MINIMUM DESCRIPTION LENGTH IN BOOLEAN MATRIX FACTORIZATION

Martin Trnečka

Rigorous thesis

Department of Computer Science
Faculty of Science
Palacký University Olomouc
2019

Abstract – During the past few years Boolean matrix factorization (BMF) has become an important direction in data analysis. The minimum description length principle (MDL) was successfully adapted in BMF for the model order selection. Nevertheless, an BMF algorithm performing good results from the standpoint of standard measures in BMF is missing. In this thesis, we propose a novel from-below Boolean matrix factorization algorithm based on formal concept analysis. The algorithm utilizes the MDL principle as a criterion for the factor selection. On various experiments we show that the proposed algorithm outperforms—from different standpoints—existing state-of-the-art BMF algorithms.

To my son Tomáš

Contents

Preface

This thesis focuses on exploiting minimum description length principle (MDL) in from-below decompositions of the Boolean matrices—matrices that consist of zeros and ones. The thesis is based on two papers listed below:

- Makhalova, T., Trnecka, M.: From-Below Boolean Matrix Factorization Algorithm Based on MDL. *Advances in Data Analysis and Classification* (accepted).
- Makhalova, T., Trnecka, M.: A Study of Boolean Matrix Factorization Under Supervised Settings. In Proceedings of the 15th Internationl Conference on Formal Concept Analysis, ICFCA 2019, Frankfurt, Germany, 2019 (accepted).

The full list of my publications can be found on my personal web page `www.martin-trnecka.cz`.

This thesis consists of six chapters. The first chapter includes a brief introduction and an overview of related works. Then, in next chapter, a notation used in the thesis, a short introduction to BMF and MDL, and a background of the thesis are presented. Chapter 3 describes a design of our algorithm. The algorithm is experimentally evaluated in Chapter 4. The next chapter provides a brief evaluation of quality of factors in supervised settings. Chapter 6 draws a conclusion.

Chapter 1

Introduction

Boolean matrix factorization (BMF), also known as Boolean matrix decomposition, is a powerful and widely used data mining tool. Like classical matrix factorization methods, e.g., non-negative matrix factorization (NNMF) or singular value decomposition (SVD), BMF provides a different description (see Chapter 2.2) of Boolean data via new, more fundamental variables, called factors.

In BMF a given input data matrix is approximated by a product of so-called object-factor and factor-attribute matrices. All matrices contain zeros and ones only. The quality of the factorization—i.e., the quality of factors themselves—is usually measured by standard measures in BMF, namely by the number of factors and by the coverage (how large is the portion of data is described by factors, see Chapter 2). Both can be easily implemented—in fact each successful BMF algorithm already utilized them—in an arbitrary BMF algorithm. Moreover, both are very important in the evaluation of the factorization quality [2]. On the other hand, other aspects of the quality of factors, e.g., the interpretability, that are often neglected in the factor evaluation, are also important parts of the matrix factorization.

By now, various approaches to assessment of the quality of factors were developed [2, 15]. One of the most fundamental—but surprisingly not often used—is based on the well-known minimum description length principle (MDL). In terms of MDL, the best factorization is the factorization with the minimal description. Due to the MDL principle, such factorization is useful and easily interpretable. Neverthless, it was many times shown (see, e.g., [2, 3]) that mixing MDL and BMF produces poor results with respect to the BMF standard error measures (the number of factors and the coverage). More details will be provided in Chapter 2.

Recent results [16] in the field of formal concept analysis (FCA)—which is related to the BMF (see Chapter 2.3)—involving the minimum description length (MDL) motivate us to revise the use of MDL in BMF.

We propose a new heuristic BMF algorithm for from-below matrix factorization that outperforms existing state-of-the-art algorithms and produces very good result w.r.t. the standard BMF measures. The algorithm utilizes formal concept analysis and the MDL principle. Additionally, we present an extensive experimental evaluation of factors delivered by the proposed algorithm and its comparison to some already existing algorithms.

1.1 Related Work

In the last decade, many BMF methods were developed [17, 4, 14, 22, 15, 3]. It was shown [21] that applying existing non Boolean methods (e.g., NNMF, SVD) on Boolean data is inappropriate, especially from the interpretation standpoint.

A good overview of BMF and related topics can be found in [17, 3, 2]. In general, BMF is addressed in various papers involving formal concept analysis [4, 11], role mining [7], binary databases [9] or bipartite graphs [1].

In many applications of BMF, instead of a general Boolean factorization—which can be computed for instance by well-known Asso algorithm [17]—only a certain class of factorization, so-called from-below matrix factorization [3], is considered (see Section 2).

In the recent years, the minimum description length principle [10] has been applied in BMF. It was used mostly to solve the model order selection problem [18, 19]—i.e., separation of global structure from noise—or as a factor selection criteria in BMF algorithms, e.g., in the state-of-the-art algorithm PaNDa$^+$ [15], an improvement and generalized version of PaNDa algorithm [14]. As a special case of an application of MDL in BMF Hyper [22] algorithm can be considered, its objective is to minimize the description of factors instead of the minimization of the description length (for more details see [15]). The MDL4BMF algorithm presented in [19] for model order selection problem can be used for computation of factorization with MDL. More precisely in MDL4BMF the parameters of the state-of-the-art algorithm Asso are tuned to obtain the minimal total description length. However, there is a big difference between this algorithm and our approach. Our algorithm optimizes MDL directly, i.e., the cost function is based on MDL, whereas MDL4BMF optimizes MDL indirectly via parameters, but the selection of factors is based on a coverage quality (an original criterion for factor selection in Asso algorithm).

Another related work is [16], where a set of formal concepts with MDL is considered for the classification task. Our algorithm can be used for similar tasks. Instead of [16] our algorithm does not require computing the whole set of formal concepts, that makes it applicable in practice. Moreover we used a different approach to MDL measuring.

Chapter 2

Background and Basic Definitions

2.1 Notation

Through the thesis we use a matrix terminology and in some convenient places a relational terminology. Matrices are denoted by upper-case bold letters ($\mathbf{I}$). $\mathbf{I}_{ij}$ denotes the entry corresponding to the row i and the column j of $\mathbf{I}$. The set of all $m \times n$ Boolean (binary) matrices is denoted by $\{0,1\}^{m\times n}$. The number of 1s in Boolean matrix $\mathbf{I}$ is denoted by $\|\mathbf{I}\|$, i.e $\|\mathbf{I}\| = \sum_{i,j} \mathbf{I}_{ij}$.

We interpret input data $\mathbf{I} \in \{0,1\}^{m\times n}$ primarily as an object-attribute incidence matrix, i.e., a relation between the set of objects and the set of attributes. That is, the entry $\mathbf{I}_{ij}$ is either 1 or 0, indicating that the object i does or does not have the attribute j.

If $\mathbf{A} \in \{0,1\}^{m\times n}$ and $\mathbf{B} \in \{0,1\}^{m\times n}$, we have the following element-wise matrix operations. The *Boolean sum* $\mathbf{A} \oplus \mathbf{B}$ which is the normal matrix sum where $1+1=1$. The *Boolean subtraction* $\mathbf{A} \ominus \mathbf{B}$ which is the normal matrix subtraction, where $0-1=0$.

