Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Toward quality assessment of Boolean matrix factorizations

Radim Belohlavek, Jan Outrata, Martin Trnecka*

Department of Computer Science, Palacký University Olomouc, Czech Republic

ARTICLE INFO

Article history:
Received 4 December 2017
Revised 26 March 2018
Accepted 2 May 2018
Available online 9 May 2018

Keywords:
Boolean matrix
Factorization
Algorithm
Quality assessment

ABSTRACT

Boolean matrix factorization has become an important direction in data analysis. In this paper, we examine the question of how to assess the quality of Boolean matrix factorization algorithms. We critically examine the current approaches, and argue that little attention has been paid to this problem so far and that a systematic approach to it is missing. We regard quality assessment of factorization algorithms as a multifaceted problem, identify major views with respect to which quality needs to be assessed, and present various observations on the available algorithms in this regard. Due to its primary importance, we concentrate on the quality of collections of factors computed from data, present a method to assess this quality, and evaluate this method by experiments.

1. Introduction

1.1. Problem setting

During the last decade or so, Boolean matrix factorization (BMF, also Boolean matrix decomposition) became a thoroughly explored direction in data analysis. Most of the existing research contributions focused on development of new approaches and algorithms. Performance of the developed algorithms has been observed, but its evaluation has remained on intuitive grounds or has been conducted in a simple way using the matrix distance function (we review the current approaches below). No systematic treatment of quality assessment of BMF algorithms has been pursued so far and this important topic thus remains underdeveloped. The primary aim of the present paper is to examine quality assessment of BMF algorithms in detail.

Throughout this paper we use the following notation. The set of all $n\times m$ Boolean matrices shall be denoted by $\{0, 1\}^{n\times m}$ and the particular matrices by I, J, etc. These matrices are primarily interpreted as describing a relationship between n objects and m attributes. In particular, the entry I_{ij} corresponding to ith row and jth column indicates whether the object i has (in which case $I_{ij} = 1$) or does not have ($I_{ij} = 0$) the attribute j. The ith row and jth column of I are denoted by $I_{i\_}$ and $I_{\_j}$, respectively.

The basic problem in BMF, several variants of which are considered in the literature (see below), consists in finding for a given Boolean matrix $I \in \{0, 1\}^{n\times m}$ matrices $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ for which k is small and

$$I \text{ (approximately) equals } A \circ B. \tag{1}$$

* Corresponding author..
E-mail addresses: radim.belohlavek@acm.org (R. Belohlavek), jan.outrata@upol.cz (J. Outrata), martin.trnecka@gmail.com (M. Trnecka).

https://doi.org/10.1016/j.ins.2018.05.016

In such a decomposition, k represents the number of factors (i.e. discovered hidden variables, hence the term "factorization") and $\circ$ represents the Boolean matrix product, which is defined by the formula

$$(A \circ B)_{ij} = \max_{l=1}^{k} \min(A_{il}, B_{lj}).$$

The thus introduced factor model provides us with a natural interpretation. A decomposition of a Boolean matrix I into $A \circ B$ corresponds to discovery of k factors which explain, exactly or approximately, the data represented by I. In particular, the model given by (1) is described as follows: the matrices A (the object-factor matrix) and B (the factor-attribute matrix) explain the object-attribute matrix I as follows: the object i has the attribute j iff among the factors there is a factor l such that l applies to i and j is one of the particular manifestations of l.

The approximate equality $\approx$ in (1) is measured using the well-known matrix norm (L_1-norm) $||\cdot||$ defined by

$$||C|| = \textstyle\sum_{i,j=1}^{m,n} |C_{ij}|,$$

and the corresponding metric E, which is given for Boolean matrices $C, D \in \{0, 1\}^{n\times m}$ by

$$E(C, D) = ||C - D|| = \textstyle\sum_{i,j=1}^{m,n} |C_{ij} - D_{ij}|. \tag{2}$$

1.2. Illustrative example

Consider the following 5×5 Boolean matrix I:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

One may verify that I may be (exactly) factorized by the 5×4 and 4×5 matrices A and B, i.e. $I = A \circ B$, as follows:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Since the inner dimension in the matrix product is 4, the decomposition involves 4 factors. These factors correspond to 4 rectangles in that rectangle $l \in \{1, 2, 3, 4\}$ is represented by the product $A_{\_l} \circ B_{l\_}$ of the lth column of A and the lth row of B. The 4 rectangles involved in our factorization are:

$$A_{\_1} \circ B_{1\_} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_{\_2} \circ B_{2\_} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$A_{\_3} \circ B_{3\_} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_{\_4} \circ B_{4\_} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and I equals the $\vee$-superposition (i.e. max-superposition) of these rectangles, i.e. $I = A_{\_1} \circ B_{1\_} \vee A_{\_2} \circ B_{2\_} \vee A_{\_3} \circ B_{3\_} \vee A_{\_4} \circ B_{4\_}$.

If errors are acceptable, one may ask whether there exists an approximate factorization $I \approx A \circ B$ for which the error $E(I, A \circ B)$ is not too large. The following is such an approximate factorization involving two factors:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Again, the two factors, $l \in \{1, 2\}$, correspond to the product of the lth column of A and the lth row of B:

$$A_{\_1} \circ B_{1\_} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_{\_2} \circ B_{2\_} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The matrix I may be regarded as an approximate $\vee$-superposition of these rectangles, i.e.

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

It is now easily seen that $E(I, A \circ B) = 2$: The factorization commits overcovering ($I_{35} = 0$ but $(A \circ B)_{35} = 1$) as well as undercovering ($I_{21} = 1$ but $(A \circ B)_{21} = 0$).

1.3. Relevant work

Since matrix decomposition in general represents a large subject, we directly restrict to work on Boolean matrix factorization. Motivations for exploring Boolean matrix decomposition are diverse and come from various fields. In data analysis, the interest comes from the ease of interpretation of the obtained factorization and the superiority of Boolean decomposition methods over methods designed for real-valued data, whose results for Boolean data are difficult to interpret [27]. Let us also remark that various aspects regarding Boolean matrices and their decompositon appeared in the literature on graph theory, theory of binary relations, and formal concept analysis; see for instance [5,9,12,25]. Note also that closely related to BMF is the problem of biclustering which is thoroughly studied in the field of bioinformatics, see e.g. [24], and that an exploration of the relationships of BMF and biclustering is worth further exploration.

The first work on the use of BMF in data analysis is provided by Nau [21] and Nau et al. [22], in which the authors have already been aware of the provable computational difficulty, namely NP-hardness, of the decomposition problem due to the NP-hardness of the set basis problem [26]. An early algorithm, not very well known at present though, is 8M which was developed as part of the statistical software package BMDP in the late 1970s. The interest in BMF in current data mining research is mainly due to the work of Miettinen et al. In particular, the discrete basis problem explained below, complexity results, and the now classic Asso algorithm, which we discuss later, are due to [18]. The Asso algorithm and various problems regarding BMF are further discussed in several other papers co-authored by Miettinen, e.g. [19] and the references therein. In [10], the authors examine "tiling" of Boolean data and various related problems, their complexity, and algorithms. Another significant work, [2], showed that formal concepts (which are fixpoints of the Galois connections associated to the input matrix) are natural and in fact optimal factors of Boolean matrices, described transformations between attribute and factor spaces, and proposed the GreConD algorithm discussed below. A deeper insight into approximate and exact BMF and a new algorithm, GreEss, which we use, is presented in [4]. The other recent BMF algorithms we use in our evaluation include Hyper [29] and PaNDa [14]. Let us also mention that other aspects regarding Boolean matrix factorization have been discussed in numerous other papers, including [8,13,16,17,19,20,23,28].

The above literature contains numerous experimental evaluations of the existing algorithms and also provides numerous examples of testing datasets, both real and synthetic.

1.4. Structure of our paper

In Section 2, we present our view of quality assessment as a complex problem involving a variety of aspects, and provide basic information on computational aspects, namely computational complexity and approximability. Our main emphasis is on quality assessment of individual factors and of the computed factorizations and these are examined in Section 3. Section 3.1 explores what we call a knowledge-discovery view of quality assessment: We discuss interpretability of individual factors and critically examine usage of the minimum description length principle in BMF. In Section 3.2, we discuss the reduction-of-dimensionality view, i.e. the possible applications of BMF in embedding problems involving a certain Boolean space into a Boolean space with a lower dimension. Section 3.3 examines the explanatory view, i.e. the ability of Boolean factorizations to explain the factorized Boolean data. Two basic variants of this view are employed, which are inspired by two basic factorization problems examined in the literature, and a quality metric is proposed which reflects both these variants. The explanatory view and, in particular, the proposed quality metric is experimentally evaluated in Section 4. Section 5 concludes the paper and presents topics for future research.

