An Algorithm for the Multi-Relational Boolean Factor Analysis based on Essential Elements

Martin Trnecka, Marketa Trneckova

Data Analysis and Modeling Lab (DAMOL)
Department of Computer Science, Palacky University, Olomouc
martin.trnecka@gmail.com, marketa.trneckova@gmail.com

Abstract. The Multi-Relational Boolean factor analysis is a method from the family of matrix decomposition methods which enables us analyze binary multi-relational data, i.e. binary data which are composed from many binary data tables interconnected via relation. In this paper we present a new Boolean matrix factorization algorithm for this kind of data, which use the new knowledge from the theory of the Boolean factor analysis, so-called essential elements. We show on real dataset that utilizing essential elements in the algorithm leads to better results in terms of quality and the number of obtained multi-relational factors.

1 Introduction

The Boolean matrix factorization (or decomposition), also known as the Boolean factor analysis, has gained interest in the data mining community. Methods for decomposition of multi-relational data, i.e. complex data composed from many data tables interconnected via relations between objects or attributes of this data tables, were intensively studied, especially in the past few years. Multi-relational data is a more truthful and therefore often also more powerful representation of reality. An example of this kind of data can be an arbitrary relational database. In this paper we are focused on the subset of multi-relational data, more precisely on the multi-relational Boolean data. In this case data tables and relations between them contain only 0s and 1s.

It is important to say that many real-word data sets are more complex than one simple data table. Relations between this tables are crucial, because they carry additional information about the relationship between data and this information is important for understanding data as a whole. For this reason methods which can analyze multi-relational data usually takes into account relations between data tables unlike classical Boolean matrix factorization methods which can handle only one data table.

The Multi-Relational Boolean matrix factorization (MBMF) is used for many data mining purposes. The basic task is to find new variables hidden in data, called multi-relational factors, which explain or describe the original input data. There exist several ways how to represent multi-relational factors. In this work

we adopt settings from [7], where is the multi-relational factor represented as an ordered set of classic factors from data tables, always one factor from each data table. The fact, that classic factors are connected into multi-relational factor is matter of semantic of relation between data tables.

The main problem is how to connect classic factors into one multi-relational. The main aim of this work is to propose a new algorithm which utilize so-called essential elements from the theory of Boolean matrices. The essential elements provide information about factors which cover a particular part of data tables. This information can be used for a better connection of classic factors into one multi-relational factor.

Another thing is the number of obtained factors. In classical settings we want the number of obtained factors as small as a possible. In the literature can be found two main views on this requirement. In the first case we want to obtain the particular number of factors. In the second case we want to obtain factors that explain prescribed portion of data. In both cases we want to obtain the most important factors. For more details see [1]. We emphasize this fact and we reflect it in designing of our algorithm. Both views can be transferred to multi-relational case. The first one is straightforward, the second one is a little bit problematic because multi-relational factors may not be able explain the whole data. This is correct, because multi-relational factors carry different information than classical factors. We discuss this issue later in the paper.

2 Preliminaries and basic notions

We assume familiarity with the basic notions of the Formal concept analysis [4], which provides a basic framework for dealing with factors and the Boolean matrix factorization (BMF) [2]. The main goal of classical BMF is to find a decomposition $C = A \circ B$, where C is input data table, A represent object-factor data table (or matrix) and B represent factor-attribute data table (or matrix). The product $\circ$ is the Boolean matrix product, defined by

$$(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{il} \cdot B_{lj}, \tag{1}$$

where $\bigvee$ denotes maximum (truth function of logical disjunction) and $\cdot$ is the usual product (truth function of logical conjunction). Decomposition C into $A \circ B$ corresponds to discovery factors which explain the data. Factors in classical BMF can be seen as formal concepts [2], i.e. entity with the extent part and the intent part. This leads to clear interpretation of factors. Another benefit of using FCA as a basic framework is that matrices A and B can be constructed from the subset of all formal concepts. Let

$$\mathcal{F} = \{\langle A_1, B_1 \rangle, \ldots, \langle A_k, B_k \rangle\} \subseteq \mathcal{B}(X, Y, C),$$

where $\mathcal{B}(X, Y, C)$ represents a set of all formal concepts of data table, which can be seen as a formal context $\langle X, Y, C \rangle$, where X is a set of objects, Y is a set of attributes and C is a binary relation between X and Y. Matrices A and B are constructed in the following way:

$$(A)_{il} = \begin{cases} 1 \text{ if } i \in A_l \\ 0 \text{ if } i \notin A_l \end{cases} \quad (B)_{lj} = \begin{cases} 1 \text{ if } j \in B_l \\ 0 \text{ if } j \notin B_l \end{cases}$$

for $l = 1, \ldots, k$. In other words, A is composed from characteristic vectors A_l. Similarly for B.

