A System for Different Concepts Generation and Application

Xenia Naidenova[1,*,†], Victor Shagalov[2,†] and Tatiana Martirova[3,†]

[1,3] *Medical Academy, St.-Petersburg, 194044 Academic Lebedev Street, Russia*
[2] *MatrixDnA Ltd, Israel*

Abstract
In this paper, programming library and algorithms for solving formal concept related tasks in real world domains are presented. The main goal of the proposed system is the searching of all closed itemsets (concepts). Constructing Galois lattice of concepts allows to additionally generate good classification tests and functional dependences for given classifications on a given data set. In general, these tasks are based on ordinal procedure for shallow or deep machine learning for classifications. We show that formal concept analysis is closely related to modeling plausible classification reasoning

Keywords
formal concepts, good classification tests, functional dependencies, plausible reasoning

1. Introduction

Modern accent in Machine Learning (ML) is shifted to the numerical solutions as opposed to plausible reasoning. Of course, the linear additive model or kernel model allows great data compression but at the same time the source of information is lost. On the other hand, the Formal Concept Analysis (FCA) has a natural ability to model plausible reasoning. When there is no explanation based on a model that is difficult for understanding and sometimes conflicting with human sense, then the obtained results are not reliable. Obviously, the integration of plausible reasoning with the FCA as one of the instruments of ML is crucial in the context of AI.

The problem of finding all closed sets (concept lattice) has been solved by many researchers: B. Ganter, D. Borchmann, M. Zaki, S. Kuznetsov, and many others. The source for many of these works was the algorithm of B. Ganter [1]. The Next-Closure algorithm has been proposed in [2] as an improvement of previous versions of this algorithm. One of the most efficient algorithms, Charm, has been proposed by M. Zaki in [3]. The algorithm presented in this paper is based on a previously developed algorithm for extracting only good classification tests (GCTs) [4] from a given context. The algorithm uses the original decomposition of the source context into the attributive and object sub-contexts described in [5].

In the paper [4], it has been shown that the GCTs are formal concepts and therefore they are contained in the Galois lattice built over a given context with additional attribute(s) that specify the partitioning of context's objects into non-overlapping classes. However, all the algorithms developed for deriving GCTs as formal concepts did not aim to build and did not build the complete Galois lattice over a given context, on the contrary, these algorithms generate only those elements of the lattice that correspond to all good classification (diagnostic) tests (redundant and non-redundant, i.e. test generators).

FCA4AI 2024: The 12th International Workshop "What can FCA do for Artificial Intelligence?, October 19 2024, Santiago de Compostela, Spain
[*] Corresponding author.
[†] These authors contributed equally.
ksennaidd@gmail.com (X. Naidenova); shagalovv@gmail.com (V. Shagalov); martta462@yandex.ru (T. Martirova)
0000-003-2377-7093 (X. Naidenova); 0009-0001-7131-9081 (V. Shagalov); 0000-0003-0000-6608 (T. Martirova)

The first algorithm for generating good maximally redundant classification tests (GMRTs) has been implemented in system SISIF [6], but it had a very small memory. In addition to GMRTs, the SISIF also has generated functional dependencies (FDs) as the best approximation of a given classification of given objects. The system also has implemented an algorithm for extracting all generators from a given GMRT, equivalent to it. An overview of the main algorithms developed for building GCTs can be found in [5].

This paper presents a new system for extracting the different types of itemsets (concepts, dependencies, logical rules, classification tests) based on constructing the lattice of all closed frequent concepts in a given context. This system has the following features:

1. Work with large datasets;
2. Work with multivalued attributes of objects;
3. Well-structured and simple for usage;
4. Applicable for multiple FCA tasks;

Further, the work is organized as follows. Section 2 gives basic definitions related to the FCA, GCTs, and plausible reasoning rules. Section 3 describes Diagnostic Test Machine (DTM) as a software library for finding different concepts and logical rules in data sets. Section 4 briefly describes the experiments. Section 5 deals with the plausible reasoning rules application, and Section 6 offers some concluding remarks and describes some future investigation.

2. Basic definitions

Let $S = \{1, 2,\dots, N\}$ be the set of objects' indices (objects, for short) and $T = \{A_1, A_2, \dots, A_j, \dots A_m\}$ be the set of attributes' values (values, for short). Each object is described by a collection of values from T. Let $s \subseteq S$, $t \subseteq T$. Denote by t_i, $t_i \subseteq T$, $i = 1,\dots, N$ the description of object with index i.

The definition of good test is based on two mapping $2^S \rightarrow 2^T$ and $2^T \rightarrow 2^S$ determined as follows:

t = val(s) = {intersection of all t_i: $t_i \subseteq T$, $i \in s$} and

s = obj(t) = {i: $i \in S$, $t \subseteq t_i$}.