2. Quality assessment as a multifaceted problem

The assessment of quality of algorithms for Boolean matrix factorization is a complex issue because it involves a variety of aspects. In addition to the usual ones involved in assessments of algorithms in general, there are aspects connected to the particular nature of the factorization problem. In our view, the essential aspects may be categorized as follows:

- *computational complexity*, i.e. *time complexity* and *space complexity* of BMF problems and algorithms;
- *approximation quality* and possibly other characteristics regarding the algorithms' capability to compute optimal and sub-optimal solutions;
- *quality of factors and factorizations* obtained by the algorithms, which itself is a complex issue.

Even though quality of factors and factorizations is our main subject in this paper, which is dealt within the rest of our paper, we now provide a brief overview and comments on computational complexity and approximation quality.

As far as computational complexity of the basic BMF problem and its variants is concerned, a fundamental finding was obtained by Stockmeyer [26], who proved that deciding for a given input matrix $I \in \{0, 1\}^{n\times m}$ and a positive integer k whether there exist two matrices $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ with $I = A \circ B$ is NP-hard as a decision problem. As a consequence, the problem to find for an input matrix $I \in \{0, 1\}^{n\times m}$ two matrices $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ with $I = A \circ B$ such that k is as small as possible is NP-hard as an optimization problem. NP-hardness then applies to several variants of the basic factorization problem, including the discrete basis problem and approximate factorization problem mentioned below. Therefore, unless P=NP, an algorithm with polynomial time complexity that computes optimal factorizations does not exist.

As a consequence, the available factorization algorithms rely on various heuristics. The question of how well a solution obtained by a particular algorithm approximates the optimal solution(s) to a given problem, that is the question of *approximation factor* of a particular algorithm, is therefore of great importance. Recall that an approximation factor of an algorithm represents a guarantee of suboptimality of solutions obtained by the algorithm [11]. For example, approximation factor equal to 2 means that the algorithm is guaranteed to find a factorization with no more than $2k$ factors where k is the Boolean rank of the given input matrix (i.e. the least number of factors possible). Except for [2,10], almost no results on approximation factors of the proposed factorization algorithms are available in the literature. This situation is in a sense explained by a recent negative result on approximability [6] according to which the basic factorization problem is NP-hard to approximate within factor $n^{1-\varepsilon}$. This bound is certainly discouraging. For example, a linear approximation factor, n, implies that we only have the following guarantee: When factorizing, say, a 1000×1000 matrix with Boolean rank 50 (i.e. we have $n = 1000$), the algorithm computes a decomposition with at most $n \times 50 = 50\,000$ factors, which is a way more than what one would consider acceptable. On the other hand, the available factorization algorithms perform much better on benchmark data compared to the above approximability bound. Hence, practical significance of obtaining approximation factors of the algorithms is limited. Nevertheless, analyses of approximation factors of the available algorithms are needed because such analyses are likely to provide us with further knowledge of these algorithms as well as of the factorization problem itself.

As far as *computational complexity* is concerned, both time and space complexity certainly represent basic properties of factorization algorithms of crucial practical significance. Clearly, one prefers algorithms that require less time and space, i.e. those with smaller complexity. A common practice is that new algorithms presented in the literature are accompanied with information about their time complexity in terms of big-O-notation. Because the big-O-notation hides practically significant nuances of complexity and because the estimates of time complexity found in the existing literature are not tight as a rule, it seems more indicative to perform comparative experimental evaluations of the various algorithms. This enables to provide relative comparison of algorithms as e.g. in [4], and to say that a particular algorithm is on average, say, three times faster than another algorithm when evaluated on a certain collection of datasets. For a majority of the present BMF algorithms, neither time nor space complexity seems to be a critical problem in that from the viewpoint of the current applications of BMF, the factorizations are computed by the algorithms quickly enough. Typically, factorizations of benchmark datasets computed on an ordinary PC are computed in terms of seconds or a few tens of seconds [4]. The Tiling algorithm [10] and the conceptually similar Algorithm 1 of [2] represent notable exceptions because they need to compute first a large set of candidate factors.

3. Quality of factors and factorizations

We now consider quality of factors and factorizations, which is the main objective of our paper. The issue itself of whether and to what extent a set of factors delivered for a particular input data by a particular factorization algorithm involves several aspects, which are not properly discussed in the literature. We propose to recognize three basic aspects, which we address in the following order. In Section 3.1, we examine interpretability of individual factors, which pertains to *knowledge-discovery view*. The next two aspects concern the whole set of extracted factors rather than individual factors, and are discussed in Sections 3.2 and 3.3. The first is the *reduction-of-dimensionality view*, whose substance is the question of whether the factors may be thought of as new attributes using which the original data may be embedded to a lower dimensional space and which enables efficient processing of the data. The second is the *explanatory view*, whose substance is the question of whether and to what extent do the factors explain the input data well.

3.1. Knowledge-discovery view

3.1.1. Interpretability of individual factors

Interpretability of factors, arguably the most important aspect from *knowledge-discovery view* of factor analysis, is an important issue in general and pertains to every factorization method: If useful knowledge is to be obtained by means of factor analysis, the factors must first of all be well interpretable and comprehensible by users. Relevant to factorization of Boolean data is the following geometric view of the factor model.

The view was made explicit e.g. in our paper [4] and asserts that a decomposition of a Boolean matrix $I \in \{0, 1\}^{n\times m}$ may be thought of (in a one-to-one manner) as a coverage of the entries of I containing 1s by *rectangles* (cf. Section 1.2). Recall that by a rectangle we mean a matrix in which the entries containing 1s form a rectangle, or may be brought to the shape of a rectangle by permuting rows and columns. That is, a rectangle is a matrix $J \in \{0, 1\}^{n\times m}$, for which there exist vectors $C \in \{0, 1\}^{n\times 1}$ and $D \in \{0, 1\}^{1\times m}$ such that $J = C \circ D$. With the notion of rectangle at hand, it is easy to verify that a Boolean product $A \circ B$ of Boolean matrices $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ may alternatively be regarded as a max-superposition of rectangles $J_l = A_{\_l} \circ B_{l\_}$, $l = 1, \ldots, k$, i.e.

$$(A \circ B)_{ij} = \max_{l=1}^{k} (J_l)_{ij}.$$

In these expressions, $A_{\_l}$ and $B_{l\_}$ denote the lth column of A and the lth row of B, respectively. The thus established geometric view, rarely recognized in the literature on BMF, tells us that finding a decomposition (exact or approximate) of I into $A \circ B$ using k factors means finding a coverage (exact or approximate) of the 1s in I by k rectangles full of 1s and is of great practical significance for understanding the logic of Boolean factorization.

The geometric view also tells us that no matter how one computes factors (i.e. independent of any particular algorithm), *every particular factor*, say factor l, *may be identified with a rectangle* J_l full of 1s, or alternatively, as a pair $\langle A_{\_l}, B_{l\_}\rangle$ of the above-mentioned vectors, for which $J_l = A_{\_l} \circ B_{l\_}$. Such a factor may therefore rather naturally be interpreted as a new, abstract attribute (or property), generally distinct from the m original attributes. This new attribute is characterized by those of the m original attributes j for which $B_{lj} = 1$. These attributes j are viewed as particular manifestations of factor l and may be used as descriptors of the factor. If, for instance, the attributes j of factor l are 2, 5, and 6, then factor l may be described as an attribute which applies to a given object exactly when the object possesses all the attributes 2, 5, and 6.

The above considerations demonstrate that factors in Boolean matrix factorization may easily be interpreted. This represents a considerable advantage over other matrix factorization methods which are sometimes employed for Boolean data. This is confirmed by our experience and several existing studies including [27] in which the problem of interpretability of factor analysis of Boolean data using methods originally developed for real-valued matrices is critically examined.

3.1.2. Criticism of BMF methods based on the MDL principle

While the above considerations on interpretability have not been addressed explicitly so far, they have to a certain extent been partially addressed in the literature. On the other hand, the following important aspect has not been addressed at all and seems currently not properly understood.