In a multi-relation environment we have a set of input data tables C_1, $C_2, \ldots C_n$ and a set of relations $\mathcal{R}_{ij}$, where $i, j \in \{1, \ldots, n\}$, between C_i and C_j. The multi-relation factor on data tables C_1, $C_2, \ldots C_n$ is an ordered n-tuple $\langle F_1^{i_1}, F_2^{i_2}, \ldots F_n^{i_n} \rangle$, where $F_j^{i_j} \in \mathcal{F}_j$, $j \in \{1, \ldots, n\}$ ($\mathcal{F}_j$ denotes a set of classic factors of data table C_j) and satisfying relations $\mathcal{R}_{C_l C_{l+1}}$ or $\mathcal{R}_{C_{l+1} C_l}$ for $l \in \{1, \ldots, n-1\}$.

Example 1. Let us have two data tables C_1 (Table 1) and C_2 (Table 2). Moreover, we consider relation $\mathcal{R}_{C_1 C_2}$ (Table 3) between objects of the first data table and attributes of the second one.

Table 1: C_1

	a	b	c	d
1		×	×	×
2	×		×	
3		×		×
4	×	×	×	×

Table 2: C_2

	e	f	g	h
5	×			×
6		×	×	
7	×	×	×	
8			×	×

Table 3: $\mathcal{R}_{C_1 C_2}$

	e	f	g	h
1		×	×	
2	×		×	
3	×	×		×
4	×	×	×	×

Classic factors of data table C_1 are for example: $F_1^{C_1} = \langle \{1, 4\}, \{b, c, d\} \rangle$, $F_2^{C_1} = \langle \{2, 4\}, \{a, c\} \rangle$, $F_3^{C_1} = \langle \{1, 3, 4\}, \{b, d\} \rangle$ and factors of the second table C_2 are: $F_1^{C_2} = \langle \{6, 7\}, \{f, g\} \rangle$, $F_2^{C_2} = \langle \{5\}, \{e, h\} \rangle$, $F_3^{C_2} = \langle \{5, 7\}, \{e\} \rangle$, $F_4^{C_2} = \langle \{8\}, \{g, h\} \rangle$. These factors can be connected with using a relation $\mathcal{R}_{C_1 C_2}$ into multi-relational factors in several ways. In [7] were introduced three approaches how to manage this connections. We use the narrow approach from [7], which seems to be the most natural, and we obtain two multi-relational factors $\langle F_1^{C_1}, F_1^{C_2} \rangle$ and $\langle F_3^{C_1}, F_1^{C_2} \rangle$. The idea of the narrow approach is very simple. We connect two factors $F_i^{C_1}$ and $F_j^{C_2}$ if the non-empty set of attributes (if such exist), which are common (in the relation $\mathcal{R}_{C_1 C_2}$) to all objects from the first factor $F_i^{C_1}$, is the subset of attributes of the second factor $F_j^{C_2}$.

The previous example also demonstrate the most problematic part of MBMF. Usually is problematic to connect all factors from each data table. The result of this is a small number of connections between them. This leads to problematic selection of quality multi-relational factors. The reason for a small number of connections between factors is that classic factors are selected without taking relation into account.

Another very important notion for our work are so-called essential elements presented in [1]. Essential elements in the Boolean data table are entries in this data table which are sufficient for covering the whole data table by factors

(concepts), i.e. if we take factors which cover all these entries, we automatically cover all entries of the input data table. Formally, essential elements in the data table $\langle X, Y, C\rangle$ are defined via minimal intervals in the concept lattice. The entry C_{ij} is essential iff interval bounded by formal concepts $\langle i^{\uparrow\downarrow}, i^{\uparrow}\rangle$ and $\langle j^{\downarrow}, j^{\downarrow\uparrow}\rangle$ is non-empty and minimal w.r.t. $\subseteq$ (if it is not contained in any other interval). We denote this interval by $\mathcal{I}_{ij}$. If the table entry C_{ij} is essential, then interval $\mathcal{I}_{ij}$ represents the set of all formal concepts (factors) which cover this entry. Very interesting property of essential elements, which is used in our algorithm, is that is sufficient take only one arbitrary concept from each interval to create exact Boolean decomposition of $\langle X, Y, C\rangle$. For more details about essential elements we refer to [1].