2.2 Boolean Matrix Factorization

A general aim in BMF is for a given Boolean matrix $\mathbf{I} \in \{0,1\}^{m\times n}$ to find matrices $\mathbf{A} \in \{0,1\}^{m\times k}$ and $\mathbf{B} \in \{0,1\}^{k\times n}$ for which

$$\mathbf{I} \approx \mathbf{A} \circ \mathbf{B} \tag{2.1}$$

where $\circ$ is Boolean matrix multiplication, i.e., $(\mathbf{A}\circ\mathbf{B})_{ij} = \max_{l=1}^{k} \min(\mathbf{A}_{il}, \mathbf{B}_{lj})$, and $\approx$ represents approximate equality assessed by $|| \cdot ||$. The corresponding metric E is defined for matrices $\mathbf{I} \in \{0,1\}^{m\times n}$, $\mathbf{A} \in \{0,1\}^{m\times k}$ and $\mathbf{B} \in \{0,1\}^{k\times n}$ by

$$E(\mathbf{I}, \mathbf{A} \circ \mathbf{B}) = ||\mathbf{I} \ominus (\mathbf{A} \circ \mathbf{B})||. \tag{2.2}$$

A decomposition of $\mathbf{I}$ into $\mathbf{A} \circ \mathbf{B}$ may be interpreted as a discovery of k factors that exactly or approximately explain the data: interpreting $\mathbf{I}$, $\mathbf{A}$, and $\mathbf{B}$ as the object–attribute, object–factor, and factor–attribute matrices, the model (2.1) has the following interpretation: the object i has the attribute j, i.e., $\mathbf{I}_{ij} = 1$, if and only if there exists factor l such that l applies to i and j is one of the particular manifestations of l.

Note also an important geometric view of BMF: a decomposition $\mathbf{I} \approx \mathbf{A} \circ \mathbf{B}$ with k factors represents a coverage of the 1s in $\mathbf{I}$ by k rectangular areas in $\mathbf{I}$ full of 1s, the lth rectangle is the Boolean sum of the lth column in $\mathbf{A}$ and the lth row in $\mathbf{B}$. For more details see, e.g., [12].

If the rectangular areas cover only non zero elements in the matrix $\mathbf{I}$, the $\mathbf{A} \circ \mathbf{B}$ is called the *from-below matrix decomposition* [3]. An example of the from-below BMF follows.

Example 1. *Let us consider Boolean matrix with rows* $1, \ldots, 8$ *and columns* $a, \ldots, h$ *depicted in Figure 2.1. The Boolean matrix is given in the shape of table, where nonzero entries are marked by crosses. Two different factorizations of the data are shown in Figure 2.2.*

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	×	×	×				×	×
2	×	×	×				×	×
3	×	×	×	×				
4	×	×	×	×		×		
5		×	×	×				×
6		×	×	×	×	×		
7		×	×	×	×	×	×	×
8					×	×	×	×

Figure 2.1: Example data.

2.3 BMF with Help of Formal Concept Analysis

Formal concept analysis (FCA) [8] provides a basic framework for dealing with factors. The main notion of FCA is *formal context*, which is usually represented as a Boolean matrix, is defined as a triple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{I} \rangle$, where $\mathcal{X}$

$$
\begin{array}{c|c|c|c|c|}
 & f_1 & f_2 & f_3 & f_4 \\ \hline
1 & \times & & & \\ \hline
2 & \times & & & \\ \hline
3 & & \times & & \\ \hline
4 & & \times & & \\ \hline
5 & & \times & & \\ \hline
6 & & \times & \times & \\ \hline
7 & & \times & \times & \times \\ \hline
8 & & & & \times \\ \hline
\end{array}
\circ
\begin{array}{c|c|c|c|c|c|c|c|c|}
 & a & b & c & d & e & f & g & h \\ \hline
f_1 & \times & \times & \times & & & & \times & \times \\ \hline
f_2 & & \times & \times & \times & & & & \\ \hline
f_3 & & \times & \times & \times & \times & \times & & \\ \hline
f_4 & & & & & \times & \times & \times & \times \\ \hline
\end{array}
=
\begin{array}{c|c|c|c|c|c|c|c|c|}
 & a & b & c & d & e & f & g & h \\ \hline
1 & \times & \times & \times & & & & \times & \times \\ \hline
2 & \times & \times & \times & & & & \times & \times \\ \hline
3 & & \times & \times & \times & & & & \\ \hline
4 & & \times & \times & \times & & & & \\ \hline
5 & & \times & \times & \times & & & & \\ \hline
6 & & \times & \times & \times & \times & \times & & \\ \hline
7 & & \times & \times & \times & \times & \times & \times & \times \\ \hline
8 & & & & & \times & \times & \times & \times \\ \hline
\end{array}
$$

$$
\begin{array}{c|c|c|}
 & f_1 & f_2 \\ \hline
1 & \times & \\ \hline
2 & \times & \\ \hline
3 & & \times \\ \hline
4 & & \times \\ \hline
5 & & \times \\ \hline
6 & & \times \\ \hline
7 & \times & \times \\ \hline
8 & \times & \\ \hline
\end{array}
\circ
\begin{array}{c|c|c|c|c|c|c|c|c|}
 & a & b & c & d & e & f & g & h \\ \hline
f_1 & & & & & & & \times & \times \\ \hline
f_2 & & \times & \times & \times & & & & \\ \hline
\end{array}
=
\begin{array}{c|c|c|c|c|c|c|c|c|}
 & a & b & c & d & e & f & g & h \\ \hline
1 & & & & & & & \times & \times \\ \hline
2 & & & & & & & \times & \times \\ \hline
3 & & \times & \times & \times & & & & \\ \hline
4 & & \times & \times & \times & & & & \\ \hline
5 & & \times & \times & \times & & & & \\ \hline
6 & & \times & \times & \times & & & & \\ \hline
7 & & \times & \times & \times & & & \times & \times \\ \hline
8 & & & & & & & \times & \times \\ \hline
\end{array}
$$

Figure 2.2: Two examples of data factorization.

is a nonempty set of objects , $\mathcal{Y}$ is a nonempty set of attributes and $\mathcal{I}$ is a binary relation between $\mathcal{X}$ and $\mathcal{Y}$. Hence the formal context $\langle \mathcal{X}, \mathcal{Y}, \mathcal{I} \rangle$ with m objects and n attributes is a Booolean matrix $\mathbf{I} \in \{0,1\}^{m \times n}$.

To every Boolean matrix $\mathbf{I} \in \{0,1\}^{m \times n}$, one might associate the pair $\langle {}^{\uparrow}, {}^{\downarrow} \rangle$ of operators (in FCA well known as the arrow operators) assigning to sets $C \subseteq \mathcal{X} = \{1, \ldots, m\}$ and $D \subseteq \mathcal{Y} = \{1, \ldots, n\}$ the sets $C^{\uparrow} \subseteq \mathcal{Y}$ and $D^{\downarrow} \subseteq \mathcal{X}$ defined by

$$
\begin{aligned}
C^{\uparrow} &= \{ j \in \mathcal{Y} \,|\, \forall i \in C : \mathbf{I}_{ij} = 1 \}, \\
D^{\downarrow} &= \{ i \in \mathcal{X} \,|\, \forall j \in D : \mathbf{I}_{ij} = 1 \},
\end{aligned}
$$

where $C^{\uparrow}$ is the set of all attributes (columns) shared by all objects (rows) in C and $D^{\downarrow}$ is the set of all objects sharing all attributes in D.

The pair $\langle C, D \rangle$ for which $C^{\uparrow} = D$ and $D^{\downarrow} = C$ is called the *formal concept*. C and D are called the *extent* and the *intent* of formal concept $\langle C, D \rangle$, respectively. The concepts are partially ordered as follows: $\langle A, B \rangle \leq \langle C, D \rangle$ iff $A \subseteq C$ (or $D \subseteq B$), a pair $\langle A, B \rangle$ is a subconcept of $\langle C, D \rangle$, while $\langle C, D \rangle$ is a superconcept of $\langle A, B \rangle$. The set of all formal concepts we denote by

$$\mathcal{B}(\mathbf{I}) = \{ \langle C, D \rangle \mid C \subseteq \mathcal{X}, D \subseteq \mathcal{Y}, C^{\uparrow} = D, D^{\downarrow} = C \}.$$

The whole set of partially ordered formal concepts is called the *concept lattice* of $\mathbf{I}$.