Namely, some authors such as those of the PaNDa algorithm [14] aim for finding factors with small description length in the hope to obtain good factorizations. Such approach is inspired by the well-known minimum description length principle (MDL) which itself goes back to Ockham's razor. Translating to BMF, this principle says that factorizations with small total description length of its factors are preferable. In our notation, the description length of factor $\langle A_{\_l}, B_{l\_}\rangle$ is defined as the sum

$$dl(A_{\_l}, B_{l\_}) = ||A_{\_l}|| + ||B_{l\_}||,$$

of the height and width of the corresponding rectangle, which is the sum of the number $||A_{\_l}||$ of 1s in $A_{\_l}$ and the number $||B_{l\_}||$ of 1s in $B_{l\_}$. The PaNDa algorithm actually employs minimizing the sum of description lengths of all the computed factors as part of its cost function.

We claim that this view is flawed and that, instead, well interpretable factors correspond to *maximal rectangles* rather than to rectangles with a small description length. Put succinctly, it is generally not a good idea to remove attributes or objects from factors to shorten their description length because then, interpretability of factors suffers. This rationale stems from *formal concept analysis* [9], in which the basic patterns employed, called formal concepts, are in fact maximal rectangles of 1s. Our preference of maximal rectangles, articulated already in [2], derives from our experience with analyses of many datasets. We do not claim that the MDL approach is a priori wrong. But we do claim that it should be justified by concrete data analyses. Otherwise it remains a theoretical construct—an unverified hypothesis regarding usefulness of factors. Our argument against the MDL approaches in BMF is explained by the following generic example.

Example 1. Consider two taxa: Taxon A, which is characterized by 10 attributes, and taxon B, which is a subtaxon of taxon A and which is characterized by the 10 attributes of taxon A plus 5 additional attributes. That is, a natural description of taxon A is a conjunction of the 10 attributes which characterize A, while a description of taxon B is a conjunction of the 15 attributes (10 of taxon A and 5 additional ones) characterizing B. Suppose the dataset I consists of 50 exemplars of taxon A of which 25 are also exemplars of taxon B and that there are 20 attributes in total. Such a dataset is depicted in Fig. 1(a). There is a natural factorization of the 50×20 matrix I by two overlapping factors, which is depicted in Fig. 1(b): The first

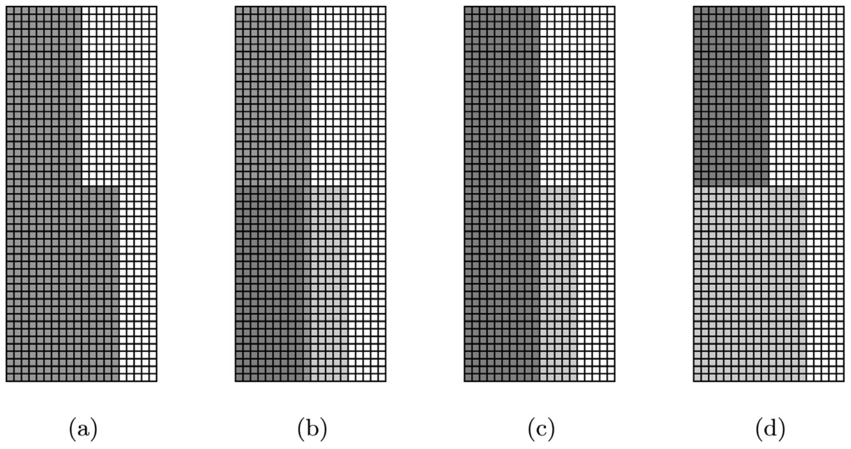


Fig. 1. (a) Dataset I from Example 1; (b) natural factorization of I by two taxa as factors; (c) and (d) factorization of I according to the MDL principle (factors do not correspond to taxa).

corresponds to the 50×10 rectangle representing taxon A; the second to the 25×15 rectangle representing taxon B. Both factors are easily interpretable: the first is characterized by the 10 attributes characteristic of taxon A and the second by the 15 attributes characteristic of taxon B.

Now, according to MDL, this set of two factors is not good because it has overly large description length, namely $(50 + 10) + (25 + 15) = 100$. Instead, MDL suggests using, as the first factor, the 50×10 rectangle representing taxon A and, as the second factor, the 25×5 rectangle representing the exemplars of taxon B and their 5 additional attributes. This factorization is depicted in Fig. 1(c): The total MDL of these two factors is $(50 + 10) + (25 + 5) = 90$. The second factor, however, is difficult to interpret: one may not interpret it as taxon B because among the characteristic attributes of this factor, there are only 5 attributes, hence there are 10 attributes missing from description of taxon B. Alternatively, MDL suggests using the 25×15 rectangle as factor representing taxon B and the 25×10 rectangle representing the exemplars of taxon A not included in B and the 10 attributes of taxon A, resulting in total description length $(25 + 15) + (25 + 10) = 75$—even a smaller description length. This factorization is depicted in Fig. 1(d). Again, however, interpretability suffers because the 25×10 rectangle may not be interpreted as taxon A for it does not cover the 25 missing exemplars of taxon B.

3.2. Reduction-of-dimensionality view

As is clear from the above discussion, a factorization of an input $n \times m$ Boolean matrix I using k factors can be regarded as a discovery of k new Boolean variables, i.e. new attributes. If $I \approx A \circ B$, then A describes the n given objects, originally described in I by the m input attributes, in terms of k factors (new attributes) and B describes a relationship between the k factors and the m original attributes. Therefore, the n objects may either be described in the m-dimensional space of the original attributes or in the k-dimensional space of factors. If $k < m$, then switching from the attribute space to the factor space actually represents what is commonly referred to as reduction of dimensionality, or, more precisely, reduction of dimension. In this case, the simple ratio $\frac{m-k}{m}$ may be taken as a measure of such a reduction.

An important aspect justifying the usefulness of various factorization methods for real-valued data derives from the possibility to process data in the less-dimensional factor space rather than the original space involving possibly a high number of variables (attributes in the case of Boolean data). The original variables (attributes) represent the directly observable features and as such they may be mutually redundant, impaired by error or otherwise flawed. On the other hand, good factors obtained from data should capture the fundamental aspects of the data and should be free of the flaws manifested by the original attributes. Then, processing of the data in the factor space proves to be more efficient than processing the data in the original attribute space. Put conversely, if the factor space provides us with a considerably better processing of data compared to the original attribute space, we may regard the factors as good ones. This perspective allows us to compare the quality of sets of factors delivered by particular factorization algorithms.

Even though much effort has been devoted to the design of new algorithms for factorization of Boolean data, virtually no attention has been paid to how the factorization methods may further be utilized in processing Boolean data, e.g. in machine learning, which is a rather surprising and strange situation. For one, this situation is in sharp contrast to that with factorization methods for real-valued data, which are being commonly employed in several fields including machine learning. Furthermore, the lack of studies on possible utilization of dimensionality reduction due to BMF implies a lack of the much needed feedback for quality assessment of the various BMF algorithms.

An exception is, nevertheless, represented by [23] in which the author employs the transformation formulas between the factor and attribute spaces proposed in [2] in the problem of classification of Boolean data. Put briefly, the idea is to first factorize the classified Boolean data using a BMF algorithm and then to do classification in the factor space rather than in the original attribute space. Interestingly, such an approach results in a significant increase of classification accuracy. In [3], the authors asked the question of which factorization methods perform well in the described scenario. It turned out that there are significant differences between the various factorization methods employed. That is to say, with respect to this particular employment of BMF, some algorithms turn out as good ones while some as not so good. We omit the details of [3] due to limited scope but also because our main point here is to point out the fact that studies on employment of BMF in machine learning and other areas may not only demonstrate usefulness of BMF in general but also provide a useful, very concrete feedback regarding quality of BMF algorithms. Without such feedback, research in BMF algorithms might easily become a speculative theorizing.

3.3. *Explanatory view*

The explanatory view, as we propose to call it, pertains to the fact that knowledge regarding the data that is provided by the individual factors may be considered valuable only if the set of factors as a whole explains the data well. In this regard, we propose two basic views, which we call the DBP view and the AFP view. These views are inspired by two well-known problems in BMF, namely the *discrete basis problem* (DBP) and the *approximate factorization problem* (AFP). Before we discuss these problems and the corresponding views, we turn to the question of quantitative explanation of data by factors.