3 Related work

There are several papers about classical BMF [1, 2, 5, 8, 10, 12], but this methods can handle only one data table. In the literature, we can found a wide range of theoretical and application papers about the multi-relation data analysis (see overview [3]), but many times were shown that these approaches are suitable only for ordinal data. The multi-relational Boolean factor analysis is more specific. The most relevant paper for our work is [7], where was introduced the basic idea that multi-relational factors are composed from classical factors which are interconnected via relation between data tables. There were also introduced three approaches how to create multi-relational factors, but an effective algorithm is missing.

The Boolean multi-relational patterns and its extraction are subject of a paper [11]. Differently from our approach data are represented via k-partite graphs. There are considered only relations between attributes and data tables contain only one single attribute. Patterns in [11] are different from our multi-relational factors (are represented as k-clique in data) and also carry different information. In [11] there is also considered other kind of measure of quality of obtained patterns which is based on entropy.

Another relevant work is [6] where were introduced the Relational Formal Concept Analysis as a tool for analyzing multi-relational data. Unlike from [6] our approach extracts a different kind of patterns. For more details see [7]. MBMF is mentioned indirectly in a very specific and limited form in [9] as the Joint Subspace Matrix Factorization.

Generally the idea of connection patterns from various data tables is not new. It can be found in the social network analysis or in the field of recommendation systems. The main advantage of our approach is that patterns are Boolean factors that carry significant information and the second important advantage is that we deliver the most important factors (factors which describe the biggest portion of input data) before others, i.e. the first obtained factor is the most important.

4 Algorithm for MBMF

Before we present the algorithm for the MBMF we show on a simple example basic ideas that are behind the algorithm. For this purpose we take the example from the previous part. As we mentioned above if we take tables C_1, C_2 and relation $R_{C_1C_2}$, we obtain with the narrow approach two connections between factors, i.e. two multi-relational factors. These factors explain only 60 percent of data. There usually exist more factorizations of Boolean data table. Factors in our example were obtained with using GreConD algorithm from [2]. GreConD algorithm select in each iteration a factor which covers the biggest part of still uncovered data. Now we are in the situation, where we want to obtain a different set of factors, with more connections between them. For this purpose we can use essential elements. Firstly we compute essential parts of C_1 (denoted $Ess(C_1)$) and C_2 (denoted $Ess(C_1)$). With the essential part of data table we mean all essential elements (tables 1 and 2).

Table 4: $Ess(C_1)$

	a	b	c	d
1			×	
2	×			
3		×		×
4				

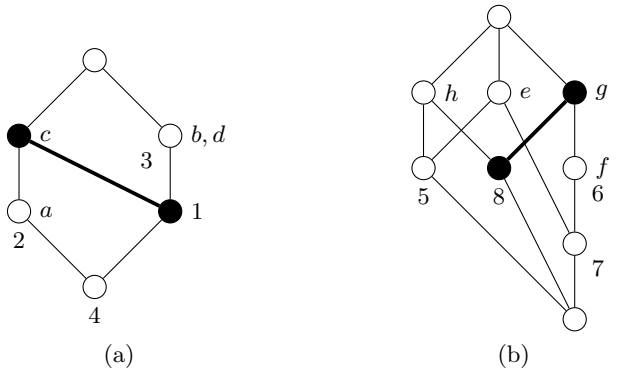
Table 5: $Ess(C_2)$

	e	f	g	h
5	×			×
6		×		
7	×			
8			×	×

Each essential element in $Ess(C_1)$ is defined via interval in concept lattice of C_1 (Fig. 1a) and similarly for essential elements in $Ess(C_2)$ (Fig 1b). In Fig. 1a is highlighted interval $\mathcal{I}_{1c}$ corresponding to essential element $(C_1)_{1c}$. In Fig. 1b is highlighted interval corresponding to essential element $(C_2)_{8g}$. Let us note that concept lattices here are only for illustration purpose. For computing $Ess(C_1)$ and $Ess(C_2)$ is not necessary to construct concept lattices at all. Now, if we use the fact that we can take an arbitrary concept (factor) from each interval to obtain a complete factorization of data table, we have several options which concepts can be connect into one. More precisely we can take two intervals and try to connect each concept from the first interval with concepts from the second one. Again, we obtain full factorization of input data tables, but now we can select factors with regard to a relation between them.

