta(g)$, where $g \in G_+$, after which this intersections are tested for the containment in descriptions of all negative examples $g \in G_-$. Similarly for negative hypotheses. Note that there are at most $|G_+|$ such positive and at most $|G_-|$ such negative hypotheses, so, the total time needed for computing classification is at most $O(|G_+| + |G_-|) \cdot (p(\sqcap) + p(\sqsubseteq))$, where $p(\sqcap) \geq p(\sqsubseteq)$. Together with the above considerations about implications this proves the following

Proposition 3. *Classification of an object can be done for*

- *implications in $O(|G| \cdot (|G| \cdot p(\sqsubseteq) + p(\sqcap)))$ time and in $O(|G|^2)$ time in projections of fixed size;*
- *lazy hypothesis evaluation in $O((|G_+| + |G_-|) \cdot p(\sqcap))$ time and in $O(|G_+| + |G_-|)$ time in a projection of fixed size.*

3.3 Parallelization and Possible Randomization

We have reduced classification with implications, association rules, and hypotheses to computing $(\delta(g) \sqcap \delta(g_n))^\diamond$ and testing the target attribute in all objects of this set, which is easily parallelizable: one partitions the dataset G in $G = G_1 \cup \ldots \cup G_k$, where k is the number of processors, computes in each G_i the set of objects $(\delta(g) \sqcap \delta(g_n))^{\diamond_i}$, tests the target attribute for all objects in the union of these sets over i. Thus, we have the following

Proposition 4. *Classification of m objects using k processors can be done for*

- *implications in $O(|G| \cdot (|G| \cdot p(\sqsubseteq) + p(\sqcap)) \cdot m/k)$ time and in $O(|G|^2 \cdot m/k)$ time in projections of fixed size;*
- *lazy hypothesis evaluation in $O((|G_+| + |G_-|) \cdot p(\sqcap) \cdot m/k)$ time and in $O((|G_+| + |G_-|) \cdot m/k)$ time in projection of fixed size.*

Randomization can be realized by taking random objects from each of G_i for computing the closures. Here, classification with association rules will not change, since the random estimate of the confidence will converge to the confidence, but will change classifications based on implications and hypotheses, making them probabilistic: one will have to assign correct probability values to them.

4 Related Work and Conclusions

Pattern structures propose a useful means for discovering implicational dependencies in data given by complex ordered descriptions. Even most concise representations of knowledge that can be mined from data can be intractable for binary contexts, the simplest type of pattern structures. We have proposed an approach where one does not need to mine all knowledge, but produce necessary classifications directly from data, saving knowledge that was used for classification. Our approach is close to some approaches outside FCA: Nearest Neighbors [41] (finding nearest classes in metric spaces), Case-Based Reasoning [21] (classifying similar to classification of similar cases), abduction in Horn theories [22] (lazy evaluation from models instead of generating implications on Boolean variables), however differs from them in being based on partially ordered structures, not metric or Boolean. Using projections, parallel computations and randomization, one can drastically reduce algorithmic complexity from double exponential to low degree polynomial and meet the challenge of big complex data.

Acknowledgments. This work was done within the project "Mathematical models, algorithms, and software tools for intelligent analysis of structural and textual data" supported by the Basic Research Program of the National Research University Higher School of Economics (Moscow).

References

1. Aït-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient Implementation of Lattice Operations. ACM Transactions on Programming Languages and Systems 11(1), 115–146 (1989)

2. Arimura, H., Uno, T.: Polynomial-Delay and Polynomial-Space Algorithms for Mining Closed Sequences, Graphs, and Pictures in Accessible Set Systems. In: Proc. SDM, pp. 1087–1098 (2009)
3. Babin, M.A., Kuznetsov, S.O.: Enumerating Minimal Hypotheses and Dualizing Monotone Boolean Functions on Lattices. In: Jäschke, R. (ed.) ICFCA 2011. LNCS (LNAI), vol. 6628, pp. 42–48. Springer, Heidelberg (2011)
4. Babin, M.A., Kuznetsov, S.O.: Computing Premises of a Minimal Cover of Functional Depedencies is Intractable. Discr. Appl. Math. 161, 742–749 (2013)
5. Baixeries, J., Kaytoue, M., Napoli, A.: Computing Functional Dependencies with Pattern Structures. In: Proc. 9th International Conference on Concept Lattices and Their Applications (CLA 2012), Malaga (2012)
6. Birkhoff, B.: Lattice Theory. ACM (1991)
7. Chaudron, L., Maille, N.: Generalized Formal Concept Analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 357–370. Springer, Heidelberg (2000)
8. Coulet, A., Domenach, F., Kaytoue, M., Napoli, A.: Using pattern structures for analyzing ontology-based annotations of biomedical data. In: Cellier, P., Distel, F., Ganter, B. (eds.) ICFCA 2013. LNCS (LNAI), vol. 7880, pp. 76–91. Springer, Heidelberg (2013)
9. Distel, F., Sertkaya, B.: On the Complexity of Enumerating Pseudo-intents. Discrete Applied Mathematics 159(6), 450–466 (2011)
10. Férré, S., Ridoux, O.: A Logical Generalization of Formal Concept Analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 371–385. Springer, Heidelberg (2000)
11. Férré, S., King, R.D.: Finding Motifs in Protein Secondary Structure for Use in Function Prediction. Journal of Computational Biology 13(3), 719–731 (2006)
12. Ferré, S.: The Efficient Computation of Complete and Concise Substring Scales with Suffix Trees. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 98–113. Springer, Heidelberg (2007)
13. Finn, V.K.: Plausible Reasoning in Systems of JSM Type. Itogi Nauki i Tekhniki, Seriya Informatika 15, 54–101 (1991) (in Russian)
14. Galitsky, B.A., Kuznetsov, S.O., Samokhin, M.V.: Analyzing Conflicts with Concept-Based Learning. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS (LNAI), vol. 3596, pp. 307–322. Springer, Heidelberg (2005)
15. Galitsky, B.A., Kuznetsov, S.O., Usikov, D.: Parse Thicket Representation for Multi-sentence Search. In: Pfeiffer, H.D., Ignatov, D.I., Poelmans, J., Gadiraju, N. (eds.) ICCS 2013. LNCS, vol. 7735, pp. 153–172. Springer, Heidelberg (2013)
16. Ganter, B., Kuznetsov, S.O.: Pattern Structures and Their Projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
17. Ganter, B., Grigoriev, P.A., Kuznetsov, S.O., Samokhin, M.V.: Concept-based Data Mining with Scaled Labeled Graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 94–108. Springer, Heidelberg (2004)
18. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
19. Garriga, G., Khardon, R., De Raedt, L.: Mining Closed Patterns in Relational, Graph and Network Data, Annals of Mathematics and Artificial Intelligence (2013)
20. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de donnees binaires. Math. Sci. Humaines 95, 5–18 (1986)
21. Hullermeier, E.: Case-Based Approximate Reasoning. Springer (2007)