Of course, we have obj(t) = {intersection of all $s(A)$: $s(A) \subseteq S$, $A \in t$}. Operations val(s), obj(t) are reasoning operations related to discovering the general feature of objects the indices of which belong to s and to discovering the indices of all objects possessing the feature t.

The basic operator of plausible reasoning [3] connecting it with the FCA, is the generalization rule (GR) defined as follows:

generalization_of(t) = t' = val(obj(t)): generalization_of(s) = s' = obj(val(s)).

Galois Lattice consists of closed pairs (s, t) called concepts and defined by the generalization rule: val(obj(t)) = t, obj(val(s)) = s.

In general, the concept has maximal coverage of examples of some dataset by a given itemset that cannot be extended by a value of any other attribute to get the same coverage.

2.1. Classification (diagnostic) tests

In classification problems, each object has a class label, which is not part of the domain description. Labeling is a kind of partitioning of a data set or an ontology.

Let $S(+)$ and $S(-) = S\backslash S(+)$ be the sets of positive and negative class of objects, respectively.

A diagnostic (classification) test for $S(+)$ is a pair (s, t) such that $t \subseteq T$ (s = obj(t) $\neq \emptyset$), $s \subseteq S(+)$ and $t \not\subset t'$ $\forall t'$, $t' \in S(-)$.

A diagnostic test (s, t), $t \subseteq T$ (s = obj(t) $\neq \emptyset$) is good for $S(+)$ if and only if any extension $s^* = s \cup i$, $i \notin s$, $i \in S(+)$ implies that (s^*, val(s^*)) is not a test for $S(+)$.

It means that if (s, t) is a good test for $S(+)$, then s of it is non-extendable, i. e. adding to s any i from $S(+)$ not belonging to s implies that for val($s \cup i$) there exists such a $t' \in S(-)$ that val($s \cup i$) $\subseteq t'$.

A good test (s, t), $t \subseteq T$ (s = obj(t) $\neq \emptyset$) for $S(+)$ is irredundant (GIRT) if any narrowing $t^* = t \backslash A$, $A \in t$ implies that (obj(t^*), t^*)) is not a test for $S(+)$.

A good test (s, t) for $S(+)$ is maximally redundant (GMRT) if any extension of $t^* = t \cup A$, $A \notin t$, $A \in T$ implies that (obj(t^*), t^*) is a test for $S(+)$, but not a good one.

To align the above original definitions with the FCA terminology, we introduce the coefficient of confidence (Confidence, for short) as follows:

Let (s, t) be a concept. Confidence = $|s+|/|s|$, where $s+ = s \cap S(+)$ and $|q|$ denotes the power of set q.

Irredundant test (IT) for S+ is a closed itemset (CI) (s, t) if any narrowing $t^* = t \backslash A$, $A \in t$ implies that obj(t^*) $\neq s$ and obj(t^*) $\supset s$.

The goodness of a diagnostic test can be characterized by its Confidence.

2.2. Functional dependencies

Let U be the set of attributes values of which compose the set T. Functional dependency $X \longrightarrow C$ is a relation between the collection $X \subseteq U$ of attributes and the given classification C of objects into classes $C1$, ..., Ck. Denote by P(X) = {$p1$, $p2$, ..., pm) the partition of S generated by the values of X, where pj, $j = 1, 2, ..., m$, $m \geq k$, are classes of P(X). Each class of P(X) consists of objects having equal values of all the attributes of X.

The definition of functional dependency (FD) between attributes is based on the definition of the relation of partial order over the set of partitions generated by the set of considered attributes. This relation is introduced as follows: P(X) $\leq$ P(Y) iff P(X) $\subseteq$P(Y), X, $Y \subseteq U$.

A pair P(X), P(Y) are said to be in the inclusion relation iff every block of P(X) is contained in one and only one block of P(Y).

If P(X) = P(C), then X is the ideal approximation of classification C or ideal functional test (FT) for C based on a functional dependency. If this condition is not satisfied, then X, $X \subseteq U$, X corresponds to a good approximation of C, if P(X) is the closest to P(C) element of Partition Lattice over a given context, i. e., for all P(Y), $Y \subseteq U$ condition (P(X) $\subset$ P(Y) $\subseteq$ P(C)) implies P(X) = P(Y). In this case, we said that $X \longrightarrow C$ is a FD in U and X is a good FT for C. FD in the form $X \longrightarrow Y$ is known as conditional FD.

In [4], A method is given to transform initial contexts into the contexts for searching for FDs by any algorithm of discovering GMRTs.