Given a set $\mathcal{F} = \{ \langle C_1, D_1 \rangle, \ldots, \langle C_k, D_k \rangle \} \subseteq \mathcal{B}(\mathbf{I})$ (with a fixed indexing of the formal concepts $\langle C_l, D_l \rangle$), induces the $m \times k$ and $k \times n$ Boolean matrices $\mathbf{A}_\mathcal{F}$ and $\mathbf{B}_\mathcal{F}$ by

$$(\mathbf{A}_\mathcal{F})_{il} = \begin{cases} 1, \text{if } i \in C_l, \\ 0, \text{if } i \notin C_l, \end{cases} \tag{2.3}$$

and

$$(\mathbf{B}_\mathcal{F})_{lj} = \begin{cases} 1, \text{if } j \in D_l, \\ 0, \text{if } j \notin D_l, \end{cases} \tag{2.4}$$

for $l = 1, \ldots, k$. That is, the lth column and lth row of $\mathbf{A}_\mathcal{F}$ and $\mathbf{B}_\mathcal{F}$ are the characteristic vectors of C_l and D_l, respectively. The set $\mathcal{F}$ is also called a set of *factor concepts.* Clearly, $\mathbf{A}_\mathcal{F} \circ \mathbf{B}_\mathcal{F}$ is the from-below matrix decomposition.

Example 2. *Let us consider two factorizations depicted in Figure 2.2. The first one corresponds to the set*

$$\begin{aligned}\mathcal{F} = \{&\langle\{1,2\},\{a,b,c,g,h\}\rangle, \langle\{3,4,5,6,7\},\{b,c,d\}\rangle,\\ &\langle\{6,7\},\{b,c,d,e,f\}\rangle, \langle\{7,8\},\{e,f,g,h\}\rangle\}.\end{aligned}$$

The second one corresponds to the set

$$\mathcal{F} = \{\langle\{1,2,7,8\},\{g,h\}\rangle, \langle\{3,4,5,6,7\},\{b,c,d\}\rangle\}.$$

For more details how the formal concept analysis is utilized in BMF and the advantages of such approach see the pioneer work [4].

2.4 A Brief Introduction to MDL

The minimum description length (MDL) principle, which is a computable version of Kolmogorov complexity [10], is a formalization of the law of parsimony well known as Occam's razor. In terms of MDL, it is formulated as follows: the best model is the model that ensures the best compression of the given data.

More formally, for a given set of models $\mathcal{M}$ and data (in our case represented via Boolean matrix $\mathbf{I}$) the best model $M \in \mathcal{M}$ is the one that minimizes the following cost function:

$$L(M) + L(\mathbf{I} \mid M), \tag{2.5}$$

where $L(M)$ is the encoding length of M in bits and $L(\mathbf{I} \mid M)$ is the encoding length in bits of the data $\mathbf{I}$ encoded with M.

In general, we are only interested in the length of the encoding, and not in the coding itself, i.e., we do not have to materialize the codes themselves.

Note that MDL requires the compression to be lossless in order to allow for a fair comparison between different models.

2.5 The Quality of Factorization

The quality of the obtained factorization (2.1) is usually evaluated via some variants of metric (2.2). From the BMF perspective there are two basic viewpoints, emphasizing the role of the first k factors and the need to account for a prescribed portion of data, respectively. They are known as the discrete basis problem (DBP) and the approximate factorization problem (AFP), see [17] and [4, 3]. Both of them emphasize the coverage of data, i.e., the geometric view of BMF.

In many applications of BMF, the interpretation of factors plays a crucial role. It is reasonable instead of the coverage of the obtained factorization empathize a different quality measures that access the interpretability of factors, e.g., the MDL.

On the other hand, the geometric view of BMF is very important and an interpretable factorization should reflect it.

In the next chapter, we propose a novel BMF algorithm which is based on well-known GreConD algorithm [4]. The algorithm computes from-below factorization via minimization of the cost function (2.5). The results of experiments show that it preserves a lot of information from the original data w.r.t. the error measure (2.2).

Chapter 3

Design of Algorithm

3.1 MDL in From-below Matrix Factorization

For matrices $\mathbf{A}_{\mathcal{F}} \in \{0,1\}^{m\times k}$, $\mathbf{B}_{\mathcal{F}} \in \{0,1\}^{k\times m}$, and $\mathbf{I} \in \{0,1\}^{m\times n}$ where $\mathbf{I} \approx (\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}})$ we define an error matrix $\mathbf{E}$ as follows:

$$\mathbf{I} = (\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}) \oplus \mathbf{E}.$$

One may observe that matrix $\mathbf{E}$ can be easily computed via metric (2.2), i.e., $\mathbf{E} = E(\mathbf{I}, \mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}})$. Hence, to provide a lossless compression of $\mathbf{I}$ it is sufficient to encode the matrices $\mathbf{A}_{\mathcal{F}}, \mathbf{B}_{\mathcal{F}}$ and $\mathbf{E}$, i.e., the MDL cost function (2.5) has the following form

$$L(\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}) + L(\mathbf{E}). \tag{3.1}$$

According to the MDL principle, the best factorization of $\mathbf{I}$ minimizes function (3.1). In the following we explain how to compute the length of the encoding of matrices $\mathbf{A}_{\mathcal{F}}, \mathbf{B}_{\mathcal{F}}$ and $\mathbf{E}$ in bits. We use a similar approach as in [18] and we modify it for the from-below matrix factorization.

More precisely, to use optimal prefix codes we need to encode the dimensions of the matrices and the matrices themselves, i.e.,

$$L(\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}) + L(\mathbf{E}) \quad = \quad L(m) + L(n) + L(k) + L(\mathbf{A}_{\mathcal{F}}) + L(\mathbf{B}_{\mathcal{F}}) + L(\mathbf{E}).$$

For the sake of simplicity we may encode the dimensions m, n, k with block-encoding, which gives us $L(m) = L(n) = L(k) = \log(\max(m, n, k))$.

To not introduce some influencing between factors, these are encoded per factor, i.e., we encode $\mathbf{A}_{\mathcal{F}}$ per column and $\mathbf{B}_{\mathcal{F}}$ per row.

In order to use optimal prefix code, we need to first encode the probability of encountering 1 in a particular column or row respectively, i.e., we need $\log m$ bits for each extent in set $\mathcal{F}$ and $\log n$ bits for each $\mathcal{F}$ intent in set $\mathcal{F}$, respectively.

For simplicity, extent C and intent D of factor concept $\langle C, D\rangle$ can be seen as characteristic vectors, i.e., $C \in \{0,1\}^{m\times 1}$ and $D \in \{0,1\}^{1\times n}$. We need to encode all ones and zeros. The length of optimal code is determined by Shannon entropy. This gives us the number of bits required for the encoding of matrices $\mathbf{A}_{\mathcal{F}}$ and $\mathbf{B}_{\mathcal{F}}$:

$$L(\mathbf{A}_{\mathcal{F}}) = \sum_{\langle C,D\rangle\in\mathcal{F}} \log m - (||C|| \cdot \log \frac{||C||}{m} + (m - ||C||) \cdot \log \frac{m - ||C||}{m}),$$

$$L(\mathbf{B}_{\mathcal{F}}) = \sum_{\langle C,D\rangle\in\mathcal{F}} \log n - (||D|| \cdot \log \frac{||D||}{n} + (n - ||D||) \cdot \log \frac{n - ||D||}{n}).$$

In a similar way we can compute the number of bits required for the encoding of matrix $\mathbf{E}$:

$$L(\mathbf{E}) = \log mn - (||\mathbf{E}|| \cdot \log \frac{||\mathbf{E}||}{mn} + (mn - ||\mathbf{E}||) \cdot \log \frac{mn - ||\mathbf{E}||}{mn}).$$

Note, we can encode matrix $\mathbf{E}$ element-by-element without any influence, because these elements are clearly independent.