22. Kautz, H.A., Kearns, M.J., Selman, B.: Reasoning with characteristic models. In: Proc. AAAI 1993, pp. 1–14 (1993)
23. Kaytoue, M., Duplessis, S., Kuznetsov, S.O., Napoli, A.: Two FCA-Based Methods for Mining Gene Expression Data. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 251–266. Springer, Heidelberg (2009)
24. Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Revisiting Numerical Pattern Mining with Formal Concept Analysis. In: Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 1342–1347 (2011)
25. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. Inf. Sci. 181(10), 1989–2001 (2011)
26. Kuznetsov, S.O.: Stability as an Estimate of the Degree of Substantiation of Hypotheses on the Basis of Operational Similarity. Nauchno-Tekhnicheskaya Informatsiya, Ser. 2 24(12), 21–29 (1990)
27. Kuznetsov, S.O.: JSM-method as a machine learning method. Itogi Nauki i Tekhniki, Ser. Informatika 15, 17–50 (1991) (in Russian)
28. Kuznetsov, S.O.: Mathematical aspects of concept analysis. J. Math. Sci. 80(2), 1654–1698 (1996)
29. Kuznetsov, S.O.: Learning of Simple Conceptual Graphs from Positive and Negative Examples. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 384–391. Springer, Heidelberg (1999)
30. Kuznetsov, S.O.: Complexity of Learning in Concept Lattices from Positive and Negative Examples. Discr. Appl. Math. 142, 111–125 (2004)
31. Kuznetsov, S.O.: Pattern Structures for Analyzing Complex Data. In: Sakai, H., Chakraborty, M.K., Hassanien, A.E., Ślęzak, D., Zhu, W. (eds.) RSFDGrC 2009. LNCS (LNAI), vol. 5908, pp. 33–44. Springer, Heidelberg (2009)
32. Kuznetsov, S.O.: Computing Graph-Based Lattices from Smallest Projections. In: Wolff, K.E., Palchunov, D.E., Zagoruiko, N.G., Andelfinger, U. (eds.) KONT/KPP 2007. LNCS (LNAI), vol. 6581, pp. 35–47. Springer, Heidelberg (2011)
33. Kuznetsov, S.O., Obiedkov, S.A.: Some Decision and Counting Problems of the Duquenne-Guigues Basis of Implications. Discrete Applied Mathematics 156(11), 1994–2003 (2008)
34. Kuznetsov, S.O., Samokhin, M.V.: Learning Closed Sets of Labeled Graphs for Chemical Applications. In: Kramer, S., Pfahringer, B. (eds.) ILP 2005. LNCS (LNAI), vol. 3625, pp. 190–208. Springer, Heidelberg (2005)
35. Kuznetsov, S.O., Revenko, A.: Finding Errors in Data Tables: An FCA-based Approach. Annals of Mathematics and Artificial Intelligence (2013)
36. Liquiere, M., Sallantin, J.: Structural Machine Learning with Galois Lattice and Graphs. In: Proc. ICML 1998 (1998)
37. Luxenburger, M.: Implications partielle dans un contexte. Math. Sci. Hum. (1991)
38. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Minining of Association Rules Based on Using Closed Itemset Lattices. J. Inf. Systems 24, 25–46 (1999)
39. Ryssel, U., Distel, F., Borchmann, D.: Fast computation of proper premises. In: Proc. CLA 2011 (2011)
40. Yan, X., Han, J.: CloseGraph: Mining closed frequent graph patterns. In: Proc. KDD 2003, pp. 286–295. ACM Press, New York (2003)
41. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search - The Metric Space Approach. Springer (2006)