2.3. Implicative dependencies as plausible rules of the first type

In this paper, we focus on conceptual knowledge the main elements of which are objects, properties (attribute values), and classifications (attributes). Taking into account that implications express the links between concepts (object $\leftrightarrow$ class, object $\leftrightarrow$ property, property $\leftrightarrow$ class) we deem classification reasoning to be based on using and searching for only one type of logical dependencies, namely, implicative dependencies.

Implicative dependences are the result of GCTs inferring. Consider, for example, a GMRT as a pair (obj(t), t). In this pair, t is a collection of attribute values, $t \subseteq T$, and |obj(t)| is the support of t, and obj(t) $\subseteq S(+)$. Thus, we can form an implicative rule $t \rightarrow S(+)$. This assertion is transformed in a reasoning rule. The left part of this rule is t (a set of values from T) and $S(+)$ can be the name of a class in the classification of S.

Implicative assertions are considered as plausible rules (PR) of the first type. Generally, we have the following rules of the first type (the left part of rules can contain any number of different values from a given context): Implication: a, b, c, ...$\rightarrow$ d. Interdiction or forbidden rule: a, b, c, ... $\rightarrow$ *false* (*never*). This rule can be transformed into several implications such as a, b, ... $\rightarrow$ not c; a, c, ... $\rightarrow$ not b; b, c, ... $\rightarrow$ not a. Compatibility (associations): a, b, c, ... $\rightarrow$ VA, where VA is the frequency of rule's occurrence (related to the confidence of the left part of this rule). Generally, the compatibility rule represents a most frequently observed combination of values. Diagnostic rule: x, $d \rightarrow a$; x, $b \rightarrow$ not a; d, $b \rightarrow$ *false*. For example, d and b can be two values of the same attribute. This rule works when the truth of 'x' has been proven and it is necessary to determine whether 'a' is true or not. If 'x & d' is true, then 'a' is true, but if 'x & b' is true, then 'a' is false. Rule of alternatives: a or $b \rightarrow$ *true* (*always*); a, $b \rightarrow$ *false*. This rule says that 'a' and 'b' cannot be simultaneously true, either 'a' or 'b' can be true but not both. In the rules, a, b, c, d, $\in T$, $x \subset T$.

The plausible reasoning rules of the first type are formed from GCTs (maximally redundant and non-redundant ones). Let X_1, X_2 and Y_1, $Y_2 \subseteq T$ be good maximally redundant and good maximally non-redundant classification tests. Let $x_1 \rightarrow q_1$, $x_2 \rightarrow q_2$, $y_1 \rightarrow q_1$, $y_2 \rightarrow q_2$ be implications, where q_1, $q_2 \in GOAL$, are two different classes of objects. We can form the following forbidden rules: $x_1 \rightarrow$ not q_2, and x_2, $\rightarrow$ not q_1.

The rules of alternative: "a or $b \rightarrow$ *true*; a, $b \rightarrow$ *false*" is indeed the case when a and b are values of the same attribute.

The diagnostic rule can be obtained from two good maximally non-redundant tests. For example, compute $int = y_1 \cap y_2$. Then we have diagnostic rule: 'if int is *true*, then $int \cup (y_1 \backslash int) \rightarrow q_1$; $int \cup (y_2 \backslash int) \rightarrow q_2$.

Compatibilities rules can be obtained from the concepts with the Confidence insignificantly different from 1.

3. Diagnostic Test Machine

Diagnostic Test Machine (DTM) is a software library for finding implicative and functional dependencies in data sets. All dependencies generated by the system are redundant and frequent, until otherwise explicitly declared. In particular, the DTM finds all value-based (like in the Charm algorithm [3]) and attribute-based frequent formal concepts that are independent of the final task. Once the lattice of concepts is found, the DTM generates all good (confident) maximally redundant diagnostic (classification) tests (GMRT) and good FTs for a given classification (partition of objects). This step is task-dependent (Figure 1). It also has the ability to generate all good non-redundant tests from good redundant ones.