A straightforward way to assess how the given data represented by a matrix $I \in \{0, 1\}^{n\times m}$ is explained by a set of k factors represented by object-factor and factor-attribute matrices $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$, respectively, is via the distance function $E(I, A\circ B)$ defined by (2), i.e.

$$E(C, D) = \sum_{i,j=1}^{m,n} |C_{ij} - D_{ij}|.$$

Furthermore, we employ the following function of $A \in \{0, 1\}^{n\times l}$ and $B \in \{0, 1\}^{l\times m}$ to measure the quality of decompositions computed by the algorithms, and call this function the *coverage quality* of the first l factors delivered by the particular algorithm:

$$c(l) = 1 - E(I, A(l) \circ B(l))/||I||, \tag{3}$$

where $A(l)$ and $B(l)$ denote the $n \times l$ and $l\times m$ matrices that correspond to the first l factors. Let us note that analogous functions are used, e.g., in [2,4,10,18]. The various modifications of the coverage function that appeared in the literature may be regarded as predecessors of our attempt to address comprehensively the problem of quality assessment. For our purpose, we shall observe how the value of c increases when adding additional factors, i.e. observe the values $c(l)$ for $l = 0, \dots, k$, where k is the number of factors delivered by a particular algorithm.

3.3.1. *The DBP view—The importance of the first few factors*

The explanatory view involves two basic particular views. We call the first one the DBP view because it is connected to the well-known DBP problem:

- *Discrete Basis Problem* [18]:
  Given $I \in \{0, 1\}^{n\times m}$ and a positive integer k, find $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ that minimize $||I - A \circ B||$.

The DBP view thus emphasizes the importance of the first few factors, which are assumed to be the most important ones. From this viewpoint, good BMF algorithms are those with high values of coverage, $c(l)$, for small numbers l of factors (say $l = 2, 5, 10$).

3.3.2. *The AFP view—The importance of explaining a large portion of data*

The second view we recognize is the AFP view because it is connected to the AFP problem:

- *Approximate Factorization Problem* [2]:
  Given $I \in \{0, 1\}^{n\times m}$ and prescribed error $\varepsilon \geq 0$, find $A \in \{0, 1\}^{n\times k}$ and $B \in \{0, 1\}^{k\times m}$ with k as small as possible such that $||I - A \circ B|| \leq \varepsilon$.

The AFP view stresses out the need to cover (and in this sense to explain) a large portion of the input data. From this viewpoint, good BMF algorithms are those which are able to attain a high coverage $c(l)$, say $c(l) = 0.8, 0.9, 1.0,$ with a small number l of factors.

3.3.3. *Combined view—A proposed quality metric*

Examination of the existing literature reveals that no commonly accepted prospect of what it means that a BMF algorithm performs well from the explanatory point of view is available. A natural criterion, which we propose and examine below, may be derived from the following intuitive requirement.

Both the DBP view and the AFP view in fact represent two boundary views. The DBP view is particularly relevant when one seeks a few important factors. Since Boolean factors are naturally interpreted as clusters (cf. the above section on interpretability of factors), a typical situation of this kind is when a small number of informative clusters are sought in the data. On the other hand, the AFP view is actually present when one desires a good representation of the whole data by factors. This is, for example, the case in the above-mentioned application of BMF as preprocessing method in classification

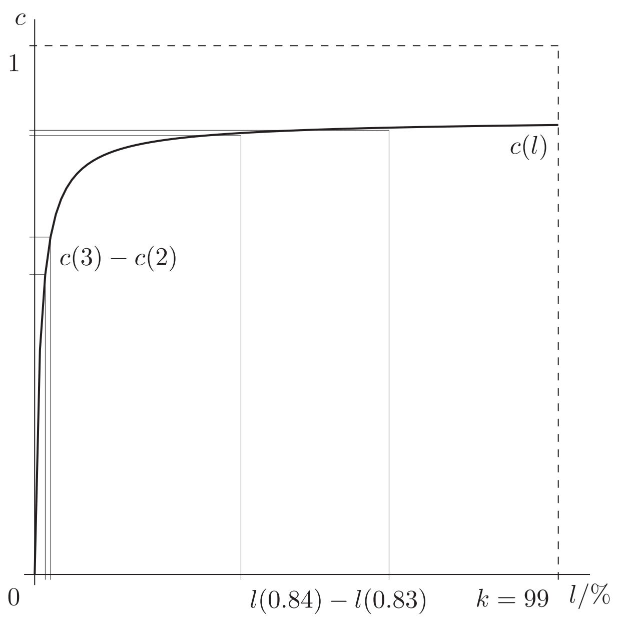


Fig. 2. Coverage quality function.

of Boolean data. If only the DBP view was taken into account, one could end up with an algorithm that is not capable of computing highly accurate factorizations; on the other hand, taking only the AFP view into account could result in an algorithm delivering as its first couple of factors some factors that are not very significant in that they do not have a high coverage.

A natural, more general view than the DBP and the AFP view is therefore a combined view, which reflects both these views. According to this combined view, a good algorithm computes a set of factors such that the first couple of them have a reasonably high coverage and, conversely, a high coverage, possibly close to 1, is obtained with a reasonably small number of factors. For a good BMF algorithm, therefore, the coverage c as a function of the number l (the first l factors produced by the algorithm) is required to be increasing in l, have reasonably large values even for small values of l (i.e. c needs to grow steeply in the beginning), and have the value $c(k)$ equal or close to 1 for a reasonably small number k of factors computed by the algorithm.

The measure of quality of BMF algorithms we propose therefore reflects the combined view and is based on the coverage quality function $c(l)$ defined by (3). A typical shape of $c(l)$ is depicted in Fig. 2. By means of this measure we attempt to "capture" the desired properties of the coverage quality function c demanded by the DBP and the AFP.

According to the definition of DBP above, a BMF algorithm should, for a given number l of factors, minimize the error $E(I, A \circ B)/||I||$ of decomposition, i.e. maximize the coverage $c(l)$. Thus, for a good BMF algorithm, the function c should be increasing in l and should be increasing steeply, in particular for small values of l. We propose to evaluate the increase of c as follows. For a particular $l \in \{1, \ldots, k\}$, where k is the final number of factors delivered by the given algorithm, the increase of coverage c due to the lth factor is $c(l) - c(l-1)$; for $l = 3$, see Fig. 2. The requirement that such increase is more significant for small values of l is reflected by weights which are greater for small values of l than for large values of l. We use the weights $1/l$ assigning the weight 1 to the first factor, 1/2 to the second factor, and so on. Hence, the gain $q'_{\mathrm{DBP}}(l-1, l)$ in quality due to factor $l \in \{1, \ldots, k\}$ reflecting the DBP view is naturally quantified by

$$q'_{\mathrm{DBP}}(l-1, l) = \frac{c(l) - c(l-1)}{l}. \tag{4}$$

The quantification of the gain reflecting the AFP view is analogous (DBP and AFP are in a sense dual), only a little more technically involved. According to the definition of AFP, a BMF algorithm should, for a given admissible amount ε of error, minimize the number l of factors using which the computed decomposition satisfies $E(I, A \circ B)/||I|| \leq \varepsilon$. Thus, for a good BMF algorithm, the number l of factors, considered as a function of $E(I, A \circ B)/||I||$, should be decreasing in $E(I, A \circ B)/||I||$ and should be decreasing steeply for low values of $E(I, A \circ B)/||I||$. Put in terms of the coverage quality function $c = 1 - E(I, A \circ B)/||I||$, the number l of factors considered as a function of c, should be decreasing as its argument c decreases and should be decreasing steeply, in particular for high values of c. However, while we evaluate the increase of c in the DBP view in the discrete steps from $l-1$ to l between the individual factors, see (4), we now need to choose discrete steps, i.e. intervals in the values of c, to evaluate the decrease of l. We choose the intervals to correspond to weights, which we again use to reflect the significance of the decrease of l for higher values of c. Naturally, the weights should be greater for higher values of c than for lower values of c. Similarly to the DBP view above, we use the weights $1/((1-c) \cdot 100)$ for $c \in \{0.99, 0.98, \ldots, 0\}$. This assigns the weight 1 to 1% decrease of c (from $c = 1.0$ to $c = 0.99$), 1/2 to 2% decrease (from $c = 1.0$ to $c = 0.98$), and so on. The decrease of l, as a function of c, in the interval $[c, c + 0.01]$ is then evaluated similarly as in the DBP view, namely as $l(c + 0.01) - l(c)$; for $c = 0.83$, see in Fig. 2 in which $l(c)$ is obtained as explained below. In order to express it relatively to the final number k of factors delivered by an algorithm we divide it by k (notice that this is the same as with the increase of c in the DBP view which is expressed relatively to the maximum $c = 1$). The value $l(c)$ for $c \in \{1, 0.99, \ldots, 0\}$ is naturally obtained from $c(l)$ as the following approximation:

$$l(c) = l - 1 \text{ such that } c(l-1) < c + 0.005 \leq c(l). \tag{5}$$

The gain in quality, $q'_{\mathrm{AFP}}(c, c + 0.01)$, reflecting the AFP view, for an interval from $c \in \{\tilde{c}(k) - 0.01, \ldots, 0\}$ to $c + 0.01$, where $\tilde{c}(k)$ is $c(k)$ rounded to hundredths, is then defined by

$$q'_{\mathrm{AFP}}(c, c + 0.01) = \frac{c(k) \cdot (l(c + 0.01) - l(c))}{k \cdot (1 - c) \cdot 100}. \tag{6}$$

Multiplying by $c(k)$ is to reflect the requirement for AFP to have the final coverage value $c(k)$ for the final number k of factors delivered by an algorithm equal or close to 1: Lower values $c(k)$ make $q'_{\mathrm{AFP}}(c, c + 0.01)$ smaller.