For example, if we take highlighted intervals, we obtain possibly four connections. First highlighted interval contains two concepts $c_1 = \langle\{1,2,4\},\{c\}\rangle$ and $c_2 = \langle\{1,4\},\{b,c,d\}\rangle$. Second consist of concepts $d_1 = \langle\{6,7,8\},\{g\}\rangle$ and $d_2 = \langle\{8\},\{g,h\}\rangle$. Only two connections ($c_1$ with d_1 and c_1 with d_2) satisfy relation $\mathcal{R}_{C_1C_2}$, i.e. can be connected.

For two intervals it is not necessary to try all combination of factors. If we are not able to connect concept $\langle A,B\rangle$ from the first interval with concept $\langle C,D\rangle$ from the second interval, we are not able connect $\langle A,B\rangle$ with any concept $\langle E,F\rangle$ from the second interval, where $\langle C,D\rangle \subseteq \langle E,F\rangle$. Also if we are not



(a) (b)

Fig. 1: Concept lattices of C_1 (a) and C_2 (b)

able to connect concept $\langle A, B\rangle$ from the first interval with concept $\langle E, F\rangle$ from the second interval, we are not able connect any concept $\langle C, D\rangle$ from the first interval, where $\langle C, D\rangle \subseteq \langle A, B\rangle$, with concept $\langle E, F\rangle$. Let us note that $\subseteq$ is classical subconcept-superconcept ordering.

Even if we take this search space reduction into account, search in this intervals is still time consuming. We propose an heuristic approach which takes attribute concepts in intervals of the second data table, i.e. the bottom elements in each interval. In intervals of the first data table we take greatest concepts which can be connected via relation, i.e. set of common attributes in relation is non-empty. The idea behind this heuristic is that a bigger set of objects possibly have a smaller set of common attributes in a relation and this leads to bigger probability to connect this factor with some factor from the second data table, moreover, if we take factor which contains the biggest set of attributes in intervals of the second data table.

Because we do not want to construct the whole concept lattice and search in it, we compute candidates for greatest element directly from relation $\mathcal{R}_{C_1C_2}$. We take all objects belonging to the top element of interval $\mathcal{I}_{ij}$ from the first data table and compute how many of them belong to each attribute in the relation. We take into account only attributes belonging to object i. We take as candidate the greatest set of objects belonging to some attribute in a relation, which satisfies that if we compute a closure of this set in the first data table, resulting set of objects do not have empty set of common attributes in a relation.

Applying this heuristic on data from the example, we obtain three factors in the first data table, $F_1^{C_1} = \langle\{2, 4\}, \{a, c\}\rangle$, $F_2^{C_1} = \langle\{1, 3, 4\}, \{c, d\}\rangle$, $F_3^{C_1} = \langle\{1, 2, 4\}, \{c\}\rangle$ and four factors $F_1^{C_2} = \langle\{5\}, \{e, h\}\rangle$, $F_2^{C_2} = \langle\{6, 7\}, \{f, g\}\rangle$, $F_3^{C_2} = \langle\{7\}, \{e, f, g\}\rangle$, $F_4^{C_2} = \langle\{8\}, \{g, h\}\rangle$ from the second one. Between this factors, there are six connections satisfying the relation. These connections are shown in table 6.

We form multi-relational factors in a greedy manner. In each step we connect factors, which cover the biggest part of still uncovered part of data tables C_1 and

Table 6: Connections between factors

	$F_1^{C_2}$	$F_2^{C_2}$	$F_3^{C_2}$	$F_4^{C_2}$
$F_1^{C_1}$			×	
$F_2^{C_1}$		×	×	
$F_3^{C_1}$		×	×	×

C_2. Firstly, we obtain multi-relational factor $\langle F_2^{C_1}, F_2^{C_2} \rangle$ which covers 50 percent of the data. Then we obtain factor $\langle F_3^{C_1}, F_4^{C_2} \rangle$ which covers together with first factor 75 percent of the data and last we obtain factor $\langle F_1^{C_1}, F_3^{C_2} \rangle$. All these factors cover 90 percent of the data. By adding other factors we do not obtain better coverage of input data. These three factors cover the same part of input data as six connections from table 6.