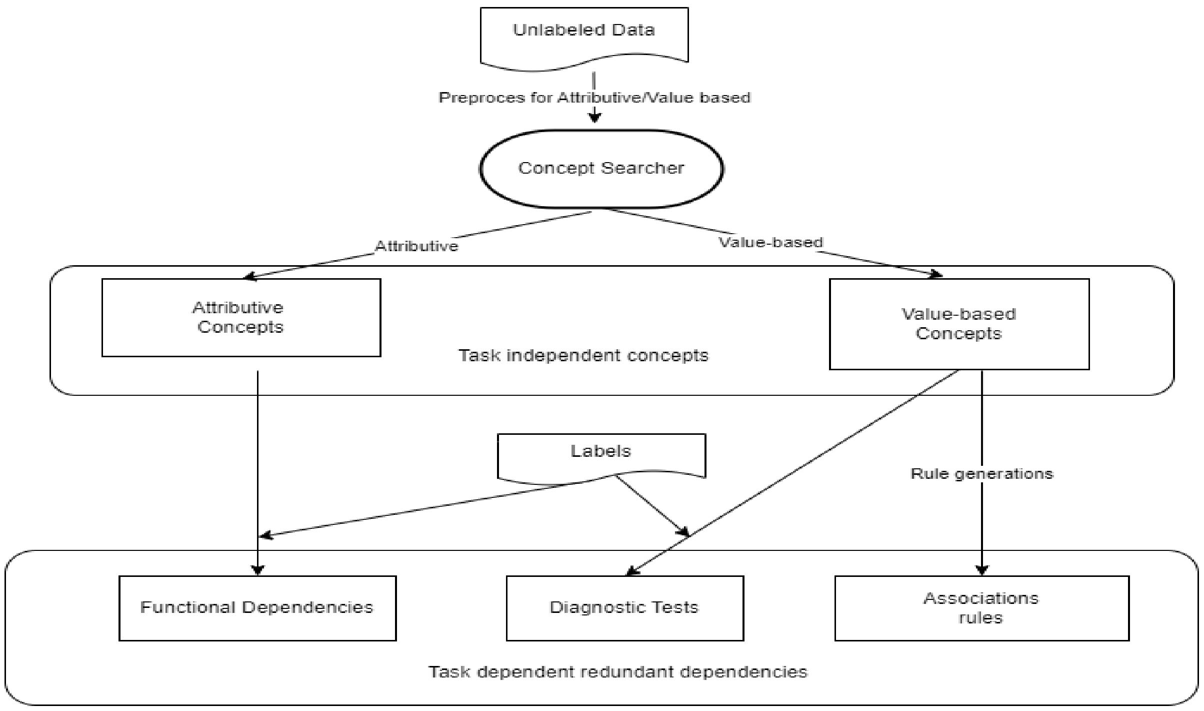

Figure 1: The tasks flow diagram in the DTM

The library is applicable for a number of scenarios and purposes, but mainly:

- to construct FDs and simultaneously the dimensionality reduction in initial data;
- to construct GCTs for classification task.

Below some details for main steps and details of the library implementation are listed.

3.1. Row data preprocessing.

The library supports categorical (ordinal/nominal) and numerical (discrete/continuous) domains for attributes. One of the main problems in this area is working with numeric attributes. In addition to the trivial simple partitioning into equal wide ranges, the library includes additional methods to solve this problem, namely: Minimum Description Length (MDL) [8] and the Kolmogorov-Smirnov algorithm. These methods must be provided with a target partition. There are two obvious options for this: to use a forward-defined classification or, in the case of categorical attributes, to use its composite partitioning. Despite this, there are still questions on this issue.

We transform the row data to dual horizontal - vertical bit vector representation. This allows to work effectively with dense datasets (like "Mushrooms") due to having the equal width records and sparse ones like any store of transactions db.

3.2. Concepts searching algorithm

This algorithm is described in [6]. It is based on the procedure of decomposing the main task into attributive and object subtasks (projections) most fully described in [7].

The root of search tree or initial task (Alg.1, init_task) is built on a given preprocessed training set. It can be an attribute-based or sample (object)-based (transposed) task, which initiates the search from the join (lower bound) or meet (upper bound) of the lattice [14]. The choice of attribute or object together with the lattice traversal strategy provides a powerful basis for implementing various algorithms for the FCA problems. Only a row coverage vector is used for concept representation, which correspondents to the 'rows' vector in the Task structure (see Main structures). The algorithm recursively decomposes the current task into depth-first search subtasks, selecting attribute/value according to selected strategy.

The search tree generates only closed elements of concept lattice (closed itemsets) (Alg. 1, find_concepts) and does not produce any redundant subtasks. The traversal strategy (attribute/sample selection) may vary depending on the task. So, if the task is to find all frequent concepts, the optimal strategy will be to select the attribute with the minimum support, but when the task is diagnostic, the strategy with the maximum support will be much more reliable. Once the attribute is selected, the subtask corresponding to some concept is formed using the generalization rule (Alg. 1, sub_task). Of course, the search tree could achieve the same task in several ways. The logic of cutting off a dead-end or solved subtasks and stopping the search is also encapsulated (Alg. 1, add_concept).