3.2 Algorithm

In this section we propose a BMF algorithm, called MDLGreConD[1], that uses the above described MDL cost function. The algorithm is a modified version—it utilizes a similar search strategy—of the GreConD[2] algorithm [4], which is one of the most successful from-below matrix decomposition algorithms (see e.g., [2]).

Pseudocode of MDLGreConD is depicted in Algorithm 1. The algorithm works as follows.

[1]MDLGreConD is an abbreviation of Minimum Description Length Greedy Concept on Demand.

[2]GreConD is an abbreviation of Greedy Concept on Demand.

```
   Input: Boolean matrix $\mathbf{I}$.
   Output: Set $\mathcal{F}$ of factor concepts.
 1 $\mathcal{F} \leftarrow \emptyset$
 2 $total\_cost \leftarrow \infty$
 3 $\mathbf{E} \leftarrow \mathbf{I} \ominus (\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}})$
 4 while $L(\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}) + L(\mathbf{E})$ is decreasing do
 5     $\langle C, D\rangle \leftarrow \langle\emptyset, \emptyset\rangle$
 6     while $\langle C, D\rangle$ is changing do
 7         $total\_cost' \leftarrow total\_cost$
 8         foreach $j \notin D$ do
 9             $D' \leftarrow (D \cup \{j\})^{\downarrow\uparrow}$
10             $C' \leftarrow D'^{\downarrow}$
11             if $\langle C', D'\rangle \in \mathcal{F}$ then
12                 continue with next $j$
13             end
14             $\mathcal{F}' \leftarrow \mathcal{F} \cup \langle C', D'\rangle$
15             $cost \leftarrow L(\mathbf{A}_{\mathcal{F}'} \circ \mathbf{B}_{\mathcal{F}'}) + L(I \ominus (\mathbf{A}_{\mathcal{F}'} \circ \mathbf{B}_{\mathcal{F}'}))$
16             if $cost < total\_cost'$ then
17                 $total\_cost' \leftarrow cost$
18                 $C'' \leftarrow C'$
19                 $D'' \leftarrow D'$
20             end
21         end
22         $total\_cost \leftarrow total\_cost'$
23         $C \leftarrow C''$
24         $D \leftarrow D''$
25     end
26     $\mathcal{F} \leftarrow \mathcal{F} \cup \langle C, D\rangle$
27 end
28 return $\mathcal{F}$
```

Algorithm 1: MDLGreConD algorithm

The algorithm computes a candidate $\langle C, D\rangle$ to a factor concept that minimizes the cost function (3.1) stored in variable *total_cost*. This is done via searching of a promising column j that is not included in D (lines 8–21). Note that the adding of j to D is realized via $^{\uparrow}$ and $^{\downarrow}$ operators mentioned in Chapter 2.3. Only the best column j is considered (lines 16–20). If a new column is added to $\langle C, D\rangle$, i.e., the $\langle C, D\rangle$ is changed, the modified $\langle C, D\rangle$ is used as a new candidate and another promising column is searched for. If there is no column that reduce the cost function (line 6), already computed candidate is added to the output set $\mathcal{F}$ of factor concepts. The algorithm ends if there is no candidate that allows for reduction of the cost function.

3.3 Computational Complexity

The Boolean matrix factorization problem is NP-hard [20] as well as the computation of factorization that minimizes the cost function (3.1). The proposed algorithm is heuristic. One may easily derive an exact algorithm with an exponential time complexity. Such algorithm is inapplicable in practice.

We now derive an upper bound of the worst case time complexity of MDLGreConD. One may easily observe, that the computing $^{\uparrow}$ or $^{\downarrow}$ on matrix $\mathbf{I} \in \{0,1\}^{m \times n}$ is done in time $\mathcal{O}(m)$ and $\mathcal{O}(n)$, respectively. The core of the algorithm lies within two loops on line 6 and 8. The outer one proceeds at most n times since no more than n attributes may eventually be added when the $\langle C, D\rangle$ is extended. Within the j-th execution of the outer cycle, the inner loop is executed at most $n+1-j$ times since this is the number of remaining candidate attributes for extending the $\langle C, D\rangle$. Hence the innermost loop is executed $\sum_{j=1}^{n}(n+1-j) = O(n^2)$ times which along with the at most $O(n \cdot m)$ steps within each execution of the innermost cycle yields that the code on lines requires time $O(m \cdot n^3)$. Note that the computation of *cost* can be done in $\mathcal{O}(m \cdot n)$ because $\mathbf{A}_{\mathcal{F}'} \circ \mathbf{B}_{\mathcal{F}'}$ can be incrementally computed from $\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}$. The loop on line 4 is repeated at most k times, where k is the number of factors. In practice, it is reasonable to assume that $k < n$. Altogether, the overall upper bound of the worst-case time complexity of MDLGreConD algorithm, which is standard for various BMF algorithms (see e.g., [3]) is $\mathcal{O}(n \cdot m \cdot n^3)$. It should be noticed that the provided estimates are not tight and a better time complexity can be obtain by making more precise estimate of some steps of the algorithm.

It should be emphasized that k is usually much smaller than n and m is smaller than n. Therefore, we observed that the practical time complexity is significantly better than the aforementioned time complexity.

The presented algorithm is only slightly slower than GreConD algorithm, which is, probably, the fastest BMF algorithm (see e.g., [3]). Both of them are able to factorize on ordinary PC, each of the data presented in Section 4 under 10 second.

Chapter 4

Experimental Evaluation

In this chapter, the results of an experimental comparison of BMF algorithms with MDLGreConD are presented.

4.1 Datasets

We use 6 different real-world datasets, namely `Breast`[1], `Ecoli`, `Iris` and `Mushroom` from UCI repository [13], and `Domino` and `Emea` from [7]. The characteristics of the datasets are shown in Table 4.1. All of them are well known and widely used as benchmark datasets in BMF.

Table 4.1: Datasets and their characteristics.

dataset	size	dens. $\mathbf{I}$	$\|\|\mathcal{B}(\mathbf{I})\|\|$
`Breast`	699×20	0.499	642
`Domino`	79×231	0.400	73
`Ecoli`	336×34	0.235	813
`Emea`	3046×35	0.068	780
`Iris`	150×19	0.263	164
`Mushroom`	8124×90	0.252	186332

4.2 Algorithms

GreConD [4] algorithm is based on the "on demand" greedy search for formal concepts of $\mathbf{I}$. It is designed to compute an exact from-below factorization. Instead of going through all formal concepts, which are the candidates for factor concepts, it constructs the factor concepts by adding sequentially

[1]Breast Cancer Wisconsin (Original)

"promising columns" to candidate $\langle C, D\rangle$ to factor concept. More formally, a new column j that minimizes the error

$$E(\mathbf{I}, \mathbf{A}_{\mathcal{F}\cup\langle (D\cup j)^{\downarrow},(D\cup j)^{\downarrow\uparrow}\rangle} \circ \mathbf{B}_{\mathcal{F}\cup\langle (D\cup j)^{\downarrow},(D\cup j)^{\downarrow\uparrow}\rangle})$$

is added to $\langle C, D\rangle$. This is repeated until no such columns exist. If there is no such column, the $\langle C, D\rangle$ is added to the set $\mathcal{F}$. The algorithm ends if $E(\mathbf{I}, \mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}})$ is smaller than the prescribed parameter ϵ or the prescribed number of factors is reached. For more details see [3]. Note, that usually $\epsilon = 0$, i.e., the whole matrix $\mathbf{I}$ is covered by factors. Such setting was adopted in our experiments.