However, we still need to take into account the following aspect. While $l(0)$ is always equal to 0, the value $l(\tilde{c}(k))$ is equal to k which is varying for the various particular algorithms because different algorithms deliver different numbers k of factors. To be able to reasonably compare different algorithms, we need to relate the numbers k of factors delivered by these algorithms to a common value, i.e. calibrate. Relating them to the Boolean rank of the given relation is not computationally feasible because computing the Boolean rank is NP-hard. A natural value to be used for our purpose is the number m of attributes as we naturally expect k to be not higher than m. The smaller than m the number k of factors computed by the algorithm, the better the algorithm, and vice versa. We thus quantify the ability of an algorithm to achieve the coverage $\tilde{c}(k)$ using a small number of factors by means of the quantity $q''_{\mathrm{AFP}}(\tilde{c}(k))$ defined as

$$q''_{\mathrm{AFP}}(\tilde{c}(k)) = \begin{cases} \frac{c(k) \cdot (m-k)}{m \cdot (1 - \tilde{c}(k)) \cdot 100} & \text{for } \tilde{c}(k) \neq 1, \\ \frac{(m-k)}{m} & \text{for } \tilde{c}(k) = 1. \end{cases} \tag{7}$$

Now, the functions proposed above evaluate the quality of a single decomposition consisting of the first l factors relatively to a decomposition consisting of the first $l - 1$ factors (in the case of q'_{DBP}) or having the coverage quality $c + 0.01$ relatively to the decomposition having the coverage quality c (for q'_{AFP}) delivered by a particular BMF algorithm. However, the quality of the algorithm is expressed by the entire coverage function c as a function of l, i.e. is expressed by the graph of c. Hence, to evaluate the quality of a BMF algorithm for a particular input data from both the DBP view and the AFP view, we need to take into account all decompositions of the input data produced by the algorithm during its run. That is, we need to consider q'_{DBP} for l from 1 up to k, consider q'_{AFP} for c from $\tilde{c}(k) - 0.01$ down to 0, and consider $q''_{\mathrm{AFP}}(\tilde{c}(k))$. Moreover, to be able to combine the DBP and the AFP views and to emphasize one or the other in our assessment, we use a weight $w \in [0, 1]$. Putting

$$q_{\mathrm{DBP}}(l) = \sum_{l'=1}^{l} q'_{\mathrm{DBP}}(l' - 1, l'), \tag{8}$$

$$q_{\mathrm{AFP}}(c) = \sum_{c' \in \{\tilde{c}(k) - 0.01, \ldots, c\}} q'_{\mathrm{AFP}}(c', c' + 0.01) \quad + \quad q''_{\mathrm{AFP}}(\tilde{c}(k)), \tag{9}$$

we therefore define the quality of a given BMF algorithm for a given input Boolean matrix as the number q given as

$$q = w \cdot q_{\mathrm{DBP}}(k) + (1 - w) \cdot q_{\mathrm{AFP}}(0). \tag{10}$$

Remark 1. Note that while $q_{\mathrm{DBP}}(l)$ ranges from 0 to 1, the value of $q_{\mathrm{AFP}}(c)$ can be greater than 1 and can also be negative. The former can happen when $k < m$ and $c(k)$ is close to 1 at the same time, i.e. for a good BMF algorithm. The latter can happen when $k > m$, i.e. for a poor BMF algorithm. For the commonly used BMF algorithms, however, the value of $q_{\mathrm{AFP}}(c)$ also lies between 0 and 1.

For illustration, the graphs of $q_{\mathrm{DBP}}(l)$ and $q_{\mathrm{AFP}}(c)$ corresponding to the coverage function $c(l)$ from Fig. 2 are depicted in Fig. 3 (for $m = 100$). Note that the values of $q_{\mathrm{AFP}}(c)$ are put on the horizontal axis used for l and that these values are displayed in percents, i.e. we display $q_{\mathrm{AFP}}(c) \cdot 100$ (hence the graph of $q_{\mathrm{AFP}}(c)$ is flipped and rotated).

4. Experimental evaluation

We now present experimental evaluation of our quality metric for BMF algorithms. For space reasons, we restrict to the best known factorization algorithms and to selected real datasets. We examine the algorithms' performance with respect to the DBP view, the AFP view, and the combined view as described above.

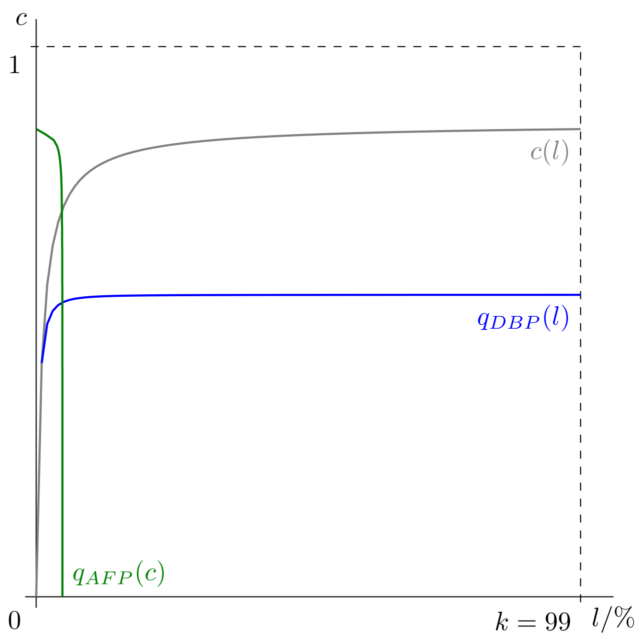


Fig. 3. Quality of decomposition reflecting the DBP view ($q_{\mathrm{DBP}}(l)$) and the AFP view ($q_{\mathrm{AFP}}(c)$).

Table 1
Datasets and their characteristics.

Dataset	Size	Density	Avg. assoc.	Med. support
DBLP	6980 × 19	0.130	0.175	0.101
Domino	79 × 231	0.040	0.642	0.025
Emea	3046 × 35	0.068	0.211	0.047
Firewall 1	365 × 709	0.124	0.459	0.154
Healthcare	46 × 46	0.702	0.818	0.619
Chess	3196 × 76	0.487	0.480	0.470
Mushroom	8124 × 119	0.193	0.193	0.074
Paleo	501 × 139	0.051	0.092	0.042
Zoo	101 × 28	0.305	0.298	0.233

4.1. Datasets used in experimental evaluation

Our evaluation involves the datasets DBLP,[1] Domino [8], Emea [8], Firewall 1 [8], Healthcare [8] Mushroom [1], Chess [1], Paleo,[2] and Zoo [1]. These datasets are well known and commonly used in the literature on BMF. Table 1 displays the basic characteristics of our datasets: The number of objects × number of attributes (in the column labeled Size), the density, i.e. percentage of 1s in the entries of the dataset (column Density), the average value in the association matrix of I [18] (column Avg. assoc.), and the median of the attributes' support (that is, the number of objects which have a given attribute; column Med. support).