Some of the main structures and operations on them are defined below. The "." operator provides access to the structure fields.

Main structures
BitVector Operations: & - bitwise and operation $\vee$ - bitwise or operation $\neg$ - bitwise not operation weight() - sum of all bit values

```
DualBitMatrix – structure effectively supports dataset (DS) representation (horizontal and vertical)

    Fields:
      rows : [BitVector] // set of BitVector corresponding to each example in the DS
      cols :   [BitVector] // transposed rows BitVector set  for each attribute in the DS
      height: int // number of rows
      width:  int // number of cols

    Operations:

      BitVector & (BitVector vector)   // returns intersection of given  rows/cols
      BitVector ∨ (BitVector vector)   // returns union of given rows/cols
```

```
Task – subset of DS (DualBitMatrix) in both dimension. It is corresponds to concept and defined
    by the generalizing rule: val(obj(t)) = t

    Fields:
        cols   : BitVector      // cols subset of the DS
        rows  : BitVector    // rows subset of the DS
        cross : BitVector     // the task rows intersection db.&( rows )
```

```
Lattice –  structure consists of founded concepts and responsible for  search tree pruning

    Fields:
        concepts :   { BitVector } //set of concept
        minsup : int   // minimal support threshold
```

Algorithm 1. Frequent concepts search procedure

```
Input:  db : DualBitMatrix, minsup: int // training set, minimal support
Output: L: Lattice

T = init_task(db)
L = Lattice (∅, minsup)

find_concepts(T, L) begin   // traversal of the task lattice
    if add_concept(L, T.rows) then
        while (sub_T = select_subtask(T, strategi)) is not null do
            find_concepts(sub_T, L)
            T.cols = T.cols & ¬sub_T.cross // removes subtask
        end while
    end if
end find_concepts

init_task(db) begin
     rows  = ¬BitVector(db.height)
     cols  = ¬BitVector(db.width)
     cross = db.&(rows)
     return Task(rows ,  cols, cross)
end init_task

select_subtask(Task t, strategy) begin
      a =  find_best_sub_task(t, strategy) // return best attribute according the strategy
     if a >=0 then
```

```
            return sub_task(t, a)
        else
            return null
        end if
end select_subtask

sub_task(Task t, int a) begin // get sub task/concept by given attribute
        rows  := t.rows  &  db.cols[a]         // t = obj(a)
        cross := db.&(rows)                     // s = val(t)
        cols  := t.cols  & ¬cross
        return Task(rows ,  cols, cross)
end sub_task

add_concept(L, c) begin
        support = c.weight()
        if support < L.minsup then
            return false
        else if c ∈ L.concepts then // all subtask were solved
            return false
        end if
        L.concepts= L.concepts ∪ {c}
        return true
end add_ concept
```

3.3. Generator of tests for given classification with maximal confidence

Once we have all frequent concepts, obtaining all tests with maximal confidence (MCTs) (frequent implications) is as trivial as intersecting of the goal vector (bit vector with ones for the target class objects) with the extent of concept and thresholding the result by the minimum confidence parameter (Alg. 2).

Algorithm 2. Concept to maximal test procedure

```
Input:
goal : BitVector,  concept : BitVector
        minconf: float [0:1]  // minimal confidence
Output:
        implication : (concept, confidence)-> goal

concept_to_implication(goal, concept , minconf) begin
        concept_weight = concept. weight()
        goal_concept_weight = (concept & goal).weight()
        confidence =  goal_concept_weight  / concept_weight
        if( confidence >= minconf) then
            return (concept, confidence)-> goal
        else then
            return null
        end if
end concept_to_implication
```

3.4. Diagnostic task

The diagnostic or classification task is to assign an unlabeled example to a certain class for which tests were obtained in the previous step. One problem here is that the tests are generally redundant.

But the task to generate all non-redundant tests has the exponential complexity. Therefore, the DTM bypasses the problem with a simple check below (Alg.3).

As mentioned earlier, the concept has a dual representation of objects/attributes, and the algorithms described above use only the first one. Of course, the diagnostic task requires the second representation, the creation of which is trivial for the given training dataset and has been omitted here. Therefore, the test structure used below has both representations (rows and columns).

Algorithm 3. Procedure for checking the equivalence of coverings

```
Input: sample: BitVector, Test test, BitMatrix db
Output: Boolean

test_sample(sample, test,  db) begin
    BitVector u  = test.cols & sample;
    return test.rows = db.&(u);
end  test_sample
```

The project code and some other datasets can be found at https://gitlab.com/shagalovv/dtm

3.5. Example

To illustrate the process, we use a small dataset from [3] (Table 1). The original data is transformed into an internal dense representation with an additional column, which is the external classification. The classification column will be masked during the concept discovery stage. Now, the Examples are presented in Tables 2-6.