Remark 1. As we mentioned above and what we can see in the example, multi-relational factors are not always able to explain the whole data. This is due to nature of data. Simply there is no information how to connect some classic factors, e.g. in the example no set of objects from C_1 has in $\mathcal{R}_{C_1C_2}$ a set of common attributes equal to $\{e, h\}$ (or only $\{e\}$ or only $\{h\}$). From this reason we are not able to connect any factor from C_1 with factor $F_1^{C_2}$.

Remark 2. In previous part we explain the idea of the algorithm on a object-attribute relation between data tables. It is also possible consider different kind of relation, e.g. object-object, attribute-object or attribute-attribute relation. Without loss of generality we present the algorithm only for the object-attribute relation. Modification to a different kind of relation is very simple.

Now we are going to describe the pseudo-code (Algorithm 1) of our algorithm for MBMF. Input to this algorithm are two Boolean data tables C_1 and C_2, binary relation $\mathcal{R}_{C_1C_2}$ between them and a number $p \in [0, 1]$ which represent how large part of C_1 and C_2 we want to cover by multi-relational factors, e.g. value 0.9 mean that we want to cover 90 percent of entries in input data tables. Output of this algorithm is a set $\mathcal{M}$ of multi-relational factors that covers the prescribed portion of input data (if it is possible to obtain prescribed coverage). The first computed factor covers the biggest part of data.

First, in lines 1-2 we compute essential part of C_1 and C_2. In lines 2-4 we initialize variables U_{C_1} and U_{C_2}. These variables are used for storing information about still uncovered part of input data. We repeat the main loop (lines 5-18) until we obtain a required coverage or until it is possible to add new multi-relational factors which cover still uncovered part (lines 12-14).

In the main loop for each essential element we select the best candidate from interval $\mathcal{I}_{ij}$ from the first data table in the greedy manner described in the algorithm idea, i.e. we take the greatest concept which can be connected via relation. Than we try to connect this candidate with factors from the second data table. We compute cover function and we add to $\mathcal{M}$ the multi-relational factor maximizing this coverage.

In lines 16-17 we remove from U_{C_1} and U_{C_2} entries which are covered by actually added multi-relational factor.

Algorithm 1: Algorithm for the multi-relational Boolean factors analysis

Input: Boolean matrices C_1, C_2 and relation $R_{C_1C_2}$ between them and $p \in [0,1]$
Output: set $\mathcal{M}$ of multi-relational factors

```
1  E_{C_1} ← Ess(C_1)
2  E_{C_2} ← Ess(C_2)
3  U_{C_1} ← C_1
4  U_{C_2} ← C_2
5  while (|U_{C_1}| + |U_{C_2}|)/(|C_1| + |C_2|) ≥ p do
6  |   foreach essential element (E_{C_1})_{ij} do
7  |   |   compute the best candidate ⟨a, b⟩ from interval I_{ij}
8  |   end
9  |   ⟨A, B⟩ ← select one from set of candidates which maximize cover of C_1
10 |   select non-empty row i in E_{C_2} for which is A^{↑R_{C_1C_2}} ⊆ (C_2)_{i_-}^{↓↑C_2} and which
   |   maximize cover of C_1 and C_2
11 |   ⟨C, D⟩ ← ⟨(C_2)_{i_-}^{↑↓C_2}, (C_2)_{i_-}^{↑C_2}⟩
12 |   if value of cover function for C_1 and C_2 is equal to zero then
13 |   |   break
14 |   end
15 |   add ⟨⟨A, B⟩, ⟨C, D⟩⟩ to M
16 |   set (U_{C_1})_{ij} = 0 where i ∈ A and j ∈ B
17 |   set (U_{C_1})_{ij} = 0 where i ∈ C and j ∈ D
18 end
19 return F
```

Our implementation of the algorithm follows the pseudo-code conceptually, but not in details. For example we speed up the algorithm by precomputing candidates or instead computing candidates for each essential elements, we compute candidates for essential areas, i.e. essential elements which are covered by one formal concept.

Remark 3. The input of our algorithm are two Boolean data tables and one relation between them. In general we can have more data tables and relations. Generalization of our algorithm for such input is possible. Due to lack of space we mentioned only an idea of this generalization. For the input data tables $C_1, C_2, \ldots, C_n$ and relations $\mathcal{R}_{C_iC_{i+1}}, i \in \{1, 2, \ldots, n-1\}$ we firstly compute multi-relational factors for C_{n-1} and C_n. Then iteratively compute multi-relational factors for C_{n-2} and C_{n-1}. From this pairs we construct n-tuple multi-relational factor.