PaNDa$^+$ [15] is an algorithmic framework based on PaNDa [14] algorithm. The algorithm aims to extract a set $\mathcal{F}$ of pairs $\langle C, D\rangle$ that minimizes the cost function:

$$\sum_{\langle C,D\rangle\in\mathcal{F}} (|C| + |D|) + E(\mathbf{I}, \mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}).$$

Every $\langle C, D\rangle$ in $\mathcal{F}$ is computed in two stages. On the first stage the core of $\langle C, D\rangle$ is computed, on the second stage the core is extended. A core is a rectangle, not necessarily a formal concept, contained in $\mathbf{I}$ and it is computed by adding columns from a sorted list. Extension to $\langle C, D\rangle$ is performed by adding columns and rows to a core while such an addition allows for reducing the cost. Note, that PaNDa$^+$ does not produce the from-below factorization. The computation of PaNDa$^+$ is driven by several parameters (see [15]). All of them are tuned for each dataset. The best obtained results are reported.

Hyper [22] algorithm aims to extract a set $\mathcal{F}$ of pairs $\langle C, D\rangle$ that minimize the cost function which is defined as follows:

$$\sum_{\langle C,D\rangle\in\mathcal{F}} (|C| + |D|)/E(\mathbf{I}, \mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}}).$$

As candidates to factors the set of all formal concepts $\mathcal{B}(\mathbf{I})$ together with all single attribute rectangles in data are considered. Each candidate is divided into a set of single row rectangles that are sorted according to the number of uncovered elements in $\mathbf{I}$. Then the algorithm tries to add the single row rectangles back to the candidate, until the above mentioned cost function decreases. After this, the algorithm in each iteration selects the concept $\langle C, D\rangle$ from the modified set of candidates that minimizes the cost function. Hyper algorithm produces the from-below factorization. The size of $\mathcal{B}(\mathbf{I})$ can be exponentially large. In such case Hyper has the exponential time complexity. To reduce computational cost [22] proposes to use only frequent formal concepts (the frequency is an additional parameter of the algorithm). Our experiments show that the frequency highly affects the performance of the algorithm. In our experiments we use the whole set of formal concepts $\mathcal{B}(\mathbf{I})$, (for the set sizes see the last column of Table 4.1).

MDL4BMF [19] algorithm utilizes well-known BMF algorithm Asso [17]. The Asso first computes Boolean matrix $\mathbf{X}$ where $\mathbf{X}_{ij} = 1$ if the confidence of association rule $\{i\} \Rightarrow \{j\}$ exceeds a threshold parameter τ. The rows of $\mathbf{X}$, called basis vectors, are used as candidates for the rows of the factor–attribute matrix $\mathbf{B}$. The corresponding columns of matrix $\mathbf{A}$ from rows of $\mathbf{B}$ are computed. More precisely, the algorithm aims to extract a set $\mathcal{F}$ of pairs $\langle C, D\rangle$ (general rectangle which may contains zeros and ones) that maximizes the following cost function:

$$\sum_{\langle C,D\rangle\in\mathcal{F}} ||\mathbf{I} \wedge \mathbf{Y}|| - ||\bar{\mathbf{I}} \wedge \mathbf{Y}||,$$

where $\mathbf{Y}_{ij} = 1$ iff $i \in C$ and $j \in D$, otherwise $\mathbf{Y}_{ij} = 0$, $\wedge$ is an element-wise matrix product and $\bar{\mathbf{I}}$ is matrix $\mathbf{I}$ where all elements are flipped, i.e. ones to zeros and zeros to ones, respectively. MDL4BMF computes factorization via Asso algorithm for different values of τ. Then (k, τ), where k is the number of factors, minimizes the total description length is selected and the final decomposition for this parameters is delivered. The total description length is measured in a similar way as was described in Section 3. Note that Asso and thus MDL4BMF are not able to compute from-bellow matrix factorization, this fact is also reflected in a measuring of the description length. Our approach presented in Section 3 is tailored for such kind of decompositions. Moreover, computing various factorization for different τ is time consuming.

4.3 Evaluation

In our experiments we compare MDLGreCond algorithm with GreConD, Hyper, PaNDa$^+$ and MDL4BMF. We study factors themselves and how well they cover the analyzed datasets.

4.3.1 The Number of Factors

One of the main characteristic of BMF algorithms is the number of factors they produce. We measure not only the total number of factors, but also how many non-singleton factors are computed. Under singleton factors we mean the single-attribute ones. The results are shown in Table 4.2.

As it can be seen from the table, PaNDa$^+$ tends to produce only few factors (w.r.t. the number of attributes, see Table 4.1).

Table 4.2: The number of factors.

		no. of factors	
dataset	algorithm	non-singleton	singleton
`Breast`	GreConD	15	4
	Panda$^+$	4	0
	Hyper	36	0
	MDL4BMF	12	0
	MDLGreConD	6	1
`Domino`	GreConD	13	8
	Panda$^+$	3	0
	Hyper	10	132
	MDL4BMF	7	4
	MDLGreConD	7	3
`Ecoli`	GreConD	38	3
	Panda$^+$	6	0
	Hyper	35	30
	MDL4BMF	7	0
	MDLGreConD	8	1
`Emea`	GreConD	9	33
	Panda$^+$	3	0
	Hyper	3	35
	MDL4BMF	3	0
	MDLGreConD	7	2
`Iris`	GreConD	8	12
	Panda$^+$	8	0
	Hyper	13	15
	MDL4BMF	3	7
	MDLGreConD	7	0
`Mushroom`	GreConD	98	3
	Panda$^+$	8	0
	Hyper	89	2
	MDL4BMF	64	0
	MDLGreConD	50	0

Hyper returns the number of factors which is close to the number of attributes. Moreover, more than a half of them are singleton. This is true on all datasets with an exception of `Breast` and `Mushroom` data.

On average (see Figure 4.1), the number of non-singleton factors of GreConD is lower than the number of non-singleton factors of Hyper algorithm. MDLGreConD generates a small set of factors, most of them are non-singleton. The third best result is provided by MDL4BMF which is only slightly worse than MDLGreConD. PaNDa$^+$ tends to produce the smallest number of factors. All of them are non-singleton.

However, considering only the number of factors might be insufficient, since usually one wants to find not just the smallest set of factors, but the set that captures (covers) a large portion of data. Further we will show how the algorithms capture the analyzed data.

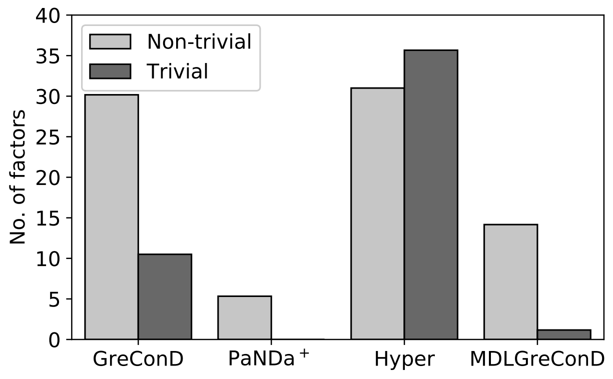

Figure 4.1: The average number of factors.

4.3.2 Data Coverage

Another important characteristic of factors is how much information from the analyzed dataset they retain. We measure it by coverage rate. We differentiate *data coverage* and *object coverage*. Data coverage measures the rate of "crosses" covered by factors in the dataset—this is a standard measure in BMF, see e.g., [2]. However, data coverage might be an inappropriate measure in cases where a dataset contains a lot of redundant attributes. Taking into consideration these cases, we measure the object coverage rate, i.e. how many objects are covered at least by one factor. The following example explains how the coverage measures are computed.

Example 3. *The factor set of the first factorization (Figure 2.2) covers almost all crosses in data, while the second set covers around a half of crosses. The coverings for both of them are given below. The crosses covered by one factor are light gray, the crosses covered by more factors are colored with darker gray.*

Note that both factor sets cover all objects, i.e. every row in the dataset has at least one colored cross, thus the object coverage rate is equal to 1 for both factorizations.