Table 1: Raw dataset

Object Index	Itemset
1	A C T W 0
2	C D W 0
3	A C T W 0
4	A C D W 1
5	A C D T W 1
6	C D T 1

Value-based dependencies are in Table 2.

Table 2: Closed frequent itemsets (min confidence = 1)

N	Support	Objects	Itemset
1	1	5	ACDTW
2	3	1 3 5	ACTW
3	2	4 5	ACDW
4	3	2 4 5	CDW
5	4	1 3 4 5	ACW
6	5	1 2 3 4 5	CW
7	2	5 6	CDT
8	4	1 3 5 6	CT
9	4	2 4 5 6	CD
10	6	1 2 3 4 5 6	C

Table 3: Frequent tests in the case (min confidence = 1)

N	Support	Confidence	Goal	Tests (intents)
1	1	1	GOAL[1]	A C D T W
2	2	1	GOAL[1]	A C D W
3	2	1	GOAL[1]	C D T

Functional dependencies: dense source data is transformed to the data for functional dependencies search (with no duplicates for brevity) [4]. As in value-based task, the classification column will be masked on concepts discovery stage.

Table 4: Transformed raw data for inferring FDs

N	A	C	D	T	W	GOAL
1	0	1	1	0	0	0
2	0	1	0	1	0	0
3	1	1	0	0	1	0
4	0	1	1	0	1	0
5	1	1	0	1	1	0
6	0	1	1	0	0	1
7	0	1	1	1	0	1
8	0	1	0	0	1	1
9	1	1	1	0	1	1
10	1	1	0	1	1	1

Table 5: Intents of concepts containing frequent functional dependencies (min support = 1)

N	Support	Objects	Intents of concepts
1	1	7	CDT
2	1	9	ACDW
3	2	4 9	CDW
4	5	1 4 6 7 9	CD
5	2	5 10	ACTW
6	4	2 5 7 10	CT
7	4	3 5 9 10	ACW
8	6	3 4 5 8 9 10	CW
9	10	1 2 3 4 5 6 7 8 9 10	C

Table 6: Frequent functional dependencies for the given classification (min confidence = 1)

N	Support	Confidence	Goal	Left part of dependency
1	1	1	GOAL[0]	C D T
2	1	1	GOAL[0]	A C D W

4. Experiments

For the DTM performance testing experiments, the well-known Mushroom dataset and the lesser-known Adult dataset were used, see Table 7. Both were shuffled and split in a ratio of 80% training set to 20% testing set

Table 7: Datasets description

Data sets	Type	Attributes per Types	Number of Records	Problems
mushrooms	dense	22- categorical + label	8124	missing values
adults	dense	8 - categorical 6 - numerical + label	32561	missing values, class imbalance, repeated samples

The search processes are controlled by a search strategy for selecting subtasks by attributes. Namely, the strategies are: "support (max)" (the choice of attributes with max support, "the unordered or left-to-right choice of attributes" (uno), and "maximum support (min)" (the choice of attributes with min support).

Table 8 shows the results of the search for value-based concepts, and Table 9 shows the results of searching for the diagnostic tests. The number of solved subtask/time in the Table 8 is determined for concept task only. The '-' means the absence of data.

Table 8: Value-based concepts result for min support 1

Data sets	Task dimensions	Number of Concepts	Coverage	Number of Solved subtasks/time(ms): min/uno/max
mushrooms	6499 x 116	212959	6499	301718/ 2189 793889/ 5533 1447275/ 12161
adults	26048 x 146	2037104	26048	2456837/ 52693 7522715/ - 20394837/ -

Table 9: Test search results for min confidence 1

Data sets	Class/members №	Number of Tests	Coverage
Mushrooms	p /3161	76855	3161
	e/ 3338	78867	3338
Adults	≤ 50K/ 19729	683836	17257
	> 50K/ 6319	55822	4260

In Tables 8 and 9, "Coverage" means the number of objects in the given context belonging to at list one of obtained tests.

Table 10 shows the results of the search for both functional dependencies and conditional ones.

Table 10: Functional dependencies search results for min support 1 and confidence 1

Data sets	Task dimensions	Number of Concepts	Number of FDs	Number of Solved sub-tasks /time(ms): min/uno/max
mushrooms	16901x 22	202150	27254	225445/ 3497 332896/ 5391 438333/7161

adults	23879x 14	12288	0	12288/ 248 12288/234 16384/303

5. Plausible rule application

The FCA is certainly one of the most powerful tools for analyzing data and building knowledge models based on the lattice of formal concepts extracted from a training context. Remarkable introduction to the FCA and its applications in the information retrieval and related fields is contained in [9].