4.2. Algorithms used in experimental evaluation

8M is an early BMF algorithm which is part of the BMDP statistical software since the late 1970s. In our paper, we use the version of 8M described in the 1992 edition [7], which accompanies release 7 of BMDP. Asso [18] is a classic, probably the most frequently discussed algorithm in the literature on Boolean matrix factorization. It performs a greedy search for factors which are constructed using the so-called association matrix induced by the input dataset I. The algorithm has been proposed to solve the discrete basis problem and commits both types of errors, i.e. the undercovering $E_u = |\{\langle i, j\rangle;\, (A \circ B)_{ij} < I_{ij}\}|$ and the overcovering $E_o = |\{\langle i, j\rangle;\, (A \circ B)_{ij} > I_{ij}\}|$. GreConD [2] (denoted Algorithm 2 in that paper) performs a particular greedy search for factors which are in the form of formal concepts in the concept lattice associated to the input dataset I and are computed "on demand", avoiding thus the need to compute the whole concept lattice in advance. GreConD is

[1] http://www.informatik.uni-trier.de/~ley/db/.
[2] NOW public release 030717, available from http://www.helsinki.fi/science/now/.

Table 2
Coverage of data by the first k factors (DBP view).

Dataset	k	8M	Asso	GreConD	NaiveCol	PaNDa	Hyper	GreEss	PaNDa$^+$
DBLP	1	0.033	0.111	0.131	0.111	0.122	0.111	0.111	0.122
	2	0.151	0.217	0.238	0.217	0.208	0.217	0.217	0.208
	5	0.305	0.475	0.468	0.475	0.217	0.475	0.475	0.217
	10	0.414	0.738	0.692	0.738	0.217	0.738	0.738	0.217
Domino	1	0.019	0.553	0.548	0.286	0.548	0.286	0.286	0.548
	2	0.573	0.700	0.697	0.449	0.586	0.449	0.449	0.586
	5	0.729	0.837	0.834	0.801	0.678	0.766	0.801	0.633
	10	0.847	0.937	0.948	0.942	0.678	0.834	0.942	0.633
Emea	1	0.199	0.168	0.163	0.077	0.142	0.077	0.077	0.132
	2	0.218	0.244	0.239	0.153	0.178	0.153	0.153	0.162
	5	0.287	0.443	0.423	0.347	0.184	0.347	0.347	0.169
	10	0.427	0.691	0.665	0.592	0.184	0.592	0.592	0.169
Firewall 1	1	0.726	0.726	0.651	0.651	0.131	0.019	0.648	0.126
	2	0.743	0.818	0.804	0.804	0.491	0.027	0.801	0.145
	5	0.839	0.908	0.940	0.932	0.491	0.048	0.942	0.862
	10	0.909	0.917	0.981	0.976	0.491	0.083	0.981	0.862
Healthcare	1	0.653	0.666	0.636	0.636	0.677	0.636	0.636	0.636
	2	0.779	0.901	0.910	0.910	0.871	0.656	0.910	0.910
	5	0.948	0.938	0.965	0.964	0.895	0.713	0.964	0.910
	10	0.953	0.941	0.988	0.991	0.895	0.795	0.994	0.910
Chess	1	0.600	0.497	0.197	0.119	0.448	0.172	0.236	0.305
	2	0.610	0.574	0.340	0.177	0.511	0.178	0.297	0.426
	5	0.632	0.628	0.515	0.323	0.578	0.187	0.455	0.550
	10	0.652	0.703	0.655	0.461	0.612	0.281	0.605	0.635
Mushroom	1	0.218	0.226	0.129	0.129	0.215	0.127	0.168	0.168
	2	0.319	0.323	0.234	0.234	0.291	0.129	0.260	0.258
	5	0.453	0.461	0.461	0.398	0.346	0.205	0.410	0.409
	10	0.508	0.555	0.582	0.512	0.346	0.339	0.546	0.518
Paleo	1	0.007	0.027	0.027	0.027	0.018	0.027	0.027	0.013
	2	0.018	0.047	0.047	0.047	0.022	0.047	0.047	0.027
	5	0.070	0.105	0.106	0.105	0.040	0.105	0.105	0.042
	10	0.111	0.182	0.181	0.182	0.040	0.182	0.182	0.042
Zoo	1	0.188	0.341	0.265	0.190	0.271	0.096	0.244	0.209
	2	0.457	0.507	0.427	0.353	0.357	0.189	0.406	0.303
	5	0.622	0.739	0.703	0.603	0.524	0.415	0.660	0.326
	10	0.665	0.856	0.853	0.834	0.539	0.676	0.849	0.326

designed to compute exact decompositions. However, the algorithm may be stopped after it computes the first k factors or after the committed error E does not exceed ε, in which case it provides approximate solutions to DBP and AFP, respectively. NaiveCol [8] is a simple algorithm which performs a straightforward greedy search for factors. The factors are the columns in the input matrix I. The algorithm may be used for both AFP and DBP. PaNDa [14] is primarily designed for a modification of the DBP problem. It employs the minimum description length principle in the criteria to select factors. PaNDa+ [15] is an enhanced version of PaNDa, which allows to incorporate various cost functions to select good factors. In particular, we use the cost function proposed by the authors, which is the sum of the description length of the so far computed set of factors and the error committed by this set of factors. Hyper [29] employs as factors the so-called hyperrectangles (in fact these are rectangles in the input matrix I) and computes an exact decomposition of the input matrix. GreEss [4] computes from the input matrix I a smaller matrix (the so-called essential part of I) to which one may restrict when computing a factorization. Like GreConD, the algorithm employs formal concepts of I as factors and may be used for both AFP and DBP.

Let us note that all the algorithms are deterministic, i.e. stop the computation for any inputs and always deliver the same factorizations for the same inputs. Only some of them, however, are capable of computing exact decompositions, which property is discussed as part of Section 4.3.

4.3. Assessment of quality

The purpose of this section is twofold. For one, we intend to demonstrate on real datasets a typical behavior of the algorithms selected for our evaluation as well as demonstrate that the DBP view and the AFP view represent two basic and in a sense complementary views on the algorithms' performance. Secondly, we intend to demonstrate that our metric may reasonably be used to assess performance of a given algorithm from the DBP view, the AFP view, and the combined view.

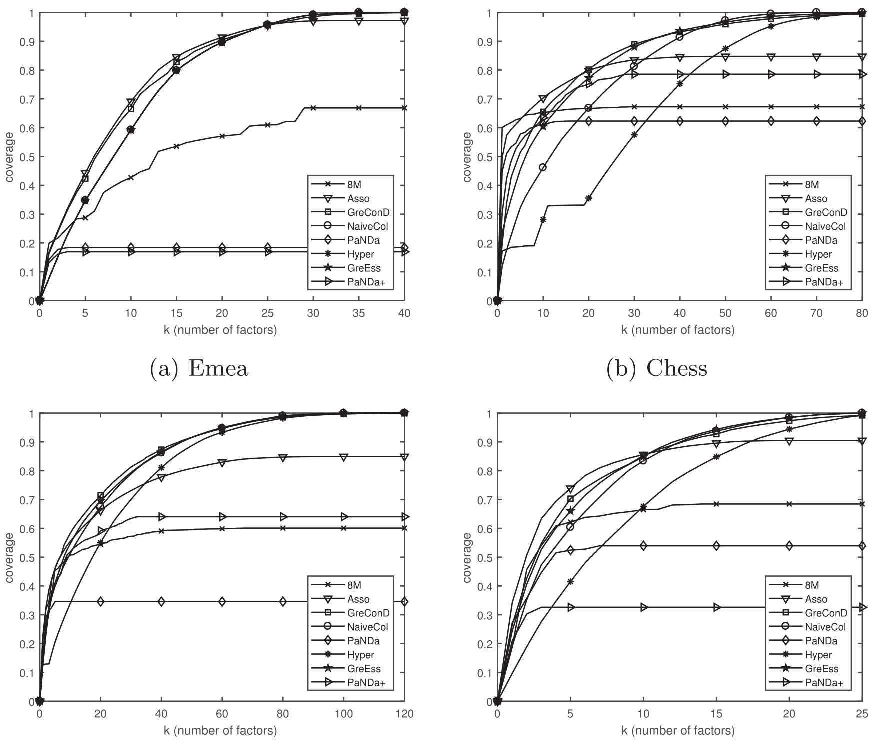
The algorithms' performance from the DBP and the AFP point of view is illustrated in Table 2 and Table 3, respectively. In particular, Table 2 presents the values of the coverage function $c(k)$ defined by (3) for the first $k = 1, 2, 5,$ and 10 factors.

Table 3
Number of factors needed for prescribed coverage c (AFP view).