We do not make a detail analysis of the time complexity of the algorithm. Even our slow implementation in MATLAB is fast enough for factorization usually large datasets in a few minutes.

5 Experimental evaluation

For experimental evaluation of our algorithm we use in a data minig community well known real dataset MovieLens[1]. This dataset is composed of two data tables that represent a set of users and their attributes, e.g. gender, age, sex, occupation and a set of movies again with their attributes, e.g. the year of production or genre. Last part of this dataset is a relation between this data sets. This relation contains 1000209 anonymous ratings of approximately 3900 movies (3952) made by 6040 MovieLens users who joined to MovieLens in 2000. Each user has at least 20 ratings. Ratings are made on a 5-star scale (values 1-5, 1 means, that user does not like a movie and 5 means that he likes a movie).

Originally data tables Users and Movies are categorical. Age is grouped into 7 categories such as "Under 18", "18-24", "25-34", "35-44", "45-49", "50-55" and "56+". Sex is from set {Male, Female}. Occupation is chosen from the following choices: "other" or not specified, "academic/educator", "artist", "clerical/admin", "college/grad student", "customer service", "doctor/health care", "executive/managerial", "farmer", "homemaker", "K-12 student", "lawyer", "programmer", "retired", "sales/marketing", "scientist", "self-employed", "technician/engineer", "tradesman/craftsman", "unemployed" and "writer". Film genres are following: "Action", "Adventure", "Animation", "Children's", "Comedy", "Crime", "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical", "Mystery", "Romance", "Sci-Fi", "Thriller", "War" and "Western". Year of production is from 1919 to 2000. We grouped years into 8 categories "1919-1930", "1931-1940", "1941-1950", "1951-1960", "1961-1970", "1971-1980", "1981-1990" and "1991-2000".

We convert the ordinal relation in to binary one. We use three different scaling. The first is that user rates a movie. The second is that a user does not like a movie (he rates movie with 1-2 stars). The last one is that user likes a movie (rates 4-5). This does not mean, that users do like (respective do not like) some genre, it means, that movies from this genre are or are not worth to see. We took the middle size version of the MovieLens dataset and we made a restriction to 3000 users and movies that were rated by that users. We take users, who rate movies the most, and we obtain dimension of the first data table 3000×30 and dimension of the second data table is 3671×26. Let us just note that for obtaining object-attribute relation we need to transpose Movies data table.

Relation "user rates a movie" make sense, because user rates a movie if he has seen it. We can understand this relation as user has seen movie. We get 29 multi-relational factors, that cover almost 100% of data (99.97%). Values of coverage, i.e. how large part of input data is covered can be seen in figure 2.

[1] http://grouplens.org/datasets/movielens/

Graphs in figure 3 show coverage of Users data table and Movies data table separately.

We can also see that for explaining more than 90 percent of data are sufficient 17 factors. This is significant reduction of input data.

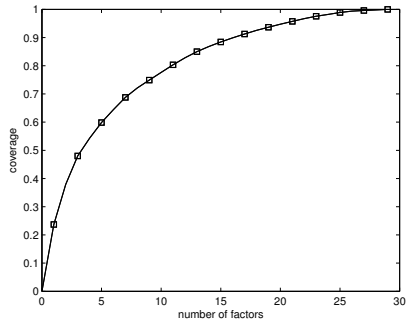
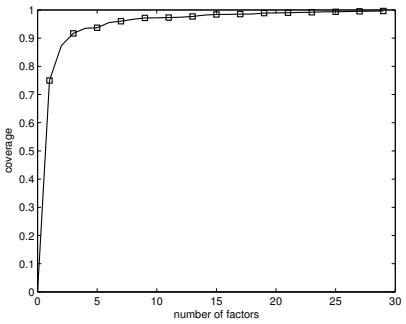
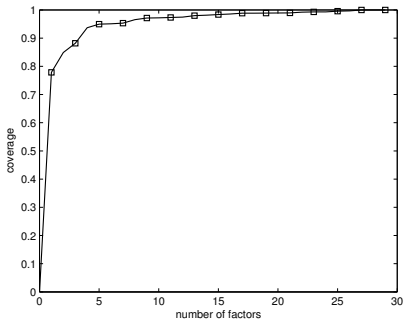


Fig. 2: Cumulative coverage of input data



(a) Coverage of Users data table



(b) Coverage of Movies data table

Fig. 3: Coverage of input data tables

The most important factors are:

- Males rate new movies (movies from 1991 to 2000).
- Young adult users (ages 25-34) rate drama movies.