However, the FCA has a number of drawbacks, one of which should be recognized as the impossibility of directly using formal concepts in the tasks of classifying objects. Computer knowledge structures are traditionally declarative, mechanisms of their using are separated from them and, as a rule, these mechanisms are often fixed.

Currently, various methods for building classifiers are proposed based on concepts extracted from training contexts. These methods use several ideas: 1) forming formal concepts as classifiers and recognizing classes of new objects by navigating through the levels of the conceptual lattice [10, 11]; 2) transition from classifiers constructed by methods other than the FCA to a lattice of formal concepts containing only concepts associated with the decision rules of these classifiers [12].

The first method is quite cumbersome. Essentially, it's about extracting concepts whose extents contain objects of only one class. To do this, the authors in [11] move from the two-digit to the nominal (multivalued) description of objects and introduce the labeling of objects of a context. Now, a nominal (multi-valued) context is a quadruple $\langle I_{nom}, A_{nom}, \varsigma, R_{nom}\rangle$, where I_{nom} is the set of n_{nom} instances, A_{nom} is the set of m_{nom} attributes, ς is the set of attributes' values, R_{nom} is a relation defined between I_{nom}, A_{nom} and ς. R_{nom} is a set of triples.

A similar idea, but more easily implemented, is given in [13]. In [12], the decision tree is considered as a set of classification rules and a method for transforming the constructed decision tree over a given context into an isomorphic lattice of concepts is proposed.

The extraction of GCTs is the basis for obtaining the rules of classification plausible reasoning.

Consider plausible reasoning rules of the second type and a model of plausible inference.

Let x be a pattern (a set of true values of some attributes observed simultaneously). Our goal is to define the target value, i.e. the label of a possible class of objects to which this pattern can belong. Deductive steps of reasoning consist of inferring consequences from some observed values with the use of the rules of the first kind (i.e., knowledge).

Using implication: Let r be an implication, left(r) and right(r) be the left and right part of r, respectively. If left(r) $\subseteq x$, then x can be extended by right(r): $x \leftarrow x \cup$ right(r). Using interdiction: Let r be an implication $x \rightarrow$ not k. If left(r) $\subseteq x$, then k is the forbidden value for all extensions of x. Using compatibility: Let r = '$a, b, c, \ldots \rightarrow k$, VA (confidence of the rule)'. If left(r) $\subseteq x$, then k can be used to extend x along with the calculated value VA for this extension. Using diagnostic rules: Let r be a diagnostic rule such as 'x, $d \rightarrow a$; x, $b \rightarrow$ not a', where 'x' is *true*, and 'a', 'not a' are hypotheses or possible values of some attribute. Using diagnostic rule implies to infer whether d or b is true.

The rules listed above are the rules of "forward inference". Another way to include the first-type rules in natural reasoning can be called "backward inference". Generating hypothesis or abduction rule: Let r be an implication y $\rightarrow k$. Then the following hypothesis is generated "if k is *true*, then y may be *true*".

When applied, the above rules generate the reasoning, which is not demonstrative. The purpose of reasoning is to infer all possible hypotheses on the value of some target attribute. It is essential that these hypotheses do not contradict with knowledge (the first type rules) and the observable real situation under which the reasoning takes place. Inference is reduced to obtain all intrinsically consistent extensions of x, in which the number of involved attributes is maximum possible and there are no prohibited pairs of values in such extensions. All hypotheses have different admissibility determined by the quantity and "quality" of compatibility rules involved in inferring each of them.

As a result of learning, we can form the following knowledge bases (KB): the Attribute Base (AtB), containing the relations between problem domain concepts (Ontology), and the Assertion Base (AsB), containing the assertions, formulated in terms of the concepts, and the rules of the first type obtained from training context. Let a request to the KB be: SEARCHING VALUE OF class of object IF (an observable pattern of object's values = *x*).

Step 1. Select all the assertions *as*, *as* ∈ AsB containing at least one value from the request *x*. Step 2. Delete from the set of selected assertions all of these that contradict with the request. Assertion contradicts with the request if it contains the value of an attribute which is different from the value of this attribute in the request. Step 3. Select the values of attributes appearing in remaining assertions. If this set of values contains several hypotheses (several names of target classes), an attempt is made to refute one of the hypotheses. For this goal, it is necessary to find a forbidden rule containing one of the hypotheses, some subset of values from the request and does not contain any other value. Step 4. If we have not a hypothesis or we cannot refute the existing hypotheses, then an attempt is made to find a value of some attribute that is not in the request (in order to extend the request). For this goal, it is necessary to find an assertion (implication) that contains a subset of values from the request and one and only one value of some new attribute which are not in the request. For extending request, the compatibilities rules can also be used. The extending obtained must not contain any forbidden set of values. Step 5. Forming the extended request. Steps 1, 2, 3, 4 are repeated.