For the first factorization, the cross coverage rate is $^{35}/_{39} = 0.897$*. In the case of the second factorization, the cross coverage rate is* $^{23}/_{39} = 0.589$*, the bigger value is better.*

Average values of data coverage and object coverage rates over all datasets as well as the minimal, maximal values and quantiles (25% and 75%) are shown in Figures 4.3 and 4.4, respectively. The average data coverage rate of non-singleton factors of MDLGreConD is slightly lower than the analogous measure for GreConD, MDL4BMF and Hyper. It is important to note

	a	b	c	d	e	f	g	h
1	×	×	×				×	×
2	×	×	×				×	×
3	×	×	×	×				
4	×	×	×	×		×		
5		×	×	×				×
6		×	×	×	×	×		
7		×	×	×	×	×	×	×
8					×	×	×	×

	a	b	c	d	e	f	g	h
1	×	×					×	×
2	×	×	×				×	×
3	×	×	×	×				
4	×	×	×	×		×		
5		×	×	×				×
6		×	×	×	×	×		
7		×	×	×	×	×	×	×
8					×	×	×	×

Figure 4.2: The covering with factors from the running examples.

that MDLGreConD provides more stable results, in other words, the data coverage rate does not depend a lot on datasets, while for Hyper algorithm, the data coverage rate changes from 0.1 to 1.0. As it can be seen from the figure, the most whole range of values of MDLGreConD (the highest and lowest values in whiskers) are much smaller even than the range of 50% of values provided by other algorithms (see the heights of boxes that corresponds to values of 25%- and 75%-quartiles). PaNDa^{+} covers slightly more than a half of data by a small set of factors. Note that the results of MDL4BMF are affected by so-called over-cover error [3], i.e. zeros in input data that are covered by some factors. In Figures 4.3 and 4.4 we give the average rate of 1s in the original dataset covered by factors. GreConD, Hyper and MDLGreConD do not commit over-cover error at all as they are from-bellow factorization algorithms. On average the over-cover error was 0.05 for MDL4BMF and close to zero for PaNDa^{+}. Taking this into account coverage rates of MDL4BMF and MDLGreConD are comparable.

Moreover, if we take into account results regarding the number of factors from Section 4.3.1, MDLGreConD outperforms all remaining algorithms. Namely, it provides a large coverage by a smaller number of factors.

Regarding the object coverage rate, all the algorithms have similar performance, however a large number of non-trivial factors in Hyper ensures its high coverage rate for all chosen datasets.

4.3.3 Redundancy of Factors

An important characteristic of a factor set is redundancy. The factor set is redundant if it contains repetitive information, i.e., if it contains some overlaps between factors. We measure redundancy by *overlapping rate* (see Example 4), i.e., how many times the covered crosses are covered by several

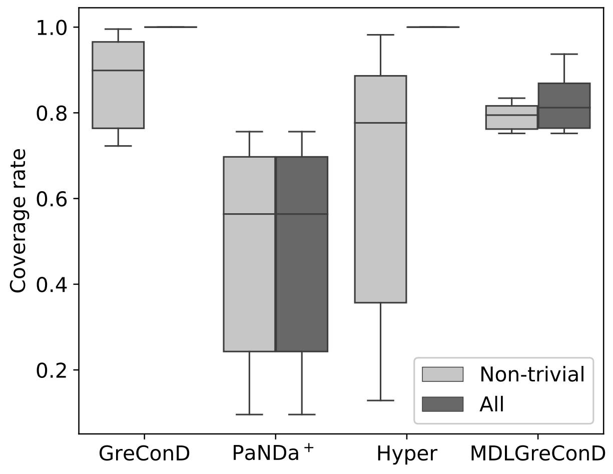

Figure 4.3: The average data coverage rate.

factors.

Example 4. *For the factor sets from Figure 2.2 the average overlapping rate is computed as follows. We count the total area of factors* $area(\langle C, D\rangle) = ||C|| \cdot ||D||$. *In the case of the first factorization we obtain* $area(f_1) = 10$, $area(f_2) = 15$, $area(f_3) = 10$ *and* $area(f_4) = 8$. *The total area is 43, the number of covered crosses is 35, thus, the average overlapping rate is* $^{43}/_{35}$. *The second factorization is without overlapped crosses, thus its average overlapping rate is 1.*

Average values of overlapping rate are shown in Figure 4.5. Our experiments show that factor sets with minimal redundancy are produced by Hyper algorithm. It can be explained regarding the previous experiments (see Section 4.3.1), where it was shown that Hyper algorithm tends to produce a large number of trivial factors. PaNDa$^+$ tends to produce a very small number of factors with low overlapping rate. As one may clearly observe, GreConD produces factorizations with the largest overlapping rate. MDLGreConD generates a non-redundant set of factors and provides better and more stable results than MDL4BMF.

4.3.4 Discussion

Let us summarize the experimental evaluation. GreConD and Hyper are both able to explain the whole data. However, the quality of factorizations they produce is lower than the quality of MDLGreConD. More precisely,

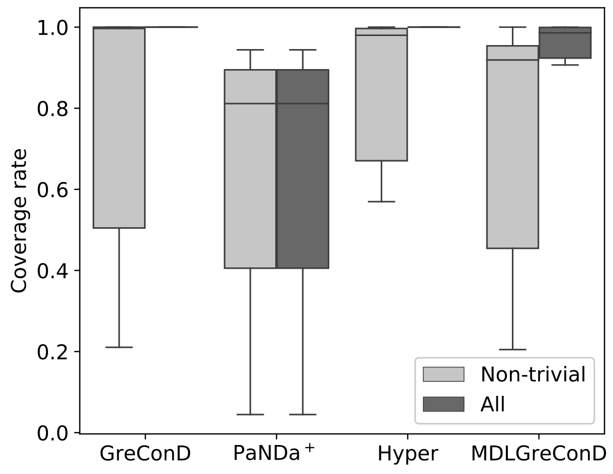

Figure 4.4: The average object coverage rate.

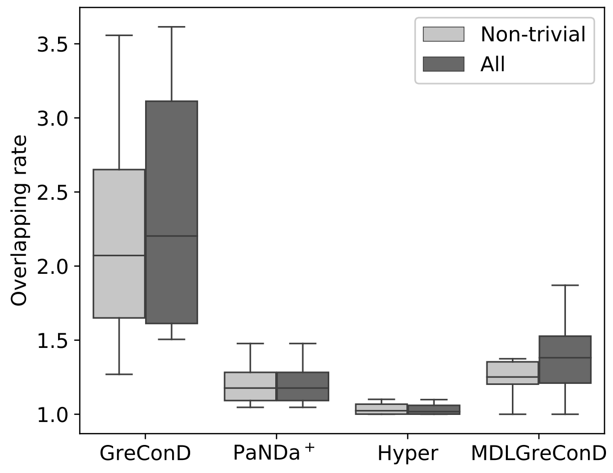

Figure 4.5: The average overlapping rate.

Hyper produces a large number of trivial factors. GreConD produces the less number of trivial factors, but with a lot of overlaps between them.

The quality of factorization obtained via PaNDa$^+$ algorithm is low as well. The factors delivered by PaNDa$^+$ cover only a small part of input data.

MDLGreConD outperforms MDL4BMF. The coverage rates of both algorithms are comparable, but MDLGreConD produces a smaller number of singleton and non-singleton factors as well. Moreover, MDL4BMF produces a set of factors with larger overlapping rate than the MDLGreConD. Additionally, the results produced by MDLGreConD are more stable than the results from MDL4BMF.

According to the experimental evaluation, MDLGreConD algorithm provides a factor set with well-balanced characteristics. The number of factors is reasonably small, factors themselves explain a large portion of data and are non-redundant.

Chapter 5

Factors Under Supervised Settings

Quality of factors is most often understood as their ability to explain data [2]. However, a lot of problems is needed to be solved under supervised settings, where class labels of objects are available.