– Females rate comedy movies.
– Youth users (18-24) rate action movies.

Another interesting factors are:

– Old users (from category 56+) rate movies from their childhood (movies from 1941 to 1950).
– Users in age range 50-55 rate children's movies. Users in this age usually have grand children.
– K-12 students rate animation movies.

Due to lack of space, we skip details about factors in relation "user does not like a movie" and relation "user does like a movie". In the first relation we get 30 factors, that covers 99.99% of data. In the second one, we get 29 factors, covering 99.96% of data. Compute all multi-relational factors on this datasets take approximately 5 minutes.

Remark 4. In case of MovieLens we are able to reconstruct input data tables almost wholly for each three relations. Interesting question is what about relation, i.e. can we reconstruct relation between data tables? Answer is yes, we can. Multi-relational factor carry also information about the relation between data tables. So we can reconstruct it, but with some error. This error is a result of choosing the narrow approach.

Reconstruction error of relation is interesting information and can be minimize if we take this error into account in phase of computing coverage. In other words we want maximal coverage with minimal relation reconstruction error. This leads to more complicated algorithm because we need weights to compute a value of utility function. We implement also this variant of algorithm. Requirement of minimal reconstruction error and maximal coverage seems to be contradictory, but this claim need more detailed study. Also it is necessary to determine correct weight settings. We left this issue for the extended version of this paper.

6 Conclusion and Future Research

In this paper, we present new algorithm for multi-relational Boolean matrix factorization, that uses essential elements from binary matrices for constructing better multi-relational factors, with regard to relations between each data table. We test the algorithm on, in data mining well known, dataset MovieLens. We obtain from these experiments interesting and easy interpretable results, moreover, the number of obtained multi-relational factors needed for explaining almost whole data is reasonable small.

A future research shall include the following topics: generalization of the algorithm for MBMF for ordinal data, especially data over residuated lattices. Construction of algorithm which takes into account reconstruction error of the

relation between data tables. Test the potential of this method in recommendation systems. And last but not least create not crisp operator for connecting classic factors into multi-relational factors.

Acknowledgment
We acknowledge support by IGA of Palacky University, No. PrF 2014 034.

References

1. Belohlavek R., Trnecka M.: From-Below Approximations in Boolean Matrix Factorization: Geometry and New Algorithm. http://arxiv.org/abs/1306.4905, 2013.
2. Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Syst. Sci. 76(1), 3–20, 2010.
3. Džeroski S.: Multi-Relational Data Mining: An Introduction. ACM SIGKDD Explorations Newsletter, 1(5), 1–16, 2003.
4. Ganter B., Wille R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin, 1999.
5. Geerts F., Goethals B., Mielikäinen T.: Tiling databases, Proceedings of Discovery Science 2004, pp. 278–289, 2004.
6. Hacene M. R., Huchard M., Napoli A., Valtechev P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. 67(1), 81–108, 2013.
7. Krmelova M., Trnecka M.: Boolean Factor Analysis of Multi-Relational Data. In: M. Ojeda-Aciego, J. Outrata (Eds.): CLA 2013: Proceedings of the 10th International Conference on Concept Lattices and Their Applications, pp. 187–198, 2013.
8. Lucchese C., Orlando S., Perego R.: Mining top-K patterns from binary datasets in presence of noise. SIAM DM 2010, pp. 165–176, 2010.
9. Miettinen P.: On Finding Joint Subspace Boolean Matrix Factorizations. Proc. SIAM International Conference on Data Mining (SDM2012), pp. 954-965, 2012.
10. Miettinen P., Mielikäinen T., Gionis A., Das G., Mannila H.: The discrete basis problem, IEEE Trans. Knowledge and Data Eng. 20(10), 1348–1362, 2008.
11. Spyropoulou E., De Bie T.: Interesting Multi-relational Patterns. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, ICDM '11, pp. 675–684, 2011.
12. Xiang Y., Jin R., Fuhry D., Dragan F. F.: Summarizing transactional databases with overlapped hyperrectangles, Data Mining and Knowledge Discovery 23(2), 215–251, 2011.