The process of pattern recognition can require inferring new rules of the first type from data when i) the result of reasoning contains several hypotheses and it is impossible to choose one and only one of them (uncertainty), and ii) it is impossible to obtain any hypothesis.

6. Conclusion

In this paper, a system for solving formal concept related tasks in real world domains is presented. The main goal of the system is the searching for all closed itemsets (concepts). Constructing Galois lattice of concepts allows to additionally generate GCTs and approximating FDs for given classifications on a given data set. In general, these tasks are based on ordinal procedure for shallow or deep machine learning for classifications. We show that the FCA is closely related to modeling plausible classification reasoning.

In future work, we plan to implement a fully scalable incremental version of the algorithm for distributed computing to cope with truly "big data" problems. We plan also to improve the lattice navigation to reduce some dead ends in the context of probabilistic reasoning.

Another urgent task is to create a system for generating plausible reasoning rules and models of plausible reasoning based on constructing and browsing a lattice of concepts.

References

[1] B. Ganter and K. Reuter. Finding all closed sets: A general approach. Oder, Vol. 8, N 3, pp. 389-290, 1991.

[2] D. Borchmann. Next-Closure Algorithm – enumerating semilattice elements for a generating set. CLA 2012, pp. 9-20, 2012.

[3] Zaki, Mohammed & Hsiao, Ching-Jiu. CHARM: An efficient algorithm for Closed Itemset Mining, in Proceeding of the Second SIAM International Conference on Data Mining, 2002, pp. 457-473. doi: 10.1137/1.9781611972726.27.

[4] Naidenova, X., Good Classification Tests as Formal Concepts, in: F. Domenach, D.I. Ignatov, and Poelmans (Eds.): ICFCA 2012, LNAI 7278, Springer-Verlag Berlin Heidelberg 2012, 226-226. doi: 10.1007/978-3-642-29892-9.

[5] Naidenova, X, An Incremental learning algorithm for inferring logical rules from examples in the framework of the common reasoning process, in: Triantaphyllow E., and Felici, G. (Eds.), Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, Massive Computing Series, Springer, Heidelberg, Germany, 2006, pp. 89-147.

[6] Naidenova, X. A., Plaksin, M.V., and Shagalov, V. L., Inductive inferring all good classification tests, in: Valkman J. (Ed.), "Knowledge-Dialog-Solution", Proceedings of the International Conference, volume 1, Kiev, Institute of Applied Informatics Press, 1995, pp. 79–84.

[7] Naidenova, X., Parkhomenko V. Contribution to attributive or object sub-contexts in inferring good maximally redundant tests. Discrete Applied Mathematics, Vol. 273, pp. 217-231, 2020.

[8] Fayyad, Usama M., and Keki B. Irani, Multi-interval discretization of continuous valued attributes for classification learning, IJCAI, Vol. 93, N 2, pp. 1022-1027, 1993.

[9] Dmitry I. Ignatov, Introduction to Formal Concept Analysis and its Applications in Information Retrieval and Related Fields. arXiv: 1703.02819v1[csIR]8Mar2017.

[10] Hayfa AZIBI, Nida Meddouri and Mondher Maddouri, Survey on formal concept analysis based supervised classification techniques, in A.I. Tallón-Ballesteros and C. Chen (Eds.), Mchine learning and Artificial Intelligence, the author and IOS Press, 2020, pp. 21-29. doi:10.3233/FAIA200762.

[11] Nida Meddouri and Mondher Maddouri, efficient Closure Operators for FCA Based Classification, International Journal of Artificial Intelligence and Machine Learning, vol. 10, no. 2, pp 79-98, 2020. doi: 10.4018/IJAIML.2020070105.

[12] Egor Dudyrev, Sergey O. Kuznetsov, Decision Concept Lattice vs. Decision Trees and Random Forests, arXiv: 2106.00387v1[csLG]1jun2021 (pp.1-8).

[13] T. Makhalova, S.O. Kuznetsov, and Amedeo Napoli, Closure structure: a deeper insight, in S.O. Kuznetsov, A. Napoli, and S. Rudolph (Eds.), Proceedings of Fights International Workshop "What can FCA do for Artificial Intelligence?" 2020, pp. 45-55.

[14] Davey, B. A. Introduction to Lattices and Order. Cambridge University Press, 2002.