In supervised settings, Boolean matrix $\mathbf{I} \in \{0,1\}^{m \times n}$ corresponds to m objects described by n attributes. A special target attribute refers to an object class. More formally, we define a function *class* that maps row $\mathbf{I}_{i\_}$ to its class label $c = class(\mathbf{I}_{i\_}) \in \mathcal{Y}$, the size of set $\mathcal{Y}$ is equal to the number of classes.

5.1 Key Components of Classifiers

5.1.1 Representation and labeling

For the Boolean matrix factorization $\mathbf{I} = \mathbf{A} \circ \mathbf{B}$ we consider *factor-classifier* as a tuple (f_i, c, sim), where f_i is the ith Boolean factor (represented by the ith column and ith row of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively), c is a class label given by *class* function, and *sim* is a classification strategy (see details below). In our study we assign to c a class label of the majority of objects from column $\mathbf{A}_{\_i}$. If the majority is not unique, we do not consider the factor as a classifier.

5.1.2 Strategy of classification

We focus on two common classification strategies, namely *rule-based* and *similarity-based.*

According to the first strategy, object $g = \mathbf{I}_{j\_}$ (given by n-dimensional vector) is classified by factor-classifier (f_i, c, sim) if $\mathbf{B}_{i\_} \cdot g = \mathbf{B}_{i\_}$, i.e., the object g has all attributes of factor f_i, "$\cdot$" denotes the element-wise multiplication.

With the second strategy, the object g is classified by factor-classifier (f_i, c, sim) if $similarity(\mathbf{B}_{i\_}, g) > \varepsilon$, i.e., the attributes of factor f_i are quite similar to the attributes of object g. The similarity can be defined by means of either a distance measure or an asymmetrical operator.

It should be noted that the rule-based classification strategy is a particular case of the similarity-based one, where for $g = \mathbf{I}_{j\_}$ $similarity(\mathbf{B}_{i\_}, \mathbf{I}_{j\_}) \equiv \sum_{l=1}^{n}(\mathbf{B}_{il} \rightarrow \mathbf{I}_{jl}) \equiv \sum_{l=1}^{n}(\overline{\mathbf{B}_{il}} \,|\, \mathbf{I}_{jl}) = n$. Operations $\rightarrow$ and $|$ represent logical implication and logical OR, respectively.

For the sake of simplicity, we will use (f_i, c) to denote a classifier, because in our experiments we use only the *similarity* function.

5.1.3 Responses of classifiers

We say that object g is classified by (f_i, c, sim) if $sim(\mathbf{B}_{i\_}, g) > \varepsilon$. To assign a class label to g, the responses of classifiers (f_i, c, sim) can be accounted with weights $w^g_{(f_i,c,sim)}$, e.g., precision, accuracy of f_i, or similarity between $\mathbf{B}_{i\_}$ and g. We assume that f_i does not contribute to the final decision on a class of g (the response is 0) if g is not classified by (f_i, c, sim). Again, for the sake of simplicity, we will use $w^g_{(f_i,c)}$ instead of $w^g_{(f_i,c,sim)}$.

To compute a class label of an object, the responses of classifiers (weights) are aggregated.

5.2 Experimental Evaluation

To evaluate factors under supervised settings, we use 11 different real-world datasets from UCI repository [5] binarized with tools from [13]. The characteristics of the datasets are shown in Table 5.1. In our experiments we use 10-fold cross-validation.

We compare the following BMF algorithms (some of them were already described in the previous chapter): 8M [6], GreConD [4], GreEss [3], Hyper [22], NaiveCol [7] and PaNDa$^+$ [15]. The MDLGreConD introduced in Chapter 3 is included in the comparison as well.

5.2.1 Factor As Classification Rule

In this section we examine factors as single classifiers. We study (i) the connection between factor ranks given by unsupervised and supervised quality

Table 5.1: Datasets and their characteristics.

dataset	size	density **I**	class distribution
`anneal`	898×66	0.20	0.76/0.04/0.11/0.07/0.01
`breast`	699×14	0.64	0.34/0.66
`hepatitis`	155×50	0.36	0.79/0.21
`horse colic`	368×81	0.21	0.63/0.37
`iris`	150×16	0.25	0.33/0.33/0.33
`led7`	3200×14	0.50	0.11/0.09/0.10 ($\times 8$ classes)
`mushroom`	8124×88	0.25	0.52/0.48
`nursery`	1000×27	0.30	0.32/0.34/0.34
`page block`	5473×39	0.26	0.90/0.02/0.01/0.05/0.02
`pima`	768×36	0.22	0.650/0.35
`wine`	178×65	0.20	0.33/0.40/0.27

measures, and which factors are best ones w.r.t. supervised quality measures, (ii) how well the factors summarize classes.

Connection between Supervised and Unsupervised Quality Measures

The mentioned BMF algorithms are based on a greedy strategy. The generated factors are ordered w.r.t. their importance. The importance of factors is estimated by a particular objective of an algorithm. Put it differently, the factors generated first might best explain data. Since some factor sets are very small, we cannot use correlation analysis to examine the dependence between the importance of factors (unsupervised quality measure) and their precision (supervised quality measure). To assess the connection between these measures we count how many factors we need to compute to get the best k factors w.r.t. precision. The less the number of factors we need to compute, the stronger is the connection between unsupervised and supervised quality measures.

The average number of factors is given on Figure 5.1. We note that the PaNDa$^+$ factor sets are small, but it does not mean that most important factors provide the best precision. These small values are caused by the small sizes of the PaNDa$^+$-generated factor sets. The extremely small size of factor sets produced by PaNDa$^+$ affects also the factor quality in unsupervised settings [2, 3].

Figure 5.1 shows that the lowest values correspond to the MDLGreConD factors. It means that we need to compute only few factors to get the most precise classifiers. The most important factors w.r.t. the MDLGreConD objective have relatively higher precision than the most important factors generated by other BMF algorithms.

In the next section we discuss the ability of factors to explain classes rather than data as a whole, i.e., their ability to distinguish a single class

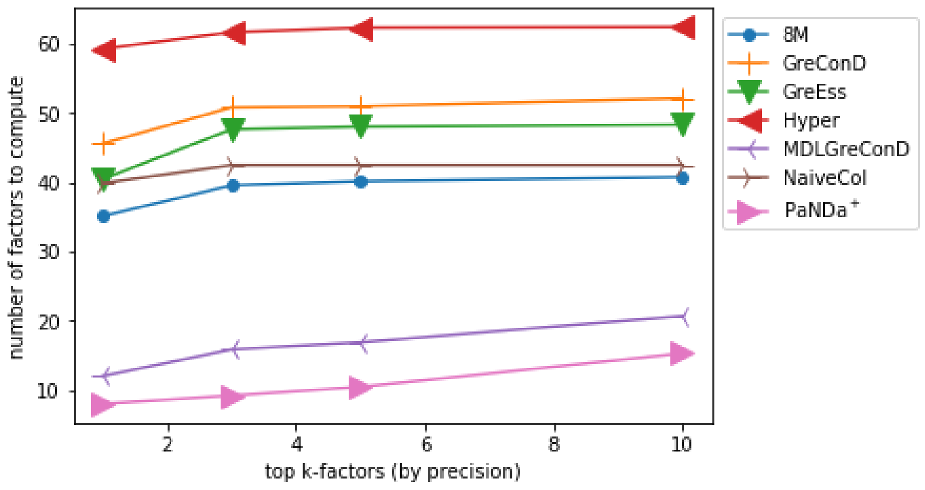

Figure 5.1: The no. of factors required to be computed in order to get k best factors w.r.t. precision.

from others.