Dataset	c	8M	Asso	GreConD	NaiveCol	PaNDa	Hyper	GreEss	PaNDa$^+$
DBLP	0.80	NA	12	13	12	NA	12	12	NA
	0.90	NA	15	17	15	NA	15	15	NA
	0.95	NA	17	19	17	NA	17	17	NA
	1.00	NA	19	21	19	NA	19	19	NA
Domino	0.80	8	4	4	5	NA	7	5	NA
	0.90	NA	8	8	9	NA	26	9	NA
	0.95	NA	12	11	11	NA	45	11	NA
	1.00	NA	NA	21	20	NA	79	20	NA
Emea	0.80	NA	13	15	16	NA	16	16	NA
	0.90	NA	19	20	21	NA	21	21	NA
	0.95	NA	25	25	25	NA	25	25	NA
	1.00	NA	NA	42	34	NA	35	34	NA
Firewall 1	0.80	5	2	2	2	NA	193	2	4
	0.90	7	5	4	4	NA	223	4	NA
	0.95	NA	NA	6	7	NA	239	6	NA
	1.00	NA	NA	66	71	NA	365	64	NA
Healthcare	0.80	3	2	2	2	2	11	2	2
	0.90	3	2	2	2	NA	18	2	2
	0.95	7	NA	4	4	NA	21	4	NA
	1.00	NA	NA	17	16	NA	26	14	NA
Chess	0.80	NA	21	20	29	NA	44	23	NA
	0.90	NA	NA	33	39	NA	53	33	NA
	0.95	NA	NA	47	46	NA	60	45	NA
	1.00	NA	NA	124	72	NA	90	113	NA
Mushroom	0.80	NA	47	29	32	NA	39	31	NA
	0.90	NA	NA	46	47	NA	53	47	NA
	0.95	NA	NA	62	62	NA	66	61	NA
	1.00	NA	NA	120	110	NA	120	105	NA
Paleo	0.80	NA	84	86	83	NA	83	83	NA
	0.90	NA	109	110	107	NA	107	106	NA
	0.95	NA	125	127	122	NA	122	122	NA
	1.00	NA	NA	151	139	NA	139	145	NA
Zoo	0.80	NA	7	9	9	NA	14	9	NA
	0.90	NA	16	13	13	NA	18	13	NA
	0.95	NA	NA	17	17	NA	21	16	NA
	1.00	NA	NA	30	25	NA	28	25	NA

These values are displayed for the selected real datasets. If a particular algorithm computed less factors than the value of k, which happens particularly for PaNDa and PaNDa+, we display the value of $c(k)$ corresponding to the collection of all factors computed by the algorithm. Note that when observing results on real datasets, one has to keep in mind that some datasets are very specific with respect to how the algorithms behave when factorizing them. For example, all the algorithms except for 8M and PaNDa(+) obtain the same collection of the first k factors for $k = 1, 2, 5,$ and 10 on the Paleo data, which is a very untypical behavior caused by the fact that this dataset "strongly offers" the factorization revealed by almost all algorithms. Overall, however, one may observe that Asso performs best from the DBP viewpoint, which is natural because Asso has been designed to solve the DBP. On the other hand, neither PaNDa nor PaNDa+, which are claimed by their authors [14,15] to perform greatly for DBP, display a particularly good behavior. For some datasets, their performance is only slightly worse than that of Asso; in several cases, however, these two algorithms are considerably weak. As is seen from Table 2, it happens often that the coverage by PaNDa(+) stops to increase after only a very small number of factors is generated, hence PaNDa(+) is not capable of explaining any larger portion of input data in general. In addition to Asso, the purely greedy strategy employed by GreConD results in a performance which is slightly worse from the DBP view compared to Asso but is comparable to it as a rule. A little worse than GreConD fares, on the one hand, the old 8M algorithm which displays a specific behavior, and, on the other hand, the group of algorithms represented by GreEss, NaiveCol, and to some extent also Hyper, which are basically designed for producing highly precise factorizations.

Let us now consider Table 3 which focuses on the AFP view. For each of the examined algorithms, the table displays for each particular dataset the number k of factors needed to achieve the prescribed value c of coverage for $c = 0.8, 0.9, 0.95,$ and 1. As is clearly seen, higher coverage values are practically unattainable by PaNDa and PaNDa+, which along with the relatively poor behavior of PaNDa(+) from the DBP view leaves PaNDa(+) as the overall worst of the examined algorithms. Another algorithm which is clearly not suitable from the AFP view is 8M. On the other hand, even though Asso has been designed for the DBP, it performs reasonably well in computing highly precise decompositions. The results in the table are indicative of an important property, namely the ability of an algorithm to compute exact decomposition for any input



(a) Emea (b) Chess

(c) Mushroom (d) Zoo

Fig. 4. Coverage quality of the first k factors on selected real-world datasets.

matrix. Which algorithms have this ability is shown in the following table; we refer the reader to the respective papers for justification of this property:

8M	Asso	GreConD	NaiveCol	PaNDa	Hyper	GreEss	PaNDa$^+$
no	no	yes	yes	no	yes	yes	no

The algorithms which are capable of computing exact decompositions all perform well from the AFP view and we refer e.g. to [4] for a detailed comparison.

The two tables above demonstrate our main point in this section, namely the fact that the DBP and the AFP view represent two basic views using which one may assess quality of BMF algorithms. While some algorithms perform well from the DBP view but not so well from the AFP view, for some the situation is the opposite. This is also shown in Fig. 4 which displays the graphs of the coverage function of the observed algorithms on four selected datasets: One may observe that the coverage function of some of the algorithms grows steeply at the beginning (good performance from the DBP view) but then stops, while for some other algorithms, coverage eventually reaches the value close to 1 representing full coverage (good performance from the AFP view) but grows slowly at the beginning.

Tables 2 and 3 thus present the characteristics relevant to the DBP view and the AFP view that may directly be observed from a factorization of each individual benchmark dataset obtained by a given algorithm. Table 4, however, still makes a significant step forward: It presents for a given factorization of each individual dataset obtained by a given algorithm a value of our quality metric. That is, instead of observing for a given factorization computed by a given algorithm a large number of relevant values (i.e. the coverage values relevant to the DBP view and thus corresponding e.g. to $k = 1, ..., k = 10$, and the numbers of factors relevant to the AFP view, i.e. corresponding to e.g. to $c = 0.8, ..., c = 1$), the metric aggregates all these observed values into a single number—the value of the metric for the given factorization. Note that the metric takes into account not only the values displayed in Tables 2 and 3 but rather all the observable values as explained in the description of the design of our metric. The metric depends on the parameter w which says how much emphasis the metric puts on the DBP view and the AFP view. We thus present three values corresponding to $w = 1$ (full emphasis is on the DBP view), $w = 0$ (full emphasis on AFP), and $w = 0.5$ (both the DBP and AFP are taken into account). One would hence use the setting with $w = 0.5$ of the metric when a factorization is sought which is good both from the DBP view and the AFP view. For example, consider the factorizations of Domino obtained by Asso and GreConD. We see from Table 2 that Asso and GreConD have comparable performances from the DBP view, and from Table 3 that GreConD performs considerably better from the AFP view. In particular, Asso is not capable to compute exact factorization, which is indicated by N/A in its

Table 4
Quality metric: AFP view ($w = 0$); DBP view ($w = 1$); combined view ($w = 0.5$).

Dataset	w	8M	Asso	GreConD	NaiveCol	PaNDa	Hyper	GreEss	PaNDa$^+$
DBLP	0.0	0.010	0.081	0.018	0.081	0.005	0.081	0.081	0.005
	0.5	0.081	0.183	0.158	0.183	0.086	0.183	0.183	0.086
	1.0	0.153	0.286	0.298	0.286	0.168	0.286	0.286	0.168
Domino	0.0	0.087	0.303	1.083	1.090	0.037	0.221	1.090	0.027
	0.5	0.224	0.492	0.881	0.789	0.315	0.347	0.789	0.297
	1.0	0.361	0.680	0.680	0.488	0.593	0.474	0.488	0.567
Emea	0.0	0.015	0.128	0.122	0.228	0.004	0.223	0.228	0.004
	0.5	0.138	0.219	0.213	0.226	0.082	0.224	0.226	0.075
	1.0	0.260	0.310	0.304	0.224	0.160	0.224	0.224	0.147
Firewall 1	0.0	0.200	0.207	1.569	1.564	0.003	0.249	1.573	0.094
	0.5	0.483	0.504	1.171	1.167	0.067	0.148	1.172	0.207
	1.0	0.765	0.802	0.772	0.770	0.131	0.048	0.770	0.321
Heatlhcare	0.0	0.122	0.100	1.135	1.154	0.142	0.602	1.173	0.034
	0.5	0.447	0.447	0.963	0.973	0.462	0.643	0.983	0.335
	1.0	0.771	0.794	0.792	0.792	0.782	0.684	0.792	0.636
Chess	0.0	0.032	0.058	-0.150	0.325	0.028	0.053	-0.002	0.048
	0.5	0.323	0.313	0.101	0.277	0.265	0.133	0.172	0.234
	1.0	0.615	0.568	0.352	0.229	0.502	0.212	0.347	0.421
Mushroom	0.0	0.020	0.050	0.344	0.408	0.009	0.324	0.413	0.027
	0.5	0.167	0.194	0.309	0.335	0.137	0.257	0.353	0.151
	1.0	0.314	0.338	0.274	0.263	0.264	0.190	0.292	0.274
Paleo	0.0	0.006	0.070	0.027	0.089	0.001	0.089	0.075	0.001
	0.5	0.023	0.078	0.056	0.087	0.013	0.087	0.080	0.012
	1.0	0.041	0.085	0.085	0.085	0.025	0.085	0.085	0.024
Zoo	0.0	0.028	0.086	0.212	0.350	0.018	0.204	0.353	0.008
	0.5	0.203	0.298	0.331	0.367	0.191	0.232	0.393	0.132
	1.0	0.377	0.510	0.451	0.383	0.364	0.259	0.432	0.256