Summary of Classes

For every factor-classifier (f_i, c) that corresponds to row $\mathbf{A}_{\_i}$ and column $\mathbf{B}_{i\_}$ we compute precision, recall and accuracy as follows:

$$prec(f_i, c) = \frac{tp}{tp + fp},\ recall(f_i, c) = \frac{tp}{tp + fn},\ accuracy(f_i, c) = \frac{tp + tn}{m},$$

where $tp = |\{\mathbf{A}_{ji} \mid \mathbf{A}_{ji} = 1, class(\mathbf{I}_{j\_}) = c, j = 1, \ldots, m\}|$ is the true positive rate, $fp = |\{\mathbf{A}_{ji} \mid \mathbf{A}_{ji} = 1, class(\mathbf{I}_{j\_}) \neq c, j = 1, \ldots, m\}|$ is the false positive rate, $fn = |\{\mathbf{A}_{ji} \mid \mathbf{A}_{ji} = 0, class(\mathbf{I}_{j\_}) = c, j = 1, \ldots, m\}|$ is the false negative rate.

Precision and accuracy characterize how well f_i describes class c. The factors with high values of these measures summarize better the given class c. The only difference between accuracy and precision is the following one: precision is the “local” class specificity (it shows how well objects from c are distinguished among the classified objects), while accuracy is the “global” class specificity (it shows how well objects from c are distinguished among all objects). Precision and accuracy give preference to classifiers with low values of fp and $fp + fn$, respectively.

The results of the experiments given in Table 5.2 show that the highest average precision is achieved for factors computed by PaNDa$^+$ (0.78, on

Table 5.2: The average values of precision on training/test sets. The best values are highlighted in bold.

	8M	GreConD	GreEss	Hyper	MDLGreConD	NaiveCol	PaNDa$^+$
anneal	0.86/0.67	0.85/0.66	0.86/0.64	0.84/0.62	0.85/0.75	0.84/0.63	**0.87**/**0.87**
breast	0.88/0.73	0.88/0.85	0.84/0.84	**0.93**/0.64	0.87/0.87	0.80/0.80	0.85/0.81
hepatitis	0.80/0.64	0.81/0.61	0.81/0.60	0.81/0.68	**0.83**/0.75	0.79/0.59	**0.83**/0.55
horse colic	0.70/0.48	0.69/0.60	0.69/0.60	0.72/0.61	0.70/0.63	0.69/0.59	**0.80**/0.56
iris	0.80/0.75	0.80/0.61	0.80/0.61	0.79/0.67	0.92/0.86	0.79/0.67	**0.96**/0.53
led7	0.40/0.44	0.33/0.32	0.33/0.32	**0.50**/0.19	0.37/0.36	0.23/0.22	0.43/0.42
mushroom	0.82/0.76	0.82/0.79	0.83/0.79	0.85/0.70	**0.87**/0.84	0.78/0.75	0.81/0.00
nursery	0.45/0.44	0.45/0.44	0.45/0.44	0.45/0.44	0.42/0.41	0.45/0.44	**0.58**/0.53
page blocks	0.82/0.35	0.82/0.46	0.84/0.43	0.78/0.33	0.80/0.51	**0.83**/0.51	0.80/0.74
pima	0.70/0.43	0.68/0.49	0.68/0.48	0.69/0.44	0.68/0.61	0.67/0.45	**0.77**/0.73
wine	0.66/0.40	0.69/0.57	0.68/0.56	0.67/0.49	0.84/0.77	0.64/0.50	**0.88**/0.66
average	0.72/0.53	0.71/0.58	0.71/0.57	0.73/0.53	0.74/**0.67**	0.68/0.56	**0.78**/0.65

average), the MDLGreConD factors also have quite high values of precision (0.74, on average). The MDLGreConD factors have the most stable quality measures (the precision on test sets is smaller by 0.07 than on training sets).

More than that, Table 5.2 provides the precision of factor-classifiers on training and test data. The precision on training data for all algorithms is quite similar (the best algorithm is PaNDa), while MDLGreConD demonstrates the best precision on test sets. It should be noticed that MDLGreConD has the smallest difference in precision for training and test data. That might indicate its ability to generalize well (i.e., it is less likely to overfit). Almost the same quality of factors, but under unsupervised settings, were observed in [2].

Chapter 6

Conclusions

In this thesis an MDL-based from-below factorization algorithm, which utilizes formal concept analysis, has been proposed. It produces a small subset of formal concepts having a low information loss rate.

The proposed algorithm does not require computing the whole set of formal concepts, that makes it applicable in practice. More than that, it computes factor sets that have better overall characteristics than factor sets computed by the existing BMF algorithms. The MDLGreConD-generated factor sets are small, contain few single-attribute factors and have a high coverage with low overlapping rate.

Bibliography

[1] S. D. Monson ans S. Pullman and R. Rees. *A survey of clique and biclique coverings and factorizations of (0,1)-matrices.* Number 14. 1995.

[2] Radim Belohlavek, Jan Outrata, and Martin Trnecka. Toward quality assessment of boolean matrix factorizations. *Inf. Sci.*, 459:71–85, 2018.

[3] Radim Belohlavek and Martin Trnecka. From-below approximations in boolean matrix factorization: Geometry and new algorithm. *J. Comput. Syst. Sci.*, 81(8):1678–1697, 2015.

[4] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20, 2010.

[5] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[6] WJ Dixon. BMDP statistical software manual to accompany the 7.0 software release, vols 1–3., 1992.

[7] Alina Ene, William G. Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert Endre Tarjan. Fast exact and heuristic methods for role minimization problems. In Indrakshi Ray and Ninghui Li, editors, *13th ACM Symposium on Access Control Models and Technologies, SACMAT 2008, Estes Park, CO, USA, June 11-13, 2008, Proceedings*, pages 1–10. ACM, 2008.

[8] B. Ganter and R. Wille. *Formal Concept Analysis Mathematical Foundations.* Springer-Verlag, Berlin, Heidelberg, 1999.

[9] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In Einoshin Suzuki and Setsuo Arikawa, editors, *Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings*, volume 3245 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2004.

[10] Peter D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning).* The MIT Press, 2007.

[11] Dmitry I. Ignatov, Elena Nenova, Natalia Konstantinova, and Andrey V. Konstantinov. Boolean matrix factorisation for collaborative filtering: An fca-based approach. In Gennady Agre, Pascal Hitzler, Adila Alfa Krisnadhi, and Sergei O. Kuznetsov, editors, *Artificial Intelligence: Methodology, Systems, and Applications - 16th International Conference, AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings*, volume 8722 of *Lecture Notes in Computer Science*, pages 47–58. Springer, 2014.

[12] Ki Hang Kim. *Boolean matrix theory and applications*, volume 70. Dekker, 1982.

[13] M. Lichman. UCI machine learning repository, 2013.

[14] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. Mining top-k patterns from binary datasets in presence of noise. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 165–176. SIAM, 2010.

[15] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. A unifying framework for mining approximate top-k binary patterns. *IEEE Trans. Knowl. Data Eng.*, 26(12):2900–2913, 2014.

[16] Tatiana P. Makhalova, Sergei O. Kuznetsov, and Amedeo Napoli. A first study on what MDL can do for FCA. In Dmitry I. Ignatov and Lhouari Nourine, editors, *Proceedings of the Fourteenth International Conference on Concept Lattices and Their Applications*, volume 2123 of *CEUR Workshop Proceedings*, pages 25–36, 2018.

[17] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008.

[18] Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 51–59. ACM, 2011.

[19] Pauli Miettinen and Jilles Vreeken. Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):18, 2014.

[20] L. J. Stockmeyer. *The Set Basis Problem is NP-complete.* Research reports. IBM Thomas J. Watson Research Division, 1975.

[21] Nikolaj Tatti, Taneli Mielikäinen, Aristides Gionis, and Heikki Mannila. What is the dimension of your binary data? In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 603–612. IEEE Computer Society, 2006.

[22] Yang Xiang, Ruoming Jin, David Fuhry, and Feodor F. Dragan. Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Discov.*, 23(2):215–251, 2011.