entry for $c = 1$. These observations are succinctly expressed by the corresponding metric's values in Table 4: for $w = 1$ (pure DBP view), the metric's values are the same, for $w = 0$ (pure AFP view), the value for GreConD is much higher than that for Asso, and for $w = 0.5$ (combined view), the value for GreConD is correspondingly still significantly higher than that for Asso.

Note in addition that the values of our metric need to be interpreted relative to each particular data: While some datasets are naturally amenable to factorization, other datasets are not. Hence, for some datasets the values of our quality metric are considerably higher than for other datasets. As a consequence, it makes sense to compare the values of our metric for the various algorithms only for a given particular dataset rather than across the datasets. Secondly, as with any metric, our quality metric represents a kind of aggregation of several aspects of quality, and thus represents a simplification. If a detailed comparison of the quality of factorizations by two particular algorithms on a given dataset is desired, one needs to examine and compare these factorizations factor by factor. However, when a rough indication is sufficient, the values of our quality metric will serve the purpose. Indeed, one may observe that the values of our quality metric for the particular datasets nicely reflect the considerably more detailed description of the factorizations reported in Tables 2 and 3.

5. Conclusions

In this paper, we addressed a grossly neglected problem, namely assessment of quality of BMF algorithms. In particular, we identified key aspects of this important problem and proposed a quantitative way to assess the quality of BMF algorithms. We argued that quality assessment of BMF algorithms is not being paid proper attention in the BMF research and that to a considerable extent a reason for this is a lack of research on applications of BMF in machine learning. This observation is surprising in view of generally recognized usefulness of decomposition methods for real-valued matrices in machine learning, and is also in contrast with the large amount of existing work on BMF algorithms. We identified two basic views for quantitative assessment of quality. We proposed a particular metric based on these viewpoints, which expresses a natural requirement for a good algorithm for factorizing Boolean matrices. The experimental evaluation we conducted demonstrates that the proposed quality metric is reasonable in that it corresponds to prior experimental evidence as well as to the expectations derived from the properties of various factorization algorithms.

Further research in quality assessment of BMF algorithms is clearly desirable and should include examination of further methods for quality assessment. Since employment of BMF in machine learning and other areas is an important source of information regarding the quality of BMF algorithms, research on employment of BMF in these areas is needed. Last but not least, development of quality assessment methods for further criteria, such as the frequently discussed ability to deal with noise in Boolean data by BMF algorithms, should be explored.

Acknowledgments

Supported by grant no. GA15-17899S of the Czech Science Foundation. We thank the anonymous reviewers for their comments.

References

[1] K. Bache, M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2013. http://archive.ics.uci.edu/ml
[2] R. Belohlavek, V. Vychodil, Discovery of optimal factors in binary data via a novel method of matrix decomposition, J. Comput. Syst. Sci. 76 (1) (2010) 3–20. (Preliminary version in Proc. SCIS & ISCIS 2006).
[3] R. Belohlavek, J. Outrata, M. Trnecka, Impact of Boolean factorization as preprocessing methods for classification of Boolean data, Ann. Math. Artif. Intell. 72 (1–2) (2014) 3–22.
[4] R. Belohlavek, M. Trnecka, From-below approximations in Boolean matrix factorization: geometry and new algorithm, J. Comput. Syst. Sci. 81 (8) (2015) 1678–1697.
[5] R.A. Brualdi, H.J. Ryser, Combinatorial Matrix Theory, Cambridge University Press, 1991.
[6] P. Chalermsook, S. Heydrich, E. Holm, A. Karrenbauer, Nearly tight approximability results for minimum biclique cover and partition, ESA (2014) 235–246.
[7] W.J. Dixon (ed.), BMDP Statistical Software Manual, University of California Press, Berkeley, CA, 1992.
[8] A. Ene, et al., Fast exact and heuristic methods for role minimization problems, in: Proc. SACMAT, 2008, pp. 1–10.
[9] B. Ganter, R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer, Berlin, 1999.
[10] F. Geerts, B. Goethals, T. Mielikäinen, Tiling databases, in: Proc. Discovery Science, 2004, pp. 278–289.
[11] J. Hromkovic, Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics, Algorithmics for Hard Problems, Springer, Berlin, 2010.
[12] K.H. Kim, Boolean Matrix Theory and Applications, M. Dekker, NY, 1982.
[13] H. Lu, J. Vaidya, V. Atluri, Y. Hong, Constraint-aware role mining via extended boolean matrix decomposition, IEEE Trans. Depend. Secure Comput. 9 (5) (2012) 655–669.
[14] C. Lucchese, S. Orlando, R. Perego, Mining Top-K Patterns from Binary Datasets in Presence of Noise, SIAM DM, 2010, pp. 165–176.
[15] C. Lucchese, S. Orlando, R. Perego, A unifying framework for mining approximate top-k binary patterns, IEEE Trans. Knowl. Data Eng. 26 (12) (2014) 2900–2913.
[16] P. Miettinen, The Boolean column and column-row matrix decompositions, Data Min. Knowl. Discov. 17 (2008) 39–56.
[17] P. Miettinen, Sparse Boolean matrix factorizations, in: Proc. IEEE ICDM, 2010, pp. 935–940.
[18] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, H. Mannila, The discrete basis problem, IEEE Trans. Knowl. Data Eng. 20 (10) (2008) 1348–1362. (Preliminary version in Proc. PKDD 2006).
[19] P. Miettinen, J. Vreeken, Model order selection for Boolean matrix factorization, in: Proc. ACM SIGKDD, 2011, pp. 51–59.
[20] S.D. Monson, S. Pullman, R. Rees, A survey of clique and biclique coverings and factorizations of (0,1)-matrices, Bull. ICA 14 (1995) 17–86.
[21] D.S. Nau, Specificity Covering, Technical Report CS-1976-7, Duke University, 1976.
[22] D.S. Nau, G. Markowsky, M.A. Woodbury, D.B. Amos, A mathematical analysis of human leukocyte antigen serology, Math. Biosci. 40 (1978) 243–270.
[23] J. Outrata, Boolean factor analysis for data preprocessing in machine learning, in: Proc. ICMLA, 2010, pp. 899–902.
[24] B. Pontes, R. Giráldez, J.S. Aguilar-Ruiz, Biclustering on expression data: a review, J. Med. Inform. 57 (2015) 163–180.
[25] G. Schmidt, Relational Mathematics, Cambridge University Press, 2011.
[26] L. Stockmeyer, The Set Basis Problem is NP-Complete, Technical Report RC5431, IBM, Yorktown Heights, NY, USA, 1975.
[27] N. Tatti, T. Mielikäinen, A. Gionis, H. Mannila, What is the dimension of your binary data? in: Proc. IEEE ICDM, 2006, pp. 603–612.
[28] J. Vaidya, V. Atluri, Q. Guo, The role mining problem: finding a minimal descriptive set of roles, in: Proc. SACMAT, 2007, pp. 175–184.
[29] Y. Xiang, R. Jin, D. Fuhry, F.F. Dragan, Summarizing transactional databases with overlapped hyperrectangles, Data Min. Knowl. Discov. 23 (2011) 215–251. (Preliminary version in Proc. ACM KDD